Designing LLM Agents for User-Centered Language Service Selection

Ryoichiro Ogawa, Donghui Lin, and Fumito Uwano
Graduate School of Environmental, Life, Natural Science and Technology
Okayama University
Okayama, Japan
pirj1b1u@s.okayama-u.ac.jp, {lindh, uwano}@okayama-u.ac.jp

Abstract

With the rapid expansion of language resources and services across repositories and platforms, users face an overwhelming number of options. While this diversity promises flexibility, non-experts struggle to compose appropriate resource pipelines and select services that satisfy both functional and non-functional requirements. We propose a user-centered framework of LLM agents that interprets natural-language requests and performs end-to-end language service selection. The agents extract functional requirements to form coherent task compositions and select suitable language services for each component by interpreting non-functional quality aspects embedded in contextual cues. To ensure reliable and explainable decisions, we employ a four-step structured reasoning procedure that combines Few-Shot exemplars and Chain-of-Thought reasoning: extracting functional requirements, inducing non-functional evaluation axes, applying these axes as constraints in candidate retrieval, and determining a final composition. We construct a benchmark dataset pairing diverse user requests with standardized language service profiles containing metadata and quality indicators, and evaluate our framework against representative prompting-based baselines. Results show consistent gains in Precision, Recall, and F1-score, demonstrating improved capture of both functional intent and quality preferences. These findings demonstrate that structured LLM agents can bridge natural-language user intents and language service configurations, enabling end-to-end selection and composition in a transparent and user-centered manner.

Keywords: Language Service Selection, Large Language Models, LLM Agents, Structured Reasoning, User-Centered Framework

1. Introduction

Over the past decades, the language resources community has made continuous efforts to provide not only data and tools but also *service-oriented infrastructures* that make these resources accessible through web APIs. Typical examples of such *language service infrastructures* include PANACEA (Toral et al., 2011), META-SHARE (Piperidis, 2012), DKPro Core (Eckart de Castilho and Gurevych, 2014), the LAPPS Grid (Ide et al., 2014), the Language Grid (Ishida et al., 2018; Lin et al., 2018), and the European Language Grid (Rehm et al., 2024). While these infrastructures primarily focus on enabling researchers and developers to share, compose, and deploy language resources and services, our study shifts the focus toward how non-expert end users can effectively select and utilize such services according to their own requirements. Building upon these long-standing infrastructures, recent advances in artificial intelligence—particularly in Large Language Models (LLMs)—have fundamentally transformed how language technologies are accessed, combined, and orchestrated.

Among these advances, LLMs have introduced a transformative capability to interpret and reason over natural-language instructions (Wang et al., 2024). Unlike earlier task-specific Natural Language Processing (NLP) models, LLMs demonstrate strong generalization, contextual understanding, and step-by-step reasoning abilities, allowing them to process complex, multi-step user requests in an interpretable manner (Wei et al., 2022; Zhao et al., 2023). These capabilities open up a new opportunity: performing end-to-end language service selection directly from natural-language user requests, without predefined schemas or explicit configuration inputs. This shift redefines how users—especially non-experts—can interact with language technologies, moving from manual service orchestration to intelligent, dialogue-based composition.

Meanwhile, the growing availability of language resources and services across platforms has created an unprecedented diversity of options for constructing intelligent language applications. These services exhibit remarkable performance in tasks such as machine translation, speech recognition, and text generation, and are now provided through diverse platforms, from on-device to large-scale cloud APIs (Zhao et al., 2023). This diversity greatly enhances flexibility but also increases the complexity of selecting suitable services that meet both functional and non-functional user requirements.

Specifically, users, especially non-experts, are

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 599–608
11–16 May 2026.

now expected to navigate a vast landscape of heterogeneous language services that differ in capabilities, interfaces, and trade-offs. In practice, they must not only identify functional components (e.g., translation, summarization, or speech recognition) but also consider non-functional requirements such as latency, privacy, and reliability. These aspects are often implicit in natural-language user requests, making language service selection an inherently ambiguous and cognitively demanding task.

Previous research on service selection has explored quantitative optimization frameworks (Zeng et al., 2004; Hayyolalam and Kazem, 2018) and adaptive learning-based methods (Yang et al., 2024). Yet, they often assume explicitly defined, numerical requirements, making them inadequate for interpreting qualitative or context-dependent user intents expressed in natural language. For instance, a request such as "read this confidential report image, translate it, and create a summary" implicitly involves multiple steps (OCR, translation, summarization) and a privacy constraint from the word "confidential," which traditional methods fail to capture.

To address these challenges, we propose a user-centered framework of LLM agents that interprets natural-language requests and autonomously performs end-to-end language service selection. The core idea is to treat service selection not as a single-step retrieval problem but as a structured reasoning process that bridges the semantic gap between user intentions and service configurations. To this end, we introduce a four-step structured reasoning framework that decomposes the task into: (1) analyzing user intent, (2) extracting and organizing functional requirements, (3) interpreting context-dependent non-functional quality aspects, and (4) selecting appropriate language services for each component. The LLM agent is guided through these steps via Few-Shot prompting (Brown et al., 2020) and Chain-of-Thought (CoT) reasoning (Wei et al., 2022), enabling it to infer both explicit and implicit requirements while maintaining transparency and consistency in its decision process.

Our main contributions are threefold. First, we propose a user-centered framework of LLM agents that interprets natural-language requests and autonomously performs end-to-end language service selection, bridging user intents and service configurations. Second, we introduce a four-step structured reasoning procedure that integrates Few-Shot prompting and Chain-of-Thought reasoning to enable explainable and robust decision-making for functional and non-functional requirements. Finally, we construct a benchmark dataset pairing diverse user requests with standardized language service profiles containing metadata and quality indicators, and empirically demonstrate that our approach consistently outperforms prompting-based baselines in Precision, Recall, and F1-score.

This paper is structured as follows. Section 2 reviews related work. Section 3 details the proposed methodology. Section 4 presents evaluation results, followed by discussion in Section 5. Section 6 concludes the paper.

2. Related Work

Previous research relevant to our study falls into two areas: service selection methodologies and recent advances in LLMs for reasoning and decision-making.

Traditional approaches to service selection have been widely studied in the fields of service-oriented computing and cloud systems. Representative works include Quality-of-Service (QoS)-aware frameworks that rank candidate services based on numerical metrics such as latency, availability, or cost (Zeng et al., 2004; Hayyolalam and Kazem, 2018). Later extensions employed heuristic optimization and reinforcement learning to handle dynamic environments (Yang et al., 2024). While these methods provide formalized selection models, they rely on the assumption that user requirements are explicit and numerically quantifiable, which limits their applicability to qualitative, context-dependent tasks such as language service selection. Recent reviews (Zeng et al., 2025) have also highlighted persistent challenges in capturing user-centered constraints under dynamic or incomplete information.

In the language resources community, a number of initiatives have developed infrastructures to standardize the sharing and composition of linguistic data and tools—such as META-SHARE (Piperidis, 2012), the LAPPS Grid (Ide et al., 2014), the Language Grid (Ishida et al., 2018), and the European Language Grid (Rehm et al., 2024). These platforms have established metadata schemas and APIs for accessing and combining language services, forming the foundation for interoperable language technology ecosystems. However, they still rely on pre-defined workflows or manually annotated quality information, and are primarily targeted at developers or researchers. As a result, they offer limited support for end users who wish to select or compose services dynamically based on natural-language specifications or evolving requirements.

Beyond these infrastructures, recent progress in artificial intelligence, particularly in LLMs, has opened new opportunities for addressing this gap. LLMs exhibit strong generalization and reasoning abilities across diverse tasks by interpreting natural-language instructions (Wang et al., 2024). Prompting techniques such as Few-Shot learning (Brown et al., 2020) and Chain-of-Thought reasoning (Wei

et al., 2022) have further enhanced their ability to reason transparently. Nevertheless, unstructured prompting often leads to inconsistent reasoning when multiple quality aspects (e.g., accuracy, latency, and privacy) must be balanced simultaneously. Recent agent frameworks such as AutoGPT, LangChain, and AutoGen (Significant Gravitas, 2023; LangChain AI, 2022; Wu et al., 2023) have begun to explore modular reasoning with planning and memory, but these architectures remain loosely guided.

In parallel, tool-augmented LLM research has addressed the selection and invocation of external tools from large repositories. ToolLLM (Qin et al., 2024) fine-tunes models on a large-scale tool-use dataset to enable open-domain API selection, while Gorilla (Patil et al., 2023) trains LLMs to generate accurate API calls grounded in documentation. These methods primarily focus on functional tool matching through model fine-tuning, whereas our framework operates without fine-tuning and additionally incorporates non-functional quality reasoning to support user-centered service selection in the language resource domain. Rather than relying on model fine-tuning, our study introduces a structured reasoning framework that systematically decomposes the language service selection process, enabling explainable, user-centered decision-making directly from natural-language requests.

3. Design of the Language Service Selection Agent

This section describes the architecture and core reasoning mechanism of our language service selection agent. We first introduce the overall design concept and modular structure, including the use of a knowledge base composed of Few-Shot examples and language service quality information. We then detail the structured reasoning process that guides the agent's decision-making over natural-language requests.

3.1. Overview of the Design Concept

User-centered language service selection poses a fundamental challenge: interpreting user requests written in natural language, which often blend both explicit functional requirements and implicit non-functional quality preferences. Unlike traditional approaches that rely on structured queries or numeric utility functions, real-world user requests typically lack clearly specified constraints. This makes it difficult for conventional systems to identify user intent and optimize service choices accordingly.

To address this challenge, we propose an LLM agent that performs structured reasoning for end-to-end language service selection. As illustrated in Figure 1, this agent is powered by a Large Language Model (LLM) and guided by a structured prompt architecture. To support the agent's decision-making process, we provide a knowledge base composed of two key elements: a set of Few-Shot examples that encode reusable reasoning policies, and a Service Quality Information Table that offers factual grounding for candidate evaluation. The Few-Shot examples demonstrate reusable reasoning policies that the agent generalizes to unseen inputs. The Service Quality Information Table is constructed using LLM-based analysis of various public service-related sources, such as official specifications, technical reports, and academic papers. Details of this construction methodology are presented in Section 4.2.

In this architecture, the selection agent interprets the request, consults the few-shot policy, and queries the Service Quality Information Table to retrieve service-specific metadata. Based on this knowledge base, the agent selects appropriate language services by executing the structured reasoning process, which will be elaborated in Section 3.2.

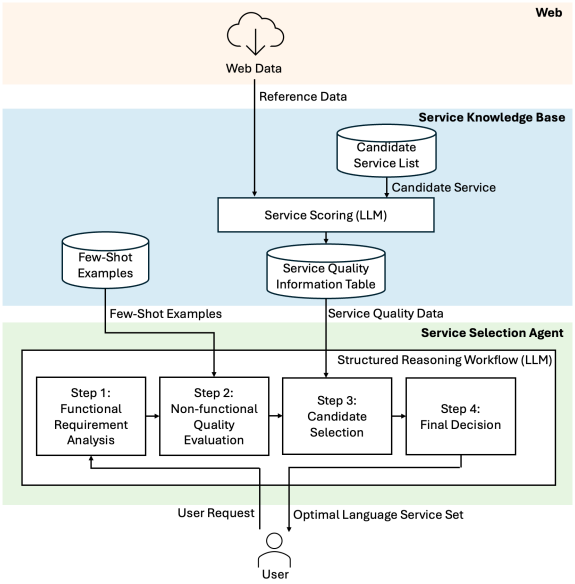

Figure 1: Architecture of the proposed LLM agent with structured reasoning

For example, consider the request: "Summarize in Japanese the causes of customer dissatisfaction from the meeting audio with our overseas client." The agent must recognize that this involves three functional sub-tasks: (1) transcribing the audio (speech recognition), (2) translating the result (machine translation), and (3) summarizing the translated content (text summarization). In parallel, it must also infer contextual non-functional quality requirements—most notably, the need for privacy due to the phrase "our overseas client." This illustrates the dual need for functional interpretation

and context-sensitive quality assessment, which our system is explicitly designed to support in language service selection.

3.2. LLM Agent with Structured Reasoning

To ensure consistent and interpretable decision-making, our language service selection agent employs a structured reasoning framework executed entirely by the LLM at runtime. The agent's behavior is guided by a carefully designed prompt format, as illustrated in Figure 2, which decomposes the task into four fixed reasoning steps. This approach goes beyond traditional in-context learning by embedding not just input-output examples but complete chains of reasoning derived from Few-Shot examples. The prompt begins with Few-Shot examples that define the agent's policy by demonstrating how to extract both functional requirements and implicit non-functional quality preferences from natural-language inputs. Unlike examples that simply provide final answers, each instance in our setup illustrates the full reasoning trajectory across the four steps. This ensures that the agent not only understands the task but also follows a consistent decision-making logic across diverse inputs. The Service Quality Information Table is included to ground the agent's evaluation in real-world language service metadata, minimizing hallucinations and ensuring factual reliability.

The reasoning process consists of the following steps:

- **Step 1: Functional Requirement Analysis**

  The agent identifies and structures the functional requirements explicitly described in the user request and infers any implied non-functional quality preferences. For instance, in the request "summarize this confidential document and translate into English," the functional requirements are "summarization" and "translation."

- **Step 2: Non-functional Quality Evaluation**

  The agent ranks language service quality dimensions (e.g., accuracy, latency, privacy) based on contextual relevance. Instead of assigning explicit weights, it produces a qualitative prioritization (e.g., privacy > responsiveness > accuracy) by mapping linguistic cues to preferences using the patterns encoded in the Few-Shot examples. This step defines *what matters* in evaluating language services for the given user context.

- **Step 3: Candidate Selection**

  The agent filters language services that meet the functional requirements and compares their non-functional quality characteristics using the Service Quality Information Table. For each non-functional quality dimension identified in Step 2, the agent ranks candidate language services and generates intermediate scores, resulting in a shortlist of top candidates. This step evaluates *how each candidate performs* with respect to user preferences.

- **Step 4: Final Decision**

  The agent synthesizes the prior reasoning steps to select the most suitable language service, optionally providing a natural-language justification. This justification supports transparency, helps identify reasoning errors, and fosters user trust in the final decision.

```
# Few-Shot examples (Language service selection
 examples)
# Each example demonstrates a policy for
mapping linguistic and contextual cues in user
requests to quality evaluation of candidate
language services. Apply this learned policy to
 the inference step.
{shot_info}
---
# Service Quality Information (selected from {
service_info})
{service_info}
---
# User request
# Apply the learned policy to this user request.

{user_request}
---
## Step 1: Functional Requirement Analysis
# Analyze user requirements, identify and
structure functional requirements
...
## Step 2: Non-functional Quality Evaluation
# Based on the non-functional quality
requirements identified with reference to the
Few-Shot case study, determine the non-
functional quality requirements that should be
most important in this task
...
## Step 3: Candidate Selection
# Refer to the non-functional quality
information, be sure to adhere to the
requirements set in Step 2, and narrow down the
 optimal service
...
## Step 4: Final Decision
# Judge the investigation results of Step 3,
and select the final service selection
...
```

Figure 2: Prompt template for structured reasoning

This structured reasoning process draws inspiration from Chain-of-Thought (CoT) prompting (Wei et al., 2022), which enhances logical consistency by requiring intermediate reasoning steps. Prior research has shown that unstructured prompts often lead to hallucination, attentional drift, and inconsistent prioritization (Wang et al., 2024). By contrast, our design enforces alignment between decision rationale and observed language service metadata,

enabling robust and interpretable reasoning without model fine-tuning. Unlike standard CoT reasoning, our structured format explicitly separates reasoning control from factual grounding, ensuring consistency across diverse language service contexts and service domains.

In summary, the structured reasoning framework serves as both an execution plan and an alignment mechanism, enabling the agent to generalize across diverse requests while maintaining transparency. In the next section, we empirically evaluate the agent's effectiveness using real-world language service data and request scenarios.

4. Evaluation

This section evaluates the effectiveness, robustness, and generalizability of the proposed language service selection framework. We first assess its accuracy compared to baseline prompting methods. We then examine its sensitivity to the number of Few-Shot examples (K) and its adaptability across different LLMs. All experimental resources developed for this study—including the benchmark dataset, prompt designs, language service profiles, and evaluation results—are available at: https://github.com/Ogaaawa/LLM_Agents_for_User-Centered_Language_Service_Selection.

4.1. Evaluation Metrics

We treat language service selection as a set-based prediction task, where the goal is to select the correct set of language services for a given user request. We use standard metrics, Precision, Recall, and F1-score, defined as follows:

$$\text{Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{|sc_i \cap cc_i|}{|sc_i|} \tag{1}$$

$$\text{Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{|sc_i \cap cc_i|}{|cc_i|} \tag{2}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Here, sc_i is the predicted set of language services for request i, and cc_i is the corresponding ground-truth set.

4.2. Experimental Setup

4.2.1. User Requests

We manually constructed 60 user request sentences: 30 clearly stated (specific) and 30 ambiguous (abstract) requests. These cover realistic use cases (e.g., daily support, workplace productivity), each including at least one explicit non-functional requirement.

To clarify this distinction, consider the following examples. A **specific** request might be: "Quickly perform character recognition on this sign written in English and translate the text into Japanese." This clearly defines two functional requirements—text recognition and translation—and implies a need for low latency. In contrast, an **abstract** request could be: "Quickly make this English sign understandable to Japanese people." This type of request is more ambiguous and requires the agent to infer the necessary sub-tasks.

4.2.2. Candidate Services and Profiling

We created a reproducible evaluation testbed consisting of 30 candidate language services, including services across six categories, selected to address generalized user requirements by encompassing both linguistic and visual modalities: Speech Recognition, Machine Translation, Text Generation, Optical Character Recognition (OCR), Image Recognition, and Action Recognition. For each language service, we collected publicly available reference materials, such as official documentation, technical whitepapers, and academic articles. These documents were analyzed by an LLM-based service scoring module (Figure 1), which extracted scores across three language service quality dimensions: Responsiveness, Privacy Protection, and Accuracy. The scoring criteria for each dimension were defined as follows:

- **Responsiveness:** Scores were assigned based on latency thresholds grounded in human-computer interaction (HCI) literature, such as Nielsen's widely accepted perceptual boundaries (e.g., 0.1s for instantaneous response, 1s for uninterrupted interaction, and 10s for task-level feedback).
- **Privacy Protection:** A five-level scale was developed based on data protection principles such as data minimization and purpose limitation, as defined in regulations like the GDPR (General Data Protection Regulation). Language services performing all computation on the user's device (i.e., on-device inference) received the highest score.
- **Accuracy:** Where available, scores were based on standardized benchmark metrics (e.g., BLEU for machine translation, or WER for speech recognition) reported in third-party evaluations.

This process yielded the Service Quality Information Table (Figure 3), a grounded knowledge base for the language service selection agent.

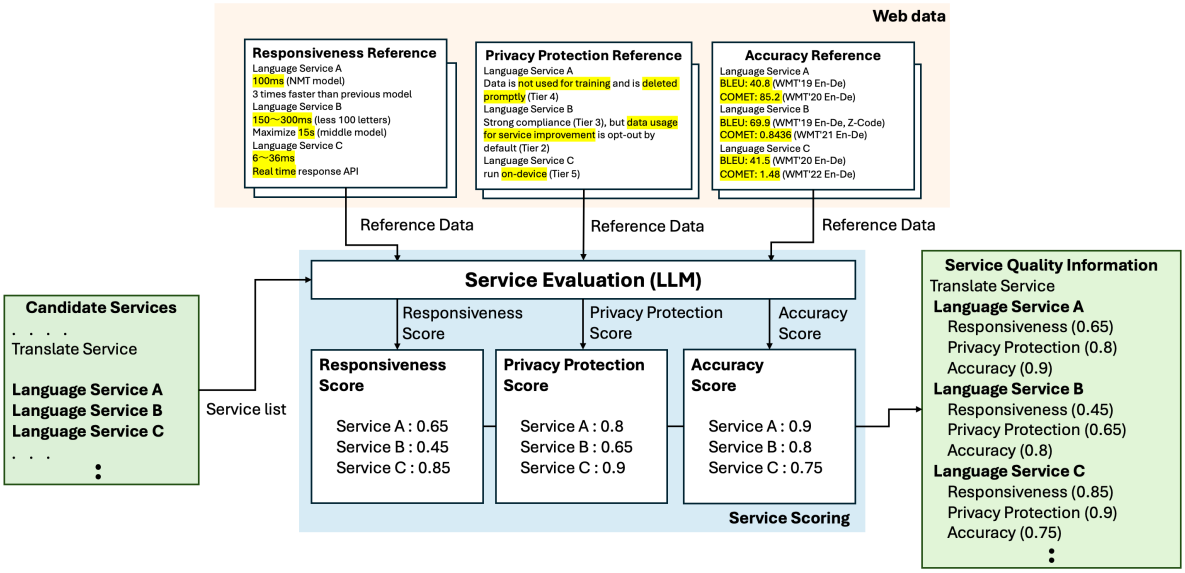

Figure 3: Generation of the service quality information table

Table 1: Performance comparison across prompting strategies for language service selection

	Specific Requests			Abstract Requests			Overall		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Zero-Shot Prompting	0.275	0.286	0.279	0.106	0.128	0.114	0.190	0.207	0.197
Few-Shot Prompting	0.511	0.511	0.510	0.222	0.239	0.227	0.367	0.375	0.368
Unstructured CoT	0.572	0.578	0.574	0.339	0.344	0.338	0.456	0.461	0.456
Proposed Method	0.665	0.668	0.666	0.449	0.415	0.424	0.557	0.542	0.545

4.2.3. Ground-Truth Annotation

Ground-truth labels were created using the Simple Additive Weighting (SAW) method (Hwang and Yoon, 2012), a standard Multi-Criteria Decision-Making (MCDM) approach. Experts assigned non-functional quality weights and computed service scores; the top-scoring language service candidate was selected as ground truth. This setup links traditional MCDM with LLM-based reasoning.

4.3. Baseline Methods

To evaluate the effectiveness of our proposed framework, we compare it against three prompting baselines commonly used in LLM-based decision-making:

- **Zero-Shot Prompting:** The LLM agent receives only the user request and the *Service Quality Information Table* without any examples or reasoning steps. This serves as a naive baseline and represents the model's performance without guidance.

- **Few-Shot Prompting:** The LLM agent is provided with a small number of example request-response pairs to guide inference, but no explicit reasoning process is enforced. This setting tests the benefit of in-context learning alone.

- **Unstructured Chain-of-Thought (CoT):** In this setting, the prompt includes examples with step-by-step reasoning, but the reasoning structure is unconstrained. The agent is instructed to "think step-by-step," but is not bound to our four-step framework.

By comparing them, we can assess the impact of structured reasoning and the use of Few-Shot examples in improving selection accuracy.

4.4. Experimental Results

4.4.1. Effectiveness of the Proposed Method

We used Gemini 1.5 Pro as the default LLM backend for our primary evaluations (Tables 1 and 2) due to its superior reasoning capabilities. To ensure result stability, each score reported in this section represents the average of 10 trials with different random seeds. We evaluated the effectiveness of our method by comparing it against baseline prompting strategies (Table 1). The task is treated as a set-based prediction problem, with performance measured using Precision, Recall, and F1-score.

Our method, which integrates a structured Chain-of-Thought (CoT) framework with ten Few-Shot ex-

Table 2: Effect of Few-Shot example count (K) on structured reasoning performance

	Specific Requests			Abstract Requests			Overall		
No. of Examples (K)	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
K=0	0.550	0.572	0.557	0.300	0.261	0.274	0.425	0.417	0.415
K=1	0.469	0.469	0.469	0.328	0.311	0.307	0.399	0.390	0.388
K=5	0.572	0.578	0.574	0.533	0.450	0.470	0.553	0.514	0.522
K=10	0.665	0.668	0.666	0.449	0.415	0.424	0.557	0.542	0.545

Table 3: Generalizability of structured prompting across different LLM backends

	Specific Requests			Abstract Requests			Overall		
Model	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
GPT-4o	0.639	0.661	0.648	0.397	0.389	0.378	0.518	0.525	0.513
GPT-4o-mini	0.356	0.344	0.349	0.411	0.232	0.286	0.383	0.288	0.317
Gemini 1.5 Pro	0.665	0.668	0.666	0.449	0.415	0.424	0.557	0.542	0.545
Gemini 1.5 Flash	0.394	0.400	0.396	0.375	0.256	0.294	0.385	0.328	0.345

amples, achieves the highest overall F1-score of 0.545, demonstrating its superiority over simpler prompting techniques. A more detailed analysis reveals strong performance on both Specific and Abstract requests, with F1-scores of 0.666 and 0.424, respectively. The higher score on Specific requests indicates that the structured reasoning process works particularly well when user intent is clearly expressed. Nevertheless, the method also outperforms all baselines on Abstract requests, showing its ability to infer implicit constraints from ambiguous input. The lower score in this case reflects the inherent challenge of interpreting underspecified instructions, suggesting that additional mechanisms, such as clarification prompts or iterative reasoning, may be required for further improvement.

Baseline results further support our design choices. Zero-shot prompting yields poor performance (F1 = 0.197), confirming the importance of in-context examples. Performance improves with standard Few-Shot Prompting (F1 = 0.368), and is further enhanced by adding unstructured CoT reasoning (F1 = 0.456), indicating the benefits of guided reasoning steps. Our method provides the most substantial improvement. This supports our hypothesis that enforcing a structured four-step reasoning process—functional requirement analysis, non-functional quality evaluation, candidate selection, and final decision—enables robust and interpretable decision-making. By constraining the reasoning trajectory, the method mitigates common issues such as hallucination and inconsistency (Wei et al., 2022). Furthermore, to confirm whether the set of 60 user requests is sufficient for statistical analysis, we conducted a paired t-test comparing the F1-scores with and without the structured reasoning method. The results yielded p-values below the 5% significance level ($p < 0.05$) for both specific and abstract requests, demonstrating a statistically significant difference in service selection performance.

4.4.2. Robustness to the Number of Few-Shot Examples

Next, we examine the robustness of our method by analyzing how the number of Few-Shot examples (K) affects its performance. Providing a well-chosen set of examples is important, but the optimal number may vary. As in Section 4.4.1, each score represents the average of 10 trials with different random seeds.

Table 2 shows the performance of our method with K set to 0, 1, 5, and 10. The results indicate that performance generally improves as the number of examples increases, peaking at $K = 10$. This aligns with findings from prior work, where a greater number of examples facilitates more robust generalization (Brown et al., 2020). Interestingly, performance drops at $K = 1$ (F1 = 0.388), even lower than the score at $K = 0$ (F1 = 0.415). This suggests that a single unrepresentative example may bias the model more negatively than providing no example at all. This is consistent with previous studies on the instability of in-context learning under limited examples (Brown et al., 2020).

Overall, this analysis confirms that a sufficient number of examples is critical for the robustness of our method. The strong performance at $K = 10$ supports our choice for the final configuration, as it provides the agent with diverse patterns to learn a stable selection policy.

4.4.3. Analysis of Performance Consistency

We assess performance consistency by analyzing the variation across ten repeated runs for each request type, using Gemini 1.5 Pro with the number of Few-Shot examples set to $K = 10$. Figure 4 presents box plots of Precision, Recall, and F1-score for Specific and Abstract requests.

The agent shows consistent behavior on Specific requests, with a median F1-score of 0.668 and tightly grouped results (top boxplot), indicating that the structured reasoning process is robust

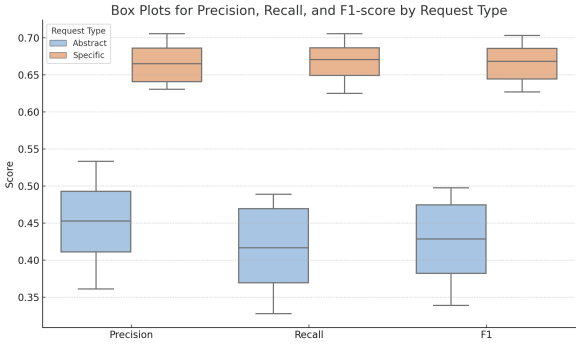

Figure 4: Box plots for evaluation results by user request type

when user requests are clear and well-defined. In contrast, Abstract requests yield more dispersed results (bottom boxplot), revealing a substantial variability in performance. This inconsistency suggests that the agent's behavior is more sensitive to prompt execution when faced with vague or underspecified instructions. A likely cause is that abstract inputs lack clear linguistic cues for inferring implicit constraints, making it harder for the model to align its reasoning path. This points to a limitation of the current Few-Shot prompting strategy under ambiguous conditions.

Improving performance consistency for such inputs may require additional clarification mechanisms, such as interactive refinement or explicit constraint elicitation, or more diverse and context-aware Few-Shot examples to better capture the variability in user intent.

4.4.4. Generalizability Across Different LLMs

Finally, we evaluate the generalizability of our framework across different LLMs. This experiment tests whether our prompting strategy is robust to changes in the underlying model architecture.

As shown in Table 3, where the number of Few-Shot examples is set to $K = 10$, the overall performance varies with model capability. The strongest models (e.g., Gemini 1.5 Pro and GPT-4o) achieve the highest F1-scores of 0.545 and 0.513, respectively. Smaller and faster variants (e.g., GPT-4o-mini, Gemini 1.5 Flash) show decreased performance, reflecting the common trade-off between capability and efficiency.

Our framework improves performance across all models tested. While absolute performance differs, the structured reasoning consistently outperforms simpler prompting methods across architectures. This indicates that our approach is generalizable and model-agnostic, and that its effectiveness scales with the reasoning power of the LLM.

5. Discussion

The evaluation in Section 4 demonstrates the effectiveness of our structured reasoning framework but also reveals several limitations. Not all language service selections were correct, and some characteristic error patterns appeared, especially for abstract or underspecified user requests. Moreover, the construction of the Service Quality Information Table relies on LLM-based profiling and heuristic rules, which may limit practical reliability.

5.1. Limitations of Structured Reasoning

To conduct a detailed error analysis, we extracted and manually reviewed the outputs from a single representative run (60 requests) out of the 10 evaluation trials. This review identified three major error types. The following percentages are computed over the total number of test instances in this selected run, not solely over incorrect predictions. The first, **Non-functional Quality Misinterpretation (12%)**, occurred when the agent linked contextual expressions to inappropriate quality dimensions. For instance, the phrase "in detail" should indicate *Accuracy* but was misclassified as *Privacy*, a pattern frequent in abstract requests lacking clear cues. The second, **Reasoning Drift (15%)**, appeared when the correct primary quality aspect was initially recognized but later lost—e.g., shifting focus from *Accuracy* to unrelated criteria such as *Responsiveness*. This suggests that structured prompts alone cannot fully stabilize multi-step reasoning without stronger anchoring from Few-Shot exemplars. Finally, **Incomplete Step Execution (6%)** occurred when early reasoning was not consistently applied in the final decision, likely due to long-context limitations known in LLM reasoning (Liu et al., 2024).

Overall, while structured reasoning reduces procedural inconsistency, future designs should strengthen the mapping from linguistic cues to non-functional qualities and enforce coherence across reasoning steps.

5.2. Limitations of Language Service Quality Profiling

The Service Quality Information Table was generated through LLM-based analysis of public documentation, which may introduce structural biases—for example, favoring on-device services in *Responsiveness* and *Privacy Protection*. Although this approach improves reproducibility, true evaluation of *Responsiveness* and *Accuracy* should rely on empirical latency and benchmark metrics (e.g., BLEU, WER). Moreover, the weighting among quality criteria remains context dependent. Future work will explore hybrid profiling that combines empirical

measurements, expert annotation, and automated document analysis to enhance transparency and reliability of language service metadata.

Despite these limitations, our study confirms the potential of structured reasoning as a foundation for explainable LLM-based language service selection. By explicitly modeling both functional and quality dimensions from natural-language requests, our framework provides a reproducible path toward interpretable agentic systems in real-world language technology environments.

6. Conclusion

This paper presented a user-centered framework of LLM agents for language service selection, addressing the challenge of interpreting natural-language requests and making multi-criteria decisions across heterogeneous language services. The proposed method integrates a four-step structured reasoning process with Few-Shot examples, aligning LLM outputs with both functional goals and non-functional quality preferences. This design enables interpretable and consistent decision-making in end-to-end language service composition. Experiments on a testbed of 60 diverse requests and 30 language services demonstrated that our approach outperforms baseline prompting methods, achieving an overall F1-score of 0.545 and showing robustness across Few-Shot configurations and LLM architectures. These findings validate that structured reasoning helps bridge natural-language user intent and quality-aware language service configuration.

While promising, the study revealed limitations such as occasional reasoning drift and sensitivity to vague expressions. Moreover, the current LLM-based QoS profiling method requires further refinement before real-world deployment. Future work includes enhancing reasoning stability under ambiguity and integrating real-time QoS monitoring. Additionally, we plan to expand our benchmark dataset to ensure broader generalization to novel services. Finally, we aim to extend our framework to complex automated workflows by evaluating the optimal execution order of multiple services in dynamic environments without a fixed gold standard.

Acknowledgements

This research was supported by a Grant-in-Aid for Scientific Research (B) (25K03227, 2025-2028; 24K03001, 2024-2026) from the Japan Society for the Promotion of Science (JSPS).

7. Bibliographical References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Reykjavik, Iceland. European Language Resources Association (ELRA).

Vahideh Hayyolalam and Ali Asghar Pourhaji Kazem. 2018. A systematic literature review on qos-aware service composition and selection in cloud environment. *Journal of Network and Computer Applications*, 110:52–74.

Ching-Lai Hwang and Kwangsun Yoon. 2012. *Multiple attribute decision making: methods and applications a state-of-the-art survey*. Springer Science & Business Media.

Nancy Ide, James Pustejovsky, Christopher Cieri, Eric Nyberg, Di Wang, Keith Suderman, Marc Verhagen, and Jonathan Wright. 2014. The language application grid. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 22–30, Reykjavik, Iceland. European Language Resources Association (ELRA).

Toru Ishida, Yohei Murakami, Donghui Lin, Takao Nakaguchi, and Masayuki Otani. 2018. Language service infrastructure on the web: The language grid. *Computer*, 51(6):72–81.

LangChain AI. 2022. Langchain. https://github.com/langchain-ai/langchain. Accessed: 2025-10-24.

Donghui Lin, Yohei Murakami, and Toru Ishida. 2018. A framework for multi-language service design with the language grid. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Stelios Piperidis. 2012. The meta-share language resources sharing infrastructure: Principles, challenges, solutions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 36–42, Istanbul, Turkey. European Language Resources Association (ELRA).

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024)*.

Georg Rehm, Stelios Piperidis, Dimitris Galanis, Penny Labropoulou, Maria Giagkou, Miltos Deligiannis, Leon Voukoutis, Martin Courtois, Julian Moreno-Schneider, and Katrin Marheinecke. 2024. European language grid: One year after. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6353–6362, Torino, Italy. ELRA and ICCL.

Significant Gravitas. 2023. Autogpt: Build, deploy, and run ai agents. https://github.com/Significant-Gravitas/AutoGPT. Accessed: 2025-10-24.

Antonio Toral, Pavel Pecina, Andy Way, and Marta Poch. 2011. Towards a user-friendly webservice architecture for statistical machine translation in the panacea project. In *Proceedings of the 15th Conference of the European Association for Machine Translation (EAMT'11)*, pages 63–70, Leuven, Belgium.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Zheming Yang, Yuanhao Yang, Chang Zhao, Qi Guo, Wenkai He, and Wen Ji. 2024. Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services. *arXiv preprint arXiv:2405.14636*.

Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Anne H. H. Ngu. 2025. Qos-aware service composition: A retrospective. *IEEE Transactions on Software Engineering*, 51(03):836–841.

Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. 2004. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.