LIM: An LLM-Independent Middleware for Robust Slot Filling in LLM-Augmented Chatbots Architecture

Gabriel Oliveira
Universidade Estadual do Ceará
Fortaleza, Brazil
gabrielluiz.barros@aluno.uece.br

Joaquim Ribeiro
Universidade Estadual do Ceará
Fortaleza, Brazil
joaquim.emanuel@aluno.uece.br

Alan Bandeira
Universidade Estadual do Ceará
Fortaleza, Brazil
alan.portela@uece.br

Paulo Henrique Maia
Universidade Estadual do Ceará
Fortaleza, Brazil
pauloh.maia@uece.br

Abstract

Developing chatbots that leverage Large Language Models (LLMs) for Natural Language Understanding (NLU) tasks such as intent recognition and slot filling has become standard in conversational agent engineering. However, depending solely on LLM outputs introduces risks including output inconsistency, vendor lock-in, and cost escalation, especially as inputs become more complex or ambiguous. To tackle these problems, this paper introduces LIM (LLM-Independent Middleware), a modular component that extends reference chatbot architectures to enhance LLM-based NLU through context-aware slot filling. LIM refines user input and exploits system context while maintaining a decoupled design that prevents reliance on specific models or providers. We evaluated LIM using a task-oriented chatbot that translates natural language into SQL queries, across multiple foundational models and 457 messages, alongside more 731 synthetic messages. Results show homogeneous gains, with F1-score improving from 64.39% to 78.71% and accuracy from 59.58% to 71.91%, alongside a 30.5% Error Reduction Rate. LIM also increased model homogeneity, raising F1-score Parity Performance from 37.43% to 80.10%, while enabling smaller models to perform competitively, with notable improvements in parameter efficiency (e.g., +79.59% for Flan-T5-base). These findings demonstrate, through a representative case study (DoME), LIM's potential to enhance slot-filling correctness and ensure robust, homogeneous behaviour across heterogeneous LLM ecosystems, serving as a promising initial validation of our architectural model.

Keywords

Large Language Models (LLMs), Conversational Agents, Natural Language Understanding (NLU), Slot Filling, Middleware Architecture, Model Independence, Robustness and Homogeneity, Task-Oriented Chatbots Architecture

1 Introduction

ELIZA [22], the first chatbot system, inspired by Alan Turing's work, introduced Natural Language Understanding (NLU) as a new computing domain [10]. Early chatbots relied on intent classification and predefined phrase correlations, requiring retraining whenever unseen inputs degraded performance[19]. The emergence of Transformers Architecture and Large Language Models (LLM) enabled more robust conversational chatbots, leading to systems such as ChatGPT[1] from OpenAI, Gemini[2] from Google and Anthropic's Claude[3].

In chatbot architectures LLMs may play different architectural roles [1, 17, 24], usually aiding in phases of User Message Analysis (UMA), such as intent identification and slot filling. Specifically, slot filling consists in the process of breaking user input into small pieces and analysing them to identify and understand the meaning of the message, which can be realized through different methods, such as text normalization, tokenization and part-of-speech tagging [10, 16]. Although LLMs substantially improve NLP tasks[7], models with this level of intricate complexity and extensive resources may introduce challenges from the software development perspective [26]. As discussed by [11] and [32], LLMs can make several mistakes and compromise the execution of systems that are not prepared to deal with unexpected errors. Modern applications often depend on external providers, such as AWS Bedrock[4] and Hugging Face[5], which simplifies deployment but increases dependence on third-party services, and risks related to vendor lock-in [18] and user's privacy [13]. These risks encompass not only the provider's availability and operational stability but also unexpected cost escalations, changes in service terms, model version deprecation and potential performance degradation over time [3, 33]. In such scenarios, migrating to a new model or service becomes a safeguard; however, this migration may be non-trivial. It typically demands recalibration of both input prompts and response-handling logic [15], as different models, even with identical prompts, may produce divergent outputs [20], as illustrated in Figure 1.

To handle these problems, we propose the Model-Independent Middleware, from now onwards referenced as LIM, a slot filling [10] architectural component whose main objective is to evaluate and handle LLMs unexpected behaviours in terms of message interpretation. LIM provides a slot filling strategy capable of reducing model inconsistency from user's input, bridging the chatbot core slot-filling components and the LLM, enabling model independence.

We evaluated our solution by answering the following research questions: **(RQ1)** How the NLU phase can benefit from a context-aware slot-filling component, regarding model response correctness

[1] https://chatgpt.com/
[2] https://gemini.google.com/
[3] https://claude.ai/
[4] https://aws.amazon.com/bedrock/
[5] https://huggingface.co/

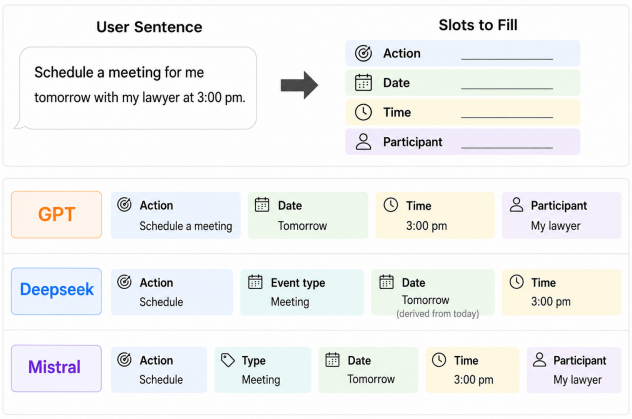

Figure 1: Example of slot extraction performed by different LLMs. the scheduling request was decomposed into slots Action, Type, Date, Time and Participant.

and homogeneity? This includes examining whether the system achieves higher accuracy compared to the unrestricted use of LLMs. **(RQ2)** Can a chatbot extended with LIM achieve correct slot-filling so that NLU quality is maintained? In particular, we investigate whether the middleware maintains quality across different LLMs **(RQ2.1)** and whether it can exchange models with different characteristics without harming NLU through LIM's actions **(RQ2.2)**. **(RQ3)** To what extent LIM is able to improve DoME's slot filling capabilities in a synthetically generated dataset?

Our approach was evaluated on a DoME architecture exemplar, a chatbot case study proposed by [8]. DoME uses Large Language Models (LLMs) and Natural Language Processing (NLP) to enable non-technical users to evolve a domain model through CRUD operations at runtime.

The remainder of this paper is as follows: section 2 Background, section 3 Methodology, section 4 Evaluation, section 5 Results and discussion, section 6 Related Work, section 7 Conclusion, section 8 Threats to Validity and section 9 Availability of Artifacts.

2 Background

2.1 Large Language Models

Large Language Models (LLMs) are statistical models based on textual datasets that use token distributions and parameters to generate responses [25]. They use as a main form of input a set of instructions, mostly presented in natural language, with the power to influence other following interactions and build a context, which are called a prompt [28]. However, there are still difficulties in creating appropriate prompts and achieving the expected results, such as Prompt Brittleness, that is when variations of the prompt syntax can result in drastic changes on the output [11]. and hallucinations, that cause factual incoherence and deviations from the original user input [30].

2.2 Chatbot Architecture

A chatbot is a conversational computer program that uses Natural Language Processing (NLP) to communicate in human language [2]. They are used in various forms throughout the areas of technology, like remote assistance, personal assistants, robots and smart devices. The first bots were developed in the 60s, for example, ELIZA, developed in 1966 by Joseph Weizenbaum at MIT Artificial Intelligence Laboratory [22], after that the advancements in chatbot development made these programs more attendant in people's lives, through applications such as digital assistants, like Siri[6] from Apple, or Google Assistant[7].

A general architecture of chatbot applications is composed of five main components, namely User Interface Component (UIC), a User Message Analysis (UMA) component, a Dialogue Management Component (DMC), a Backend Component, and a Response Generation Component (RGC) [2].

The UIC receives the user request in text or voice via a dedicated application, such as Telegram[8] or Whatsapp[9], and then the UMA component extracts the user request from the UIC to identify and extract the user's intent and involved entities. NLP is then used through three steps, first being Dialogue Act Classification, which tries to classify the dialogue as a question, affirmation, statement or another kind of dialogue act, the second step is Intent Classification, who identifies the user's primary objective, the intention, which is generally related to the domain [2], and at the end, there is the Slot Filling.

The DMC is responsible for controlling and managing the context of the conversation, and typically contains Ambiguity Handling, Data Handling and Error Handling modules. After the identification of intent, the chatbot proceeds to the next steps, which may need fetch information from the backend, through external APIs or Databases, and afterwards collecting the necessary information, the RGC works on a response to the user using one, or more, of the three models, Rule-based, Retrieval based and Generative-base [2].

3 LLM Independent Middleware

To solve the aforementioned problems and answer the RQs, we propose the LLM-Independent Middleware (LIM), a new error control and slot-filling component targeted at LLM powered chatbots, LIM operates as a middleware between the NLU and the LLM, enabling the system to perform well regardless of the model or, at least, ensuring that there are no scenarios where the user request is not met. LIM intercepts the response from the LLM, evaluates it, and treats it, making it adequate for the context model that the system understands, before returning it to the UMA component. Although LIM relies on established slot-filling techniques for information extraction, its role extends beyond slot filling, by providing an abstraction layer between the application and the LLM, ensuring interoperability response adaptation, and homogeneity across different models. In this paper, we present our solution in the form of an architectural model, showing how it can be adapted to specific chatbot contexts, and demonstrating the results over a specific case study (DoME) as an initial validation of its feasibility. Figure 2 shows the extension of the architecture introduced by [2] with LIM as an interface between the UMA component and the LLM Services component.

[6] https://www.apple.com/br/siri/
[7] https://assistant.google.com
[8] https://telegram.org/
[9] https://www.whatsapp.com/

Table 1: Summary of research questions

RQ	Objective	Metrics
RQ 1	How the NLU phase can benefit from a context-aware slot-filling component, regarding model response correctness and homogeny?	Precision, Recall, F1-Score and Accuracy
RQ 2.1	Can LIM maintain NLU quality consistently across different LLMs?	Performance Parity, Prediction Stability Index, Slot Agreement Rate and Coefficient of Variation
RQ 2.2	Can LIM enable the exchange of models without harming the NLU process?	F1-Score, Accuracy, Parameter Efficiency and Model Size Efficiency
RQ 3	To what extent LIM is able to improve DoME's slot filling in a novel and more heterogeneous dataset?	Precision, Recall, F1-Score and Accuracy

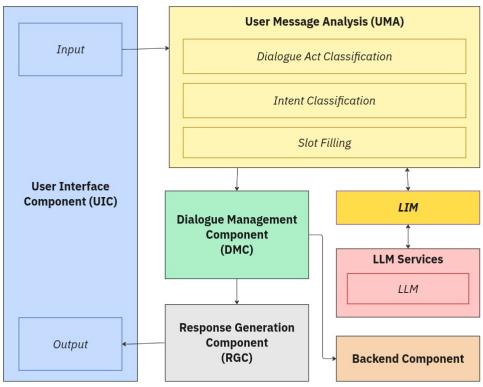

Figure 2: Chatbot architecture with LIM

LIM operates on the principle of a minimum acceptable response defined as an LLM-generated answer that, while not factually complete or semantically precise, satisfies a set of validations that preserve system integrity. These validations are tied to the context and domain of the chatbot.

Example of minimum acceptable response

In a slot-filling task where the system expects a pet name, one implementer might define validation as "output must contain exactly one noun phrase with no additional words," while another might accept any response that includes a plausible pet name substring. Thus, the same response, such as *"Pet"* (factually wrong) versus *"Ok, the name of the pet is: Luke"* (factually correct but poorly formatted), may be considered minimally acceptable in one implementation but rejected in another. This flexibility allows LIM to adapt to different integrity requirements.

One of LIM's main strengths is maintaining system integrity in adverse scenarios. Therefore, it seeks the best minimum acceptable response by applying context-aware rules for validation and message treatment. The minimum-acceptable-response problem could also be tackled by prompting strategies, however a single prompt template may behave differently across models, or even in the same one, depending on specifics and configurations [20].

3.1 LIM Architecture

The architecture comprises several integrated components: an API Layer that interfaces with the chatbot; an NLP Correction Center, which orchestrates all treatments and validations and communicates with slot-filling LLMs; a Treatment Component (TC) that implements context-aware NLP methods and heuristics and a Validation Component (VC) that enforces slot-specific constraints. The reference architecture model can be seen in Figure 3.

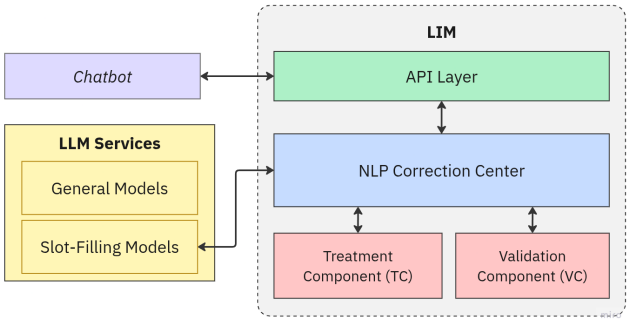

Figure 3: LIM's internal architecture

Pipelines. LIM adopts a pipeline-based architecture in which user requests and LLM responses are processed through a sequence of controlled stages. Each pipeline is responsible for a specific task, like treatment or validation. One of the main pipelines is the "Treatment pipeline", a set of rules and corrections defined by the context, which are coordinated by the NLP Correction Center to find the best-fitting answer. In this process, the message passes through a "Treatment loop" that verifies whether the answer is acceptable and treats it if not. The stage is repeated until the answer is considered minimally acceptable or until the available treatments are exhausted.

Validations. A consistent way to verify if the answer is "minimum acceptable" is through the use of validations, a pipeline containing a set of rules that checks the answer's validity to the slot-filling task and system domain. Those rules are defined by the system context and implementation to fit the model answers. Validations are standard NLP constraints and can implemented as string tests, such as checking if there are commas, if it fits some predefined pattern, or even a syntax check.

Treatments. If the model's response does not fit the restrictions

imposed by the validations, it has to undergo modifications to become satisfactory. These corrections are, from LIM's architecture perspective, system components responsible for post-processing the model answers. If the response is considered unfeasible, the treatment process must generate an alternative response that satisfies the minimum acceptable requirements. At the system level, there is a difference between the treatments that generate a brand new answer and the ones that only adapt the existing answer.

Common Treatments. Treatments that generate a new response are called common treatments, and these are executed only in cases where the system judges the model answer as a complete mistake in the aforementioned minimum acceptable point of view. If, even after all the treatments, the model answer stays outside the desired standards, the common treatments are called to extract an alternative response.

Mandatory Treatments. When the treatment does not change the answer completely, but only fits it in the constraints, we define it as a mandatory treatment. These types of treatments are performed obligatorily after a new response generation, whether generated by a model or a common treatment.

3.2 Data flow

From an architectural perspective, the system uses a standardized interface called treatment input, which serves as both input and output for each treatment and validation. This standardization ensures pipeline independence by eliminating the need to manage entity-specific entries while guaranteeing uniform data resources across all entities, thereby reinforcing system-wide consistency.

The dataflow begins when the API receives a model response in treatment input format, defined as a property based interface, which can be implemented as a JSON, for example. Then the treatment component applies mandatory treatments to the original answer and, if constraints are not satisfied, the system then iterates over common treatments, generating and validating new answers until correctness is achieved. A valid answer is returned in treatment data format with an acceptance flag; if all treatments fail, the system returns the best attempt with the acceptance flag set to false.

4 Evaluation

This section presents the experimental setup employed to assess the proposed approach. We describe the case study, the dataset characteristics, the evaluation metrics, and the experimental protocol adopted to ensure reproducibility and validity of the results.

4.1 Case Study

To evaluate LIM, we extended the DoME chatbot architecture, which is a tool for interpreting user input and evolving domain models of CRUD-based systems, through data model evolution at runtime [8]. This evaluation constitutes a case study aimed at investigating and validating our claims within the scope of the considered scenario. Given its specific domain and proportions, the results should therefore be interpreted as evidence supporting our claims in this context, rather than as a general evaluation across diverse chatbot designs.

Although the original DoME work does not explicitly describe a slot-filling approach, both its architectural model and reference implementation decompose user requests into four semantic slots, operation, entity, attributes, and filters. This decomposition follows the principles of task-oriented slot filling, where unstructured input is mapped into structured representations through slot extraction, making DoME a representative case study for evaluating our middleware. This interpretation is based on two complementary perspectives. First, the original DoME publication describes an architectural model that decompose user requests into four different slots. Second, the available DoME implementation follows the same strategy of decomposition in its NLU processing pipeline. The implementation analyzed in this study is available in the project repository.

LIM was integrated into DoME through slot-specific pipelines corresponding to the four semantic slots: *operation* (OPR), *entity* (ENT), *attributes* (ATT), and *filters* (FIL). Each pipeline applies context-aware validation and treatment strategies according to the characteristics of the corresponding slot.

LIM customizes its treatments according to the characteristics of each slot. For the OPR slot, operation filtering restricts outputs to valid CRUD commands, such as *read* or *create*. ENT slots validators verify token length and contextual deviation, and additionally preserve semantic consistency with the user input. The ATT and FIL slots correspond to multi-token pipelines, which require additional processing due to the difficulty LLMs have in preserving semantic structure [6]. The ATT pipeline combines response treatments, validators, and lexical analysis to identify attribute values, including the handling of auxiliary words.[10] The FIL pipeline applies contextual keyword detection and noun identification to capture refinement expressions, such as *where*, *with*, and *when*, iteratively extracting keyword–noun combinations from the remaining input.

4.2 Dataset

To conduct the evaluation, we constructed a composite dataset from two sources. The first source is the original dataset provided by DoME [8], consisting of 556 real user messages, their expected slots, and the slots identified by an LLM. As previously established, LIM was adapted to DoME's functionality related to CRUD operations, implying that non-CRUD messages (e.g., greetings, emojis, and malformed inputs) were filtered out, yielding 457 valid messages.

Although the original DoME dataset is validated, it inherently reflects the characteristics of a controlled and instruction-driven data collection process. Consequently, the resulting messages exhibit a degree of bias toward the experimental conditions under which they were produced. To reduce this bias and to avoid coupling the evaluation exclusively to the experimental setting and domain assumptions of DoME [8], we generated an additional dataset with greater domain diversity. Specifically, to further extend and stress the evaluation of LIM under a wider range of contextual conditions and slot configurations, we developed a secondary data source through LLM-based data augmentation.

The inclusion of this secondary dataset is particularly important because improvements observed exclusively on the original DoME benchmark could be influenced by the characteristics and assumptions of its underlying data collection process. Therefore,

[10] https://universaldependencies.org/u/pos/AUX_.html

evaluating LIM on a more heterogeneous dataset allows us to investigate the extent to which its benefits generalize beyond the original domain and interaction patterns. This motivation is formalized by **RQ3**, which examines whether LIM is capable of improving DoME's slot-filling correctness in novel contexts characterized by greater linguistic and domain variability.

For this purpose, we adapted the pipeline proposed in [14] to the context of chatbot user messages, by performing the few-shot prompting technique, with the user interaction dataset as source of examples, on Google's Gemini 2.5 Flash [9] model. The process produced 774 new messages, which underwent filtering to remove duplicates, incomplete entries and standardize formatting. This resulted in 731 synthetic messages. The entire process is illustrated in Figure 4.

Table 2 summarizes the main characteristics of the resulting dataset. The *Messages* metric corresponds to the total number of user utterances available in each source. The *Different Entity Quantity* metric represents the number of distinct entity types (i.e., domains or slot categories) appearing in the dataset. Finally, messages are categorized according to their expected slot complexity. *Single-slot messages* correspond to utterances requiring at most one attribute or filter to be identified, whereas *multi-slot messages* require the extraction of two or more attributes and/or filters.

Table 2: Summary of the original DoME dataset, the synthetic dataset, and the resulting evaluation corpus.

Metric	Original	Synthetic
Messages	457	731
Unique Domain Quantity	38	58
Single-Slot Messages	268	543
Multi-Slot Messages	189	188

The synthetic dataset contains a substantially larger number of distinct entities than the original DoME dataset (58 versus 38), indicating broader domain coverage and greater semantic variability. This increase in entity diversity is particularly relevant to RQ3, as it introduces a wider range of vocabulary, contexts, and slot configurations, allowing LIM to be evaluated beyond the assumptions of the original benchmark. Furthermore, the synthetic dataset is predominantly composed of single-slot messages, whereas the original dataset presents a more balanced distribution between single-slot and multi-slot requests. This difference suggests that the two sources capture complementary interaction patterns: the original dataset emphasizes more complex requests involving multiple constraints, while the synthetic dataset focuses on a larger variety of simpler requests across diverse domains.

After this generation, the data was organized into a table organized in five foundational models: Llama3[11], Gemma3[12], Qwen3[13], Flan-T5[14] and Mistral[15]. These models were selected based on three main criteria: (i) their status as fundamental open models, (ii) the feasibility of running them locally, and (iii) our hardware and processing constraints. The data were categorized into real and synthetic subsets. Although both subsets are collectively treated as a single dataset, they are analyzed separately due to differences in validity: real data consist of validated instances, whereas synthetic data do not undergo the same validation process. This distinction allows for a more appropriate interpretation of the results, considering the differing levels of reliability associated with each subset.

Moreover, evaluating different models that can be efficiently executed in environments with limited computational resources or constrained hardware while still achieving satisfactory performance was essential to address the research questions. Llama3, developed by Meta, contains 8 billion parameters and represents a strong general-purpose foundation model; Gemma3, developed by Google, is a lightweight model optimized for text understanding and reasoning; Qwen3, developed by Alibaba Cloud, is a versatile multilingual model designed for reasoning and instruction-following; Flan-T5, also from Google, is a reference model specialized in text generation; and Mistral, from MistralAI, includes 7.25 billion parameters and is known for its efficiency and high-quality text generation capabilities.

All models were evaluated under a zero-shot setting, without fine-tuning or parameter calibration. This design choice stems from the interoperability-oriented nature of our experiments, which aim to maximize the ease of model interchangeability within a system, avoiding the need for any calibration regarding configurations, parameters, or prompt engineering. Moreover, the zero-shot option was the standard communication structure used between DoME and LLMs. This diversity of models enables a broader and more reliable assessment of the proposed approach, reinforcing its generalization capacity and cross-model robustness. By validating the results across architectures from different vendors and parameter scales, we ensure that the evaluation is not biased toward a specific model family or design paradigm. All specifications of the models that will be useful for our evaluations are available in Table 3, other technical specifications, such as temperature and prompts, are available at the experiment repository.

Table 3: Comparison of Model Specifications

Model	Parameters (B)	Size (GB)
Flan-T5-Base	0.25	0.43
Llama3	8.00	4.70
Gemma3	4.30	3.30
Mistral	7.00	4.40
Qwen3	8.20	5.20

4.3 Metrics

To evaluate the proposed middleware and address the research questions, we employ a set of quantitative metrics grouped according to RQ1, RQ2, RQ3 and their sub-questions. It is important to note that DoME without LIM is used as our baseline, whereas DoME with LIM represents the novel configuration under evaluation. In order to adopt the classical binary evaluation framework [21, 27] based on True Positives (TP), True Negatives (TN), False Positives

[11] https://llama.meta.com/
[12] https://ollama.com/library/gemma3
[13] https://qwenlm.ai/
[14] https://huggingface.co/docs/transformers/model_doc/flan-t5
[15] https://mistral.ai/

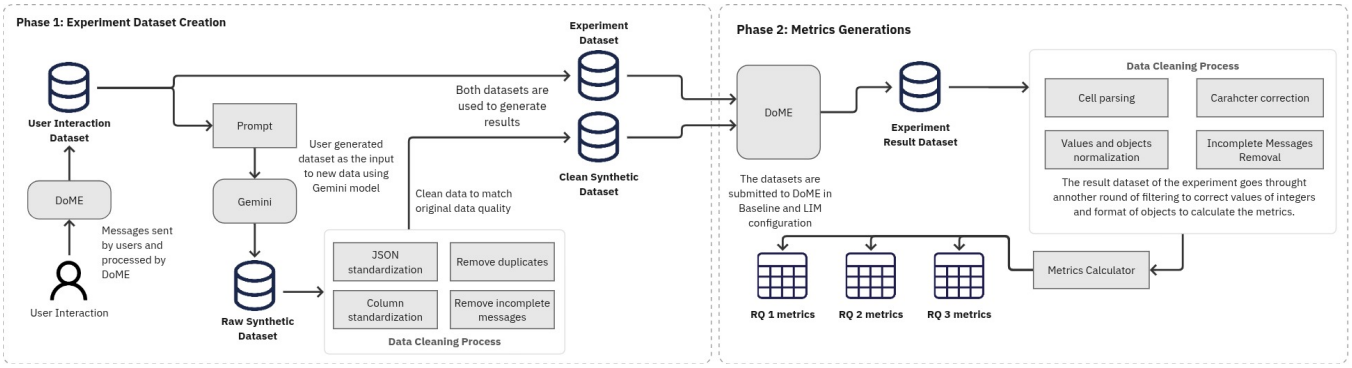

Figure 4: Dataset generation and metrics development process

(FP), and False Negatives (FN), we adapted the slot filling task to a binary decision setting. Specifically, a True Positive is counted when a slot is processed and correctly identified; a False Positive occurs when a slot is processed but incorrectly classified; a True Negative corresponds to a slot that is not processed when its absence is indeed the correct outcome; and a False Negative arises when a slot is not correctly processed even though it should have been extracted. This formulation enables us to evaluate slot filling within the same binary classification framework, allowing the use of standard metrics such as precision, recall, and F1-score.

For RQ1, which investigates the benefits of an LLM-based slot filling component on the NLU module, we evaluate a range of accuracy-oriented metrics that capture different aspects of slot extraction correctness. We categorize the evaluation measures into two groups: base metrics, which are well established in the literature [4, 26, 31] and extracted from confusion matrix components, and derived metrics, which are computed using the base metrics to better capture the relationship between the evaluated configurations and to quantify the degree of influence that a model have in slot-filling identification task, given each base metric.

Base metrics: The **Precision** measures the proportion of correctly identified slots among all slots extracted by the system,

$$\text{Precision} = \frac{TP}{TP + FP},$$

indicating the reliability of the model's predictions. **Recall** quantifies the proportion of correctly identified slots among all ground truth slots,

$$\text{Recall} = \frac{TP}{TP + FN},$$

reflecting the model's ability to capture all relevant information. The **F1-score**, defined as the harmonic mean of precision and recall,

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

provides a single measure of overall slot extraction quality. Finally, **Accuracy** measures the overall proportion of correctly predicted slots, considering both true positives and true negatives,

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN},$$

representing the general correctness of the model across all slot prediction outcomes.

Derived metrics: To assess improvements over the baseline configuration, we define **Relative Improvement** as the percentage gain in a given metric achieved by LIM compared to the baseline system:

$$\text{RI} = \frac{\text{Metric}_{\text{LIM}} - \text{Metric}_{\text{baseline}}}{\text{Metric}_{\text{baseline}}} \times 100\%.$$

Similarly, the **Error Reduction Rate** measures the proportion of errors ($FP + FN$) eliminated with the LLM-based approach:

$$\text{ERR} = \frac{\text{Errors}_{\text{baseline}} - \text{Errors}_{\text{LIM}}}{\text{Errors}_{\text{baseline}}} \times 100\%.$$

Finally, a **Slot-based NLU Analysis (SBNA)** decomposes the evaluation by slot type, enabling the identification of which slots pose greater challenges to the model. This analysis involves computing the basic metrics for each individual slot, as well as their respective improvements, as defined previously.

For RQ2, which evaluates the middleware's ability to provide independence from specific LLMs, we adopt a set of metrics explicitly aligned with the sub-questions. The **Parity Performance (PP)** quantifies the degree of homogeneity in slot-filling correctness across different models for a given evaluation metric (e.g., F1-score). It measures how evenly models perform relative to each other, with values closer to 1 indicating that all models achieve nearly the same slot-filling correctness, and values closer to 0 suggesting large disparities. In essence, PP captures the degree of parity, or fairness, among models by comparing the range of metric values to the best performing model:

$$\text{PP} = 1 - \frac{\max_m \text{Metric}(m) - \min_m \text{Metric}(m)}{\max_m \text{Metric}(m)}.$$

A high PP thus reflects a stable and uniform behaviour across models, indicating that improvements or degradations are not concentrated in a specific model. Conversely, a low PP highlights variability in model slot-filling correctness, suggesting that some models benefit more from the configuration (e.g., LIM) than others. While PP captures this aggregate-level homogeneity, the **Prediction Stability Index (PSI)** provides a finer-grained perspective by quantifying agreement at the individual prediction level. Specifically, PSI quantifies the proportion of inputs for which two models produce identical

predictions:

$$\text{PSI} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y_i^{(m_1)} = y_i^{(m_2)}\}.$$

For each configuration, PSI is first computed pairwise for every combination of models. Thus, for n models, $\binom{n}{2} = \frac{n!}{2!(n-2)!}$ pairwise PSI values are obtained. To characterize overall inter-model agreement, these pairwise values are then averaged, providing an aggregate agreement rate across all model pairs. Therefore, while PSI is intrinsically a pairwise measure, its mean across all pairs summarizes the overall agreement among the evaluated models.

The **Slot Agreement Rate (SAR)** evaluates agreement at the slot level, computed as the proportion of slots predicted identically by different models. Finally, the **Coefficient of Variation (CV)** [29] provides a complementary perspective by capturing the relative dispersion of model slot-filling correctness. Unlike absolute measures of variability, the CV expresses how large the standard deviation is in relation to the mean, making it independent of the absolute scale of the metric. Higher CV values indicate greater inconsistency or sensitivity among models, while lower CV values reflect more stable and homogeneous behaviour.

To determine if a smaller LLM can replace a larger one via the middleware without significant slot-filling correctness degradation (RQ2.2), we introduce the **Parameter Efficiency (PE)**, that relates slot-filling correctness to the number of model parameters P:

$$\text{PE} = \frac{\text{Metric}}{P \times 10^{-9}},$$

while the **Model Size Efficiency (MSE)** relates slot-filling correctness to the model's size in GB S:

$$\text{MSE} = \frac{\text{Metric}}{S}.$$

Together, these metrics enable direct comparison of how effectively the middleware leverages models of different scales.

To answer RQ3, we compare the metrics obtained from the original dataset with those from generated one. With the evaluation metrics clearly defined and structured, we proceed with the experiments outlined in Section 4.1 to address the research questions.

5 Results

The results are organized according to the research questions (RQ1, RQ2 and RQ3). All metrics were computed as detailed in 4.3.

RQ1. The results comparing baseline and LIM slot-filling correctness are presented in Figure 5. The histogram demonstrates that LIM substantially improved the system's NLU capabilities. Specifically, the increase in True Positives (TP) indicates that LIM enabled the model to correctly identify 1,405 additional slots compared to the baseline results.

Overall, the middleware led to improvements in slot-filling correctness across all evaluated metrics. The average precision increased from 61.98% to 80.04%, a RI of 29.13%, indicating that LIM made the NLU stage more reliable in identifying correct slot values. Similarly, recall rose from 67% to 77.42% (RI = 15.55%), showing that the middleware also enhanced the model's ability to capture a larger proportion of relevant information. As a result, the F1-score improved from 64.39% to 78.71% (RI = 22.23%), reflecting a balanced gain in both precision and recall. Finally, accuracy increased from

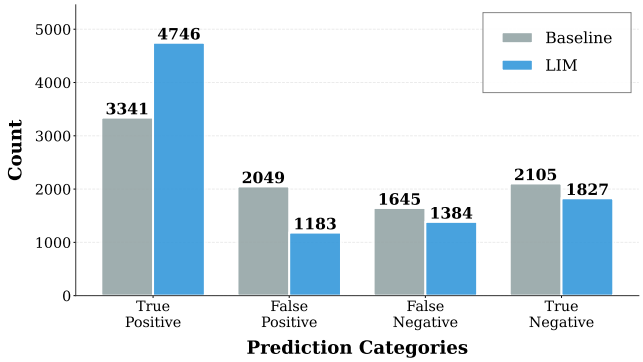

Figure 5: Baseline and LIM metrics on slot prediction

59.58% to 71.91% (RI = 20.69%), demonstrating a marked overall improvement in the slot-filling correctness of the model's predictions across all slots. Table 4 summarizes the results obtained by the baseline system and the proposed LIM.

Furthermore, we achieved a 30.5% error reduction rate, with 3,694 errors in the baseline, and 2,567 errors with LIM, a promising result that highlights the solution's capability in turning model mistakes into correct identified slots.

Table 4: Metrics results for Baseline and LIM (RQ1)

System	Precision	Recall	Accuracy	F1-score
Baseline	61.98%	67%	59.58%	64.39%
LIM	80.04%	77.42%	71.91%	78.71%
RI	**29.13%**	**15.55%**	**22.23%**	**20.69%**

In SBNA, we evaluated slot-level metrics to assess how the middleware affects system interpretation for each slot. Detailed results are available in Table 5.

For the *operation* slot, baseline correctness was 97.57% F1-score and 95.27% accuracy, improving to 98.84% and 97.72% with LIM (1.27% and 2.45% absolute gains). Despite near-ceiling baseline correctness, the results demonstrate LIM's effectiveness in refining model interpretations even when baseline correctness is already high, reducing residual ambiguity in operation classification.

For the *entity* slot, baseline correctness of 62.76% F1-score and 45.73% accuracy improved to 92.79% and 86.56% with LIM, representing relative improvements of 47.84% in F1-score and 89.28% in accuracy. These substantial improvements highlight LIM's enhanced contextual understanding and disambiguation capabilities for variable entity mentions.

The *attributes* slot showed baseline correctness of 11.82% F1-score and 36.06% accuracy, improving to 41.07% and 40.35% with LIM, corresponding to relative improvements of 247.46% in F1-score and 11.89% in accuracy. These results demonstrate LIM's capability to interpret fine-grained semantic details in this complex slot type.

For the *filters* slot, baseline correctness was 4.53% F1-score and 61.26% accuracy, improving to 12.43% and 63.01% with LIM, corresponding to relative improvements of 174.39% in F1-score and 2.85% in accuracy. Despite remaining low in absolute F1-score, these

improvements demonstrate LIM's positive influence even under challenging conditions.

Table 5: Comparison between Baseline and LIM correctness

System	Precision	Recall	Accuracy	F1-score
Baseline OPR	99.67%	95.56%	95.27%	97.57%
LIM OPR	98.98%	98.71%	97.72%	98.84%
RI OPR	**-0.69%**	**3.29%**	**2.57%**	**1.3%**
Baseline ENT	48.02%	90.55%	45.73%	62.76%
LIM ENT	90.56%	95.14%	86.56%	92.79%
RI ENT	**88.58%**	**5.06%**	**89.28%**	**47.84%**
Baseline ATT	10.56%	13.44%	36.06%	11.82%
LIM ATT	33.33%	53.49%	40.35%	41.07%
RI ATT	**215.62%**	**297.99%**	**11.89%**	**247.46%**
Baseline FIL	20.58%	2.54%	61.26%	4.53%
LIM FIL	93.75%	6.65%	63.01%	12.43%
RI FIL	**355.53%**	**161.81%**	**2.85%**	**174.39%**

Overall, these results demonstrate that LIM systematically improves slot-filling correctness across all slot types, with impact varying according to each slot's intrinsic predictability and linguistic complexity. LIM contributes meaningfully even in the most challenging cases, guaranteeing positive and consistent impact across the entire slot-filling process.

RQ2. To evaluate RQ2, we examined the middleware's ability to maintain stable slot-filling correctness across the five LLMs. Table 6 summarizes the results obtained for each model. Although there are small variations, all models achieved similar overall scores, demonstrating that the middleware effectively mitigates model-specific behaviours.

Table 6: Performance across different LLMs (RQ2.1)

Model	F1-score	Accuracy
Flan Baseline	38.62%	52.07%
Gemma3 Baseline	64.81%	75.21%
Llama3 Baseline	45.91%	67.77%
Mistral Baseline	33.39%	54.21%
Qwen3 Baseline	24.26%	48.63%
Flan LIM	69.36%	73.30%
Gemma3 LIM	61.44%	73.74%
Llama3 LIM	59.08%	72.64%
Mistral LIM	59.15%	71.60%
Qwen3 LIM	55.56%	68.27%
PP Baseline	37.43%	64.65%
PP LIM	**80.10%**	**92.58%**

The computed PP with LIM was 80.10% for the F1-score and 92.58% for accuracy, indicating a substantial reduction in slot-filling correctness disparities across models. Compared with the baseline, in which PP was considerably lower (37.43% for F1-score and 64.65% for accuracy), these results suggest that LIM reduces the system's dependence on model-specific behavior and provides a more consistent slot-filling correctness level across heterogeneous LLMs. This effect is particularly relevant to LIM's objective of providing a model-agnostic abstraction layer for the slot-filling pipeline.

The reduction in inter-model disparity, however, does not imply that LIM uniformly improves slot-filling correctness. Gemma3, which achieved the highest baseline correctness, experienced a marginal degradation under LIM, with F1-score decreasing from 64.81% to 61.44% and accuracy from 75.21% to 73.74%. This highlights a trade-off introduced by the middleware: output constraints and validation may slightly reduce the correctness of models that already produce reliable slot structures. Thus, LIM's standardization primarily reduces dependence on the characteristics of the underlying LLM rather than enforcing identical correctness levels across models.

This distinction is important for interpreting PP. The high PP values indicate that LIM establishes a more predictable slot-filling correctness boundary across different model families and parameter scales, rather than eliminating differences in their intrinsic capabilities. Conversely, the lower PP observed in the baseline reflects greater heterogeneity, with models such as Gemma3 and Llama3 substantially outperforming others. Thus, LIM contributes to reducing these disparities and making the slot-filling pipeline more homogeneous across heterogeneous LLMs, as illustrated in Figure 6.

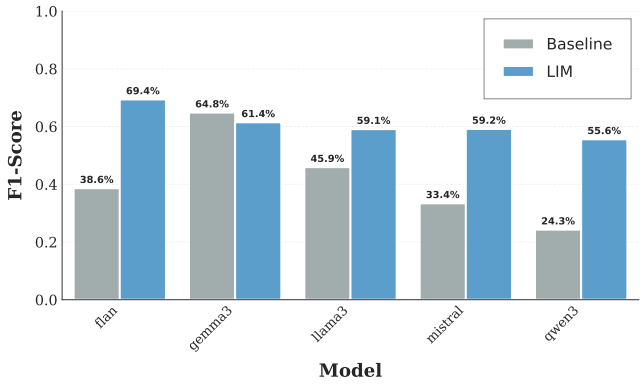

Figure 6: F1-Score comparison between models

With LIM, the average PSI increased from 5.32% to 24.10%, indicating that 24.10% of utterances received identical all-slot predictions across all models, representing substantial enhancement in inter-model agreement. The coefficient of variation of PSI decreased from 135.40% to 43.03%, reinforcing LIM's ability to standardize outputs and mitigate variability. While SAR reached 55.63% with LIM versus 47.67% without it, this similarity is misleading: slot matches were scattered across different utterances rather than forming consistent full-message alignments. PSI revealed more meaningful improvement, showing LIM fosters coherence at the message level with all slots jointly consistent.

Statistical analysis further confirmed standardization: accuracy CV decreased from 19.08% to 3.04%, F1-score CV from 36.91% to

8.48%, with absolute CV reductions of 44.09% and 25.75% for precision and recall, respectively.

Regarding RQ2.2, which investigates whether different size models can effectively perform without a bigger degradation, parameter efficiency (PE) metrics demonstrate that LIM significantly reduces the impact of model scale on slot-filling correctness. The highest PE_{F1} increased from 154.48 to 277.44 for Flan-T5-base (79.59% relative improvement), while Qwen3's PE_{F1} rose from 2.9585 to 6.7756 (129.04% relative improvement). Across all models, mean PE for accuracy and F1-score showed absolute growth rates of 36.33% and 71.75%, respectively, indicating that smaller models achieve disproportionately higher performance per parameter under the middleware framework.

Relative MSE improvements averaged 32.06% for accuracy and 64.76% for F1-Score (11.54 and 17.03 in absolute terms respectively). These results confirm that LIM enables smaller architectures to approximate the slot-filling correctness of larger models without proportional increases in computational requirements, memory footprint, or inference latency, supporting the viability of deploying compact models in resource-constrained environments while maintaining competitive slot-filling correctness standards.

RQ3. To answer RQ3, which investigates the ability of LIM to improve slot filling in synthetically generated datasets, we evaluated the middleware using an artificial dataset detailed in 4.2. The results (Table 7) demonstrate slot-filling Relative Improvement across all metrics in perspective of both datasets, confirming the robustness of LIM under diverse domains and slot configurations.

Table 7: Metrics results for Baseline and LIM on the original and synthetic dataset (RQ3)

System	Precision	Recall	F1	Accuracy
— Original Dataset —				
Baseline	61.98%	67%	64.39%	59.58%
LIM	80.04%	77.42%	78.71%	71.91%
RI	**29.13%**	**15.55%**	**22.23%**	**20.69%**
— Synthetic dataset —				
Baseline	68.90%	58.89%	63.50%	56.45%
LIM	75.47%	75.29%	75.38%	66.21%
RI	**9.53%**	**27.84%**	**18.70%**	**17.28%**

When evaluated on the synthetic dataset, which introduces greater domain and contextual variability, LIM maintained homogeneous gains across all evaluation metrics. Precision increased from 68.90% to 75.47%, recall from 58.89% to 75.29%, and F1-score from 63.50% to 75.38%, corresponding to relative improvements of 9.53%, 27.84%, and 18.70%, respectively. Accuracy also improved from 56.45% to 66.21% (RI = 17.28%). Results excel in recall, suggesting that LIM substantially enhances the ability to recover relevant slot information under more heterogeneous input conditions. These results indicate that the benefits provided by LIM generalize beyond the original benchmark, demonstrating robustness across different domains, contexts, and slot configurations.

Furthermore, as illustrated in Figure 7, slot-filling correctness across models remains highly homogeneous on the combined synthetic dataset, reinforcing LIM's robustness to data augmentation,

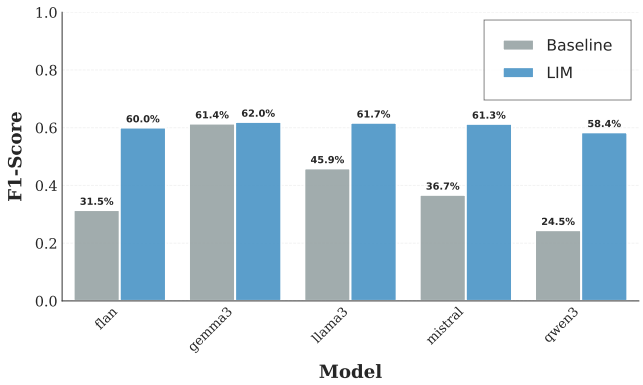

Figure 7: F1-Score comparison between models with the synthetic dataset

increased input variability, and distribution shifts. Parity Performance (PP) for F1-score increased from 39.90% in the baseline to 94.19% with LIM, an absolute gain of 54.29 percentage points. This result indicates that LIM substantially reduces inter-model disparities under more challenging and diverse data conditions. Although Gemma3 already achieved 61.4% without LIM, lower-performing models benefited more, resulting in a 3.6 percentage-point gap and demonstrating LIM's ability to promote consistent slot-filling correctness across heterogeneous models.

Overall, these results indicate that LIM preserves its slot-filling correctness gains under more variable data while strengthening generalization, robustness, and inter-model homogeneity. Beyond improving traditional performance metrics, LIM reduces slot-filling correctness disparities and promotes more consistent model behaviour under increased data variability and distribution shifts introduced by synthetic data. The substantial increase in Parity Performance further demonstrates this stabilizing effect across heterogeneous models and input conditions, supporting LIM's applicability in environments where synthetic data is used to expand training or evaluation corpora.

6 Related work

Rana et al. (2025) used black-box Knowledge Distillation from a large teacher LLM (70B+ parameters) to a smaller student model (Llama3 8B), achieving a 26% absolute F1 increase. While their fine-tuning approach enhances accuracy through slot induction and output refinement, it requires model optimization and does not ensure cross-architecture homogeneity. In contrast, LIM operates as an architectural standardization layer that decouples robustness from training methodology, enforcing correctness homogeneity through structured Validations and Treatments that actively mitigate model-specific variation.

Bhargav et al. (2024) presented a zero-shot slot-filling approach using smaller models (7-13B parameters) through instruction fine-tuning with task-specific data. While this data-centric optimization enhances individual model capabilities, it does not address architectural dependency and correctness instability across different LLMs. In contrast, LIM operates as a post-generation architectural

layer that enforces system-level homogeneity and standardization, mitigating output variance across heterogeneous language models.

Kamoi et al. (2024) introduced ReaLMistake, a benchmark that objectively evaluates LLM responses using binary criteria: logical acceptability, instruction-following, context faithfulness, and factual correctness. While similar to our work in using objective binary evaluation rules, their approach focuses on general LLM responses rather than slot-filling in conversational AI contexts.

7 Threats to Validity

This section discusses potential threats to validity, examining internal and external validity concerns that may impact the interpretation and generalizability of results from the LLM-Independent Middleware (LIM) evaluation.

7.1 Internal Validity

Internal validity concerns whether observed slot-filling correctness improvements can be confidently attributed to the LIM intervention rather than confounding factors or methodological artifacts.

Empirical Calibration of Treatment Pipelines: LIM treatments and validations were specifically tailored to DoME's requirements for translating natural language into structured SQL commands through empirically calibrated rules. While this yielded substantial improvements (29.13% relative F1-score increase), observed gains may be significantly driven by domain-specific calibration rather than LIM's general architectural principles.

Limited Domain Specificity: The evaluation focuses exclusively on the DoME chatbot domain for CRUD operations on relational databases. Slot-filling correctness disparities reveal LIM's effectiveness boundaries: while achieving near-ceiling correctness on simpler slots like "operation" (98.84% F1-score), the system showed unsatisfactory absolute slot-filling correctness on the semantically complex "filters" slot (12.43% F1-score), despite a substantial 174.39% relative improvement.

Slot-filling correctness Benchmark: This study does not incorporate direct comparisons with the original DoME benchmark due to substantially different experimental conditions and model specifications. However, validity is upheld by rigorous internal comparison between LIM-enhanced and baseline conditions, evaluated consistently across the same varied sample space, ensuring controlled assessment of LIM integration's effect.

7.2 External Validity

External validity addresses the generalizability of findings to alternative contexts, including different application domains, datasets, and language model architectures.

Reliance on Synthetic Data Generation: The evaluation dataset comprised 1,188 conversational messages: 457 authentic user interactions and 731 synthetically generated messages via Google Gemini 2.5 Flash [9]. While synthetic augmentation was necessary for statistical power and linguistic coverage, this introduces external validity threats, as synthesized data may not fully capture the complexity, variability, and characteristic errors of authentic human input.

However, this approach maintains scientific validity as the original dataset of 457 authentic messages represented a rigorously curated and validated corpus. All original messages had previously been processed and benchmarked against DoME system in our case study, establishing a verified slot-filling correctness baseline. The synthetic augmentation was specifically employed to extend the evaluation beyond this established foundation, enabling targeted assessment of the new component's capabilities on previously unseen message patterns and edge cases not encountered during baseline evaluation. This strategic expansion ensures comprehensive coverage while maintaining methodological integrity through the preserved original dataset's validated characteristics.

8 Conclusion

This paper introduced the LLM Independent Middleware (LIM), an architectural component designed to mitigate slot-filling performance inconsistency, model dependency, and vendor lock-in in slot-filling systems by decoupling system behaviour from the underlying LLM.

Experimental results obtained through our case study (DoME) demonstrated LIM's effectiveness across multiple dimensions. For slot-filling correctness (RQ1), LIM achieved a 22.23% relative improvement in F1-score and reduced errors by 30.5%, with the largest gains observed in semantically complex slots such as *attributes* (247.46%) and *filters* (174.39%).

For model independence (RQ2), LIM significantly improved homogeneity across architectures, increasing Parity Performance for F1-score from 37.43% to 80.10% and reducing variability (CV from 36.91% to 8.48%). It also enabled more efficient use of smaller models (RQ2.2), with notable gains in parameter efficiency (e.g., +79.59% for Flan-T5-base).

Under increased domain and contextual variability (RQ3), LIM maintained strong slot-filling correctness while improving robustness and inter-model agreement. On the synthetic dataset, LIM achieved relative improvements of 18.70% in F1-score and 17.28% in accuracy, while Parity Performance increased from 39.90% to 94.19%. These results indicate that LIM not only preserves its effectiveness under distribution shifts and heterogeneous inputs but also promotes more homogeneous behaviour across different models.

Overall, LIM provides a robust solution for slot-filling, ensuring homogeneous slot-filling correctness across models and data conditions while reducing dependence on specific LLMs.

Future work includes broader evaluations, multilingual settings, integration with emerging architectures, and analysis of computational overhead in production environments.

Artifact Availability

LIM code: https://github.com/JoaquimEmanuell/lim-research

Acknowledgements

This work was supported by CAPES through scholarships and by CNPq through undergraduate research scholarships. The authors gratefully acknowledge the invaluable support of these Brazilian funding agencies.

References

[1] Samuel Abedu, Ahmad Abdellatif, and Emad Shihab. 2024. LLM-Based Chatbots for Mining Software Repositories: Challenges and Opportunities. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software*

Engineering (Salerno, Italy) *(EASE '24)*. Association for Computing Machinery, New York, NY, USA, 201–210. doi:10.1145/3661167.3661218

[2] Eleni Adamopoulou and Lefteris Moussiades. 2020. Chatbots: History, technology, and applications. *Machine Learning with applications* 2 (2020), 100006.

[3] Rabab Alkhalifa, Elena Kochkina, and Arkaitz Zubiaga. 2023. Building for tomorrow: Assessing the temporal persistence of text classifiers. *Information Processing Management* 60, 2 (2023), 103200. doi:10.1016/j.ipm.2022.103200

[4] Debarag Banerjee, Pooja Singh, Arjun Avadhanam, and Saksham Srivastava. 2023. Benchmarking LLM powered Chatbots: Methods and Metrics. arXiv:2308.04624 [cs.CL] https://arxiv.org/abs/2308.04624

[5] G P Shrivatsa Bhargav, Sumit Neelam, Udit Sharma, Shajith Ikbal, Dheeraj Sreedhar, Hima Karanam, Sachindra Joshi, Pankaj Dhoolia, Dinesh Garg, Kyle Croutwater, Haode Qi, Eric Wayne, and J William Murdock. 2024. An Approach to Build Zero-Shot Slot-Filling System for Industry-Grade Conversational Assistants. arXiv:2406.08848 [cs.CL] https://arxiv.org/abs/2406.08848

[6] Ning Cheng, Zhaohui Yan, Ziming Wang, Zhijie Li, Jiaming Yu, Zilong Zheng, Kewei Tu, Jinan Xu, and Wenjuan Han. 2024. Potential and limitations of llms in capturing structured semantics: A case study on srl. In *International Conference on Intelligent Computing*. Springer, 50–61.

[7] Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A Complete Survey on LLM-based AI Chatbots. arXiv:2406.16937 [cs.CL] https://arxiv.org/abs/2406.16937

[8] Anderson Gomes and Paulo Henrique M. Maia. 2023. DoME: An Architecture for Domain Model Evolution at Runtime Using NLP. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (, Campo Grande, Brazil,) *(SBES '23)*. 186–195.

[9] Google Cloud. 2025. Gemini 2.5 Flash — Generative AI on Vertex AI. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash?hl=pt-br. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash?hl=pt-br Last Update: 10-19-2025.

[10] Xufei Huang and ASCS CIS. 2021. Chatbot: design, architecture, and applications. *Univesity of Pennsylvania: School of Engineering and Applied Science, Pennsylvania* 1 (2021).

[11] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and Applications of Large Language Models. arXiv:2307.10169 [cs.CL] https://arxiv.org/abs/2307.10169

[12] Ryo Kamoi, Sarkar Snigdha Sarathi Das, Renze Lou, Jihyun Janice Ahn, Yilun Zhao, Xiaoxin Lu, Nan Zhang, Yusen Zhang, Ranran Haoran Zhang, Sujeeth Reddy Vummanthala, Salika Dave, Shaobo Qin, Arman Cohan, Wenpeng Yin, and Rui Zhang. 2024. Evaluating LLMs at Detecting Errors in LLM Responses. arXiv:2404.03602 [cs.CL] https://arxiv.org/abs/2404.03602

[13] Caihua Li, In Gim, and Lin Zhong. 2025. Confidential Prompting: Privacy-preserving LLM Inference on Cloud. arXiv:2409.19134 [cs.CR] https://arxiv.org/abs/2409.19134

[14] Sue Lim and Ralf Schmälzle. 2023. Artificial intelligence for health message generation: an empirical study using a large language model (LLM) and prompt engineering. *Frontiers in Communication* 8 (2023), 1129082.

[15] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (Jan. 2023), 35 pages. doi:10.1145/3560815

[16] Michael McTear. 2016. The Conversational Interface: Talking to Smart Devices.

[17] Alexander Tobias Neumann, Yue Yin, Sulayman Sowe, Stefan Decker, and Matthias Jarke. 2025. An LLM-Driven Chatbot in Higher Education for Databases and Information Systems. *IEEE Transactions on Education* 68, 1 (2025), 103–116. doi:10.1109/TE.2024.3467912

[18] Justice Opara-Martins, R. Sahandi, and Feng Tian. 2016. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing* 5 (04 2016). doi:10.1186/s13677-016-0054-z

[19] Lukasz Pawlik. 2025. How the Choice of LLM and Prompt Engineering Affects Chatbot Effectiveness. *Electronics* 14, 5 (2025), 888. https://www.mdpi.com/2079-9292/14/5/888

[20] Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. 2024. Efficient multi-prompt evaluation of LLMs. arXiv:2405.17202 [cs.CL] https://arxiv.org/abs/2405.17202

[21] David M. W. Powers. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv:2010.16061 [cs.LG] https://arxiv.org/abs/2010.16061

[22] V Rajaraman. 2023. From ELIZA to ChatGPT: history of human-computer conversation. *Resonance* 28, 6 (2023), 889–905.

[23] Mansi Rana, Kadri Hacioglu, Sindhuja Gopalan, and Maragathamani Boothalingam. 2025. Zero-shot Slot Filling in the Age of LLMs for Dialogue Systems. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, Steven Schockaert, Kareem Darwish, and Apoorv Agarwal (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 697–706. https://aclanthology.org/2025.coling-industry.59/

[24] Samaneh Shafee, Alysson Bessani, and Pedro M. Ferreira. 2025. Evaluation of LLM-based chatbots for OSINT-based Cyber Threat Awareness. *Expert Systems with Applications* 261 (2025), 125509. doi:10.1016/j.eswa.2024.125509

[25] Murray Shanahan. 2024. Talking about large language models. *Commun. ACM* 67, 2 (2024), 68–79.

[26] Sonali Uttam Singh and Akbar Siami Namin. 2025. A survey on chatbots and large language models: Testing and evaluation techniques. *Natural Language Processing Journal* 10 (2025), 100128. doi:10.1016/j.nlp.2025.100128

[27] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.* 45, 4 (July 2009), 427–437. doi:10.1016/j.ipm.2009.03.002

[28] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 https://arxiv.org/abs/2302.11382

[29] Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. 2025. LIFBench: Evaluating the Instruction Following Performance and Stability of Large Language Models in Long-Context Scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 16445–16468. doi:10.18653/v1/2025.acl-long.803

[30] Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794* (2023).

[31] Anis Syafiqah Mat Zailan, Noor Hasimah Ibrahim Teo, Nur Atiqah Sia Abdullah, and Mike Joy. 2023. State of the art in intent detection and slot filling for question answering system: a systematic literature review. *International Journal of Advanced Computer Science and Applications* 14, 11 (2023), 15–27.

[32] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. arXiv:2309.01219 [cs.CL] https://arxiv.org/abs/2309.01219

[33] Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. 2025. Is Your LLM Outdated? A Deep Look at Temporal Generalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 7433–7457. doi:10.18653/v1/2025.naacl-long.381