# HuggingR$^4$: A Progressive Reasoning Framework for Discovering Optimal Model Companions

Anonymous ACL submission


## Abstract

Building effective LLM agents increasingly requires selecting appropriate AI models as tools from large open repositories (*e.g.*, HuggingFace with >2M models) based on natural language requests. Unlike invoking a fixed set of API tools, repository-scale model selection must handle massive, evolving candidates with incomplete metadata. Existing approaches incorporate full model descriptions into prompts, resulting in prompt bloat, excessive token costs, and limited scalability. To address these issues, we propose **HuggingR$^4$**, the first framework to recast model selection as an iterative reasoning process rather than one-shot retrieval. By integrating **R**easoning, **R**etrieval, **R**efinement, and **R**eflection, HuggingR$^4$ progressively decomposes user intent, retrieves candidates through multi-round deliberation, refines selections via fine-grained analysis, and validates results through reflection. To facilitate rigorous evaluation, we introduce a large-scale benchmark comprising 14,399 user requests across 37 task categories. Experiments demonstrate that HuggingR$^4$ achieves 93.01% workability and 84.25% reasonability—outperforming state-of-the-art baselines by absolute margins of 17.81% and 23.13%, respectively, while reducing token consumption by 6.9×.


## 1 Introduction

The Model Context Protocol (MCP) has catalyzed the evolution of large language models by making the integration of diverse external interfaces a key paradigm in AI agent development, with remarkable results across various fields (Zhang et al., 2025; Contreras Ortiz et al., 2025; Kadiyala et al., 2024). These external interfaces generally fall into two categories: functional APIs (Li et al., 2023a; Mastouri et al., 2025; Liu et al., 2025) and system-level interfaces (Ehtesham et al., 2025; Ahmadi et al., 2025). By leveraging these interfaces, LLMs extend beyond pure text-based reasoning, enabling them to process visual inputs, access real-time data, and perform complex tasks. As the interface ecosystem rapidly expands across cross-modal domains, selecting the most appropriate interfaces from vast repositories based on user queries has become a critical challenge in building AI agents.

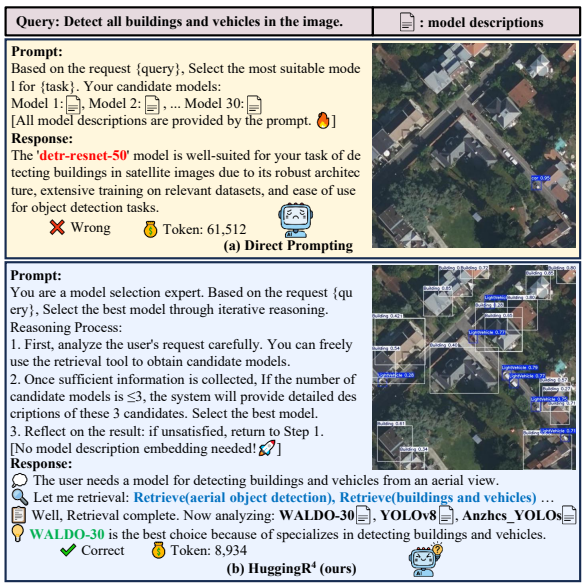


Figure 1: Comparison of baseline and HuggingR$^4$ prompts and responses. For the same query, HuggingR$^4$ chooses the correct model with 6.9× fewer tokens, whereas the baseline chooses an incorrect model.

Among these interfaces, community-driven AI models represent a rich yet underexplored resource. While existing work has primarily focused on invoking a fixed set of curated APIs (Gan and Sun, 2025; Lumer et al., 2025), the selection of appropriate models from large, evolving community repositories remains largely unexplored. To date, the HuggingFace community has amassed over 2.5 million models and 776k datasets, supporting 4,700 languages and vastly expanding the available tool set for AI agents. Models fine-tuned on domain-specific data inherently excel in meeting users' particular needs. For instance, BioBERT (Lee et al., 2020) is significantly better suited for medical text

analysis than standard BERT (Koroteev, 2021). However, due to the sheer scale of the community model ecosystem and pervasive metadata gaps, the performance of traditional retrieval systems is severely degraded. Therefore, developing model selection algorithms that balance both efficiency and accuracy has become a critical bottleneck in unlocking the full potential of community models.

Current mainstream approaches for community model selection, such as HuggingGPT (Shen et al., 2023), rely on directly embedding all candidate model descriptions into prompts, leading to significant token consumption (Figure 1). Recent research has shifted towards a paradigm driven by meta-learners that select models based on dataset similarity (Li et al., 2023b; Zhou et al., 2024; Lu et al., 2023; Zhang et al., 2024b). However, these methods are limited to fixed, small-scale, task-specific repositories and struggle to generalize to novel tasks or adapt to dynamically evolving large-scale model communities. Moreover, existing approaches lack the capability to perform fine-grained analysis of model descriptions. This makes it difficult to achieve an optimal match between the heterogeneous nature of community models and the diverse, complex needs of users.

To address these challenges, we introduce HuggingR$^4$, a novel reasoning-driven framework for scalable model selection in dynamic community repositories. It employs a coarse-to-fine strategy that combines efficient broad-scale retrieval with precise fine-grained refinement, achieving state-of-the-art performance while reducing token consumption by 85.6% (Figure 1). The framework of HuggingR$^4$ operates in three stages: **(1) Iterative Reasoning and Dual-Stream Retrieval.** The LLM performs step-by-step reasoning to decompose user queries and iteratively retrieves a compact set of candidates. **(2) Fine-Grained Semantic Refinement.** The LLM conducts fine-grained analysis of candidates' full descriptions to select the most suitable model based on task-specific requirements. **(3) Meta-Cognitive Reflection.** The LLM validates selections through self-reflection and dynamically adjusts retrieval parameters if necessary. To handle pervasive metadata gaps, we integrate a failure traceback mechanism after each retrieval operation. Additionally, we introduce a novel sliding window strategy that intelligently manages context window usage, optimizing both information coverage and token efficiency.

To evaluate our approach, we introduce **ModelSelect-Bench**, the first large-scale benchmark for repository-scale model selection. It comprises 1,016 single-task and 13,383 multi-task requests spanning vision, language, audio, and multimodal scenarios, each rigorously human-annotated with optimal model selections. We employ two complementary metrics: *workability* (whether the selected model executes successfully), and *reasonability* (whether outputs meet user requirements). In summary, our contributions are:

- We propose **HuggingR$^4$**, the first scalable framework for model selection in community repositories, combining iterative LLM reasoning with advanced retrieval techniques to scale seamlessly to large, evolving repositories.
- We construct **ModelSelect-Bench**, the first comprehensive global benchmark comprising 14K+ requests across vision, language, audio, and multimodal domains.
- Extensive experiments demonstrate that HuggingR$^4$ achieves SOTA performance across multiple LLM architectures while reducing token consumption by $6.9\times$.

## 2 Related Work

**Model Selection.** Model selection aims to identify the most suitable model from a candidate pool for specific tasks. Existing approaches primarily fall into data-driven and query-driven paradigms.

Data-driven methods require access to datasets and extract meta-features for model recommendation. Early work relied on rule-based heuristics and lightweight selector models (Taylor et al., 2018; Marco et al., 2020; Wang et al., 2017; Meng et al., 2021; Hu et al., 2023). Recent advances focus on metadata-based (Li et al., 2021; Luo et al., 2020; Lu et al., 2023; Li et al., 2023b; Zhou et al., 2024; Wanaskar et al., 2025; Zhang et al., 2024b) and graph-based (Li et al., 2024; Wang et al., 2023b) approaches. However, they are infeasible in community scenarios due to missing metadata, the inability to access datasets in real-time, and neglect of user preferences.

Query-driven methods eliminate dataset dependency via natural language queries. While HuggingGPT (Shen et al., 2023) pioneered this by embedding descriptions directly into prompts, this one-shot paradigm scales poorly to massive repositories and struggles to capture fine-grained task requirements. Broadly, existing approaches remain ill-equipped for community repositories characterized by rapid evolution, incomplete metadata, and diverse multimodal tasks.

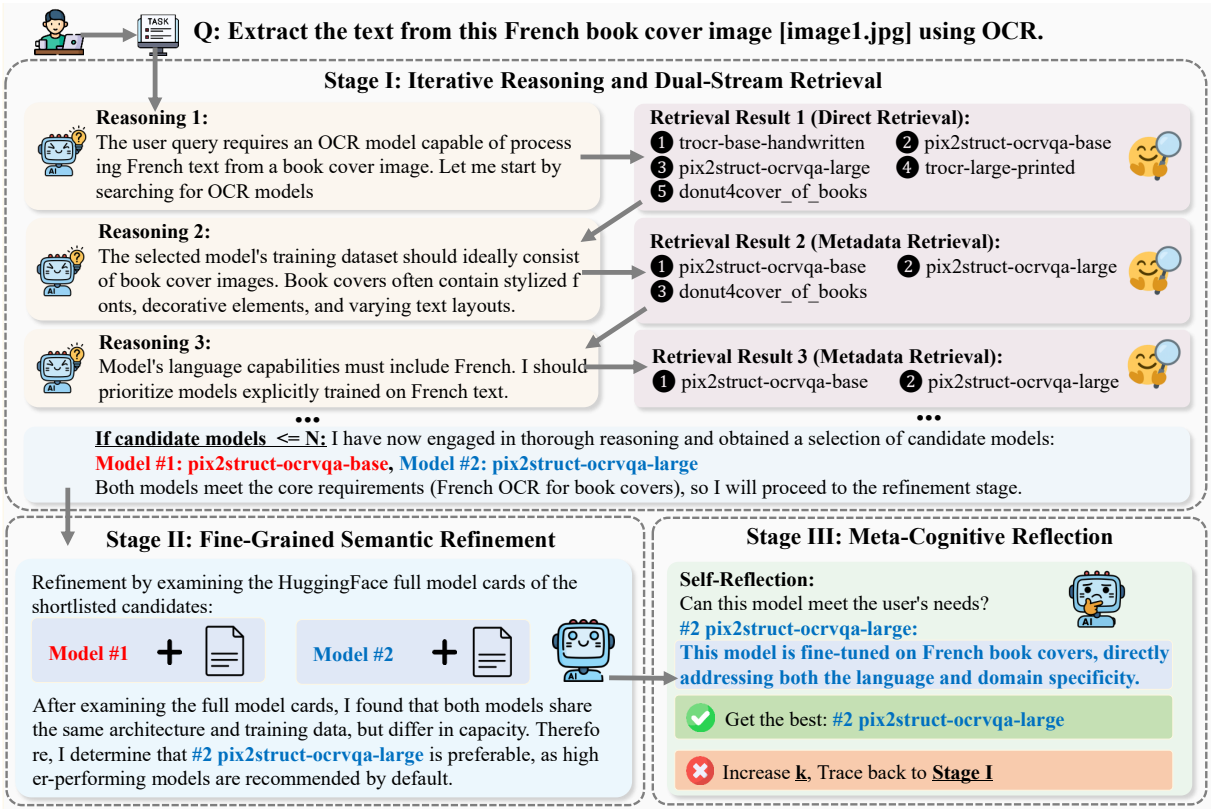


Figure 2: Workflow of HuggingR$^4$ with an example. The process includes: 1) Reasoning and Retrieval: Iterative querying and top-k candidate retrieval from the vector database. 2) Refinement: Fine-grained model selection via sliding window access to complete model cards. 3) Reflection: Self-evaluation to verify model suitability.

**Tool Selection vs. Model Selection.** Unlike prior work on API tool selection, which operates on discrete function categories (weather vs. email) with structured schemas, community model selection faces *continuous capability spaces* where thousands of models (e.g., 6,117 sentiment models) differ subtly in domain and architecture. This transforms the problem from category-based classification to fine-grained ranking under incomplete metadata and context-dependent performance. We provide a detailed comparison of these paradigm differences in Appendix D.2.

**LLMs for Reasoning.** While LLMs exhibit strong capabilities in reasoning (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023a; Kang et al., 2025; Zhang et al., 2024a; Yu et al., 2025; Tong et al., 2024; Yuan et al., 2025) and dynamic information retrieval (Huang et al., 2025; Li et al., 2025; Wu et al., 2025; Shi et al., 2025; Gerych et al., 2024), their application to model selection remains underexplored. Inspired by human iterative decision-making (Zhao et al., 2023), we uniquely frame community model selection as a step-by-step reasoning process rather than a single-step matching task.

# 3 Methodology

## 3.1 Task Formulation and Notation

We consider the task of model selection within a massive, evolving ecosystem (*e.g.*, Hugging Face). Let $\mathcal{D} = \{m_1, m_2, ..., m_n\}$ denote a repository containing candidate models, where $n$ is the number of models. Each model $m_i$ is characterized by a high-dimensional *model card* $C(m_i)$, which we define as a composite tuple:

$$C(m_i) = \langle \mathcal{A}_{\text{base}}, \mathcal{A}_{\text{struct}}, \mathcal{T}_{\text{desc}} \rangle, \tag{1}$$

where $\mathcal{A}_{\text{base}}$ represents atomic metadata (*e.g.*, unique identifier, usage statistics, and canonical task labels), $\mathcal{A}_{\text{struct}}$ the schematized attributes (*e.g.*, supported languages, dataset lineage, and licensing constraints), and $\mathcal{T}_{\text{desc}}$ unstructured semantic content (*e.g.*, technical specifications, architectural nuances, and qualitative performance metrics).

Given a user query $q$ that encodes latent functional requirements and domain constraints, the objective is to find the optimal model companions $\mathcal{M}^*$. We frame this as a maximization of an alignment function $\Phi$:

$$\mathcal{M}^* = \arg\max_{\mathcal{M} \subseteq \mathcal{D}} \Phi\left(q, C(\mathcal{M})\right). \tag{2}$$

In the context of repository-scale selection, $\Phi$ is non-trivial to compute due to three challenges. (1) *Informational Asymmetry*: the mapping between $q$ and $C(\mathcal{M})$ is often obscured by incomplete or noisy metadata in $\mathcal{A}_{\text{struct}}$. (2) *Computational Tractability*: the cardinality of $\mathcal{D}$ precludes the direct application of LLM-based reasoning across all $\mathcal{T}_{\text{desc}}$. (3) *Semantic Drift*: natural language requests $q$ often require iterative decomposition to align with the technical granularity of $C(\mathcal{M})$.

### 3.2 Stage I: Iterative Reasoning and Dual-Stream Retrieval

Stage I operationalizes a closed-loop reasoning-retrieval cycle designed to navigate the high-cardinality model space. Rather than treating retrieval as a static, one-shot operation, HuggingR[4] employs an agentic paradigm that autonomously refines search parameters based on intermediate feedback, progressively distilling the global repository into a high-precision candidate set.

**Semantic Distillation of Model Cards.** To maximize the signal-to-noise ratio within the prompt context, we perform semantic distillation on the raw model cards. For any $m_i \in \mathcal{D}$, we apply a transformation $\Psi : C(m_i) \to \tilde{C}(m_i)$ that prunes non-informative artifacts such as raw URLs, redundant citations, and organizational boilerplate, while canonicalizing technical specifications. This distillation ensures that the reasoning operates on high-density semantic representations, effectively mitigating prompt bloat and reducing token overhead by focusing on descriptors.

**Intent Decomposition and Query Synthesis.** At each iteration $t$, a reasoning agent $\mathcal{F}_{\text{reason}}$ synthesize a strategic search descriptor $s_t$ based on the user query $q$, the task category $c$, and the historical reasoning trace $H_{t-1}$. This process is formalized as:

$$s_t = \mathcal{F}_{\text{reason}}(q, H_{t-1}, c). \tag{3}$$

The agent performs three critical operations:

- **Functional decomposition:** translating user needs into discrete technical constraints (*e.g.*, "low-latency", "quantized", "domain-specific").
- **Lexical mapping:** generating targeted search terms to bridge user vernacular and technical terminology in repository metadata.
- **Policy selection:** adaptively choosing between semantic and metadata-constrained retrieval based on intent specificity.

**Dual-Stream Adaptive Retrieval.** To account for the inherent sparsity of repository metadata, we implement a *dual-stream retrieval* mechanism. Depending on the agent's assessment, the system selects the appropriate embeddings:

- **Direct stream**: encoding the full distilled description $\tilde{\mathcal{T}}_{\text{desc}}(m_i)$ into a dense vector $\mathbf{v}_{m_i}^{\text{full}} \in \mathbb{R}^d$, capturing broad functional nuances.
- **Metadata stream**: employing the structured attributes $\mathcal{A}_{\text{struct}}(m_i)$ to generate a constrained vector $\mathbf{v}_{m_i}^{\text{meta}} \in \mathbb{R}^d$, prioritizing hard-constraint satisfaction (*e.g.*, language, dataset or license).

We employ *multi-query augmentation* to enhance robustness. The system generates a set of $K$ semantically diverse variants $\mathcal{S}_t = \{s_t^{(1)}, \ldots, s_t^{(K)}\}$, which are concatenated into a composite query $s_t^* = \bigoplus_{i=1}^{K} s_t^{(i)}$. The final retrieval set $\mathcal{M}_{\text{reason}}$ is identified via cosine similarity:

$$\mathcal{M}_{\text{reason}} = \left\{ m_i \in \mathcal{D} \mid \text{sim}(s_t^*, \mathbf{v}_{m_i}^t) \in \text{Top-}k \right\}, \tag{4}$$

where $\text{sim}(\cdot)$ denotes the normalized inner product, and $\mathbf{v}_{m_i}^t$ is the embedding from the selected stream. The process terminates when $|\mathcal{M}_{\text{reason}}| \leq N$ (where $N$ is a predefined hyperparameter), ensuring the subsequent refinement stage remains computationally tractable.

**Failure Tracing and Backtracking.** To mitigate incomplete or misleading repository metadata, we introduce a failure tracing mechanism. It validates metadata stream searches by comparing their semantic density against a parallel direct stream. If the similarity divergence exceeds a threshold $\theta$, the system triggers a backtrack. This forces the agent to revise its query strategy, preventing retrieval errors from propagating to the refinement stage.

### 3.3 Stage II: Fine-Grained Semantic Refinement

While Stage I prunes the search space through coarse-grained vector similarity, Stage II facilitates fine-grained refinement. At this stage, the framework transitions from identifying relevant models to identifying the optimal one. This stage is critical because embedding-based retrieval often fails to distinguish between models with similar high-level descriptions but divergent technical nuances, such as specific quantization formats, library dependencies, or edge-case performance characteristics.

**High-Fidelity Candidate Adjudication.** Once the candidate set $\mathcal{M}_{\text{reason}}$ is reduced to a cardinality $|\mathcal{M}_{\text{reason}}| \leq N$, the refinement agent $\mathcal{F}_{\text{refine}}$ is granted access to the complete model cards $\tilde{C}(\mathcal{M}_{\text{reason}})$. These real model card provide the LLM with the necessary context to perform multi-dimensional cross-verification. The refinement process is formalized as a selection function:

$$\mathcal{M}_{\text{refine}} = \mathcal{F}_{\text{refine}}(q, \tilde{C}(\mathcal{M}_{\text{reason}})). \tag{5}$$

The agent performs a semantic analysis, evaluating candidates against the $q$ across several axes:

- **Technical compatibility:** ensuring the model's architecture and dependencies align with the user's infrastructure.
- **Performance benchmarking:** analyzing qualitative and quantitative metrics reported in the model cards to verify domain suitability.
- **Instruction adherence:** confirming that the model's intended use-case matches the operational requirements.

**Computational Tractability via the Sliding Window Mechanism.** We refer to the aforementioned approach as a sliding window mechanism: across Stage I iterations, the system progressively slides its focus across the model space, maintaining a fixed-size window $|\mathcal{M}_{\text{reason}}| \leq N$ of candidates for Stage II analysis. Unlike text-based sliding windows, this operates over the model space.

The refinement stage drives computational efficiency by restricting LLM analysis to $\mathcal{M}_{\text{reason}}$, capping context consumption at $O(N \cdot L)$ where $L$ is the average model card token length. Since $N$ is much smaller than the full repository size $|\mathcal{D}|$, this bypasses the $O(|\mathcal{D}| \cdot L)$ complexity of direct Prompting and avoids "lost-in-the-middle" effects in long-context processing. Crucially, the approach decouples computational cost from repository size, enabling our agent to scale to millions of models.

### 3.4 Stage III: Meta-Cognitive Reflection

The final stage introduces a meta-cognitive reflection layer to mitigate hallucinations and false positives caused by partial information in earlier phases. Because Stage I reasons primarily over distilled metadata, the system may occasionally converge on a "local maximum" that lacks specific functional requirements hidden in deep documentation.

To address this, $\mathcal{F}_{\text{reflect}}$ executes a "zero-trust" audit of the refined candidate $\mathcal{M}_{\text{refine}}$ against the original query $q$:

$$\mathcal{M}^* = \mathcal{F}_{\text{reflect}}(q, \tilde{C}(\mathcal{M}_{\text{refine}})). \tag{6}$$

The agent identifies any misalignment between actual model capabilities (*e.g.*, tensor constraints or licensing) and user intent. If a discrepancy is detected, it documents why $\mathcal{M}_{\text{refine}}$ is insufficient.

**Dynamic Sliding Window and State Transition** The core innovation of our reflection process lies in its integration with the Sliding Window Strategy. Rather than terminating upon a failed validation, the framework initiates a state-space transition:

1. **Freezing:** The current windows $\mathcal{M}_{\text{reason}}$ is marked as "exhausted" and cached in the reasoning history $H_t$ to prevent redundant processing.
2. **Sliding:** The selection window $\mathcal{W}_t$ slides forward by $N$ positions in the global ranked list:

$$\mathcal{W}_{t+1} \leftarrow \text{Slide}(\mathcal{W}_t, N). \tag{7}$$

3. **Recursive Recovery:** The system re-enters Stage I. The reasoning agent $\mathcal{F}_{\text{reason}}$ leverages previous failure traces to adjust search parameters and explore the next partition under improved guidance.

This recursive mechanism ensures recall robustness. By treating the repository as sequential windows, the framework recovers from initial retrieval biases without the heavy costs of global re-analysis, balancing exhaustive search with efficiency.

## 4 Experiments

### 4.1 Dataset Construction

To evaluate our approach, we constructed a comprehensive dataset comprising three components. The complete dataset construction process and associated prompts are detailed in the Appendix B.

**Model Cards.** The dataset includes 37 different tasks from HuggingFace. For each task, we ranked models based on the number of likes and selected the top-ranking models, resulting in a candidate pool of 1,110 models. The complete task distribution is provided in Appendix Table 11.

**Single-Task User Requests.** We constructed 1,016 requests covering various domains. Each was annotated by experts across two dimensions: *workability* (basic execution success) and *reasonability* (alignment with domain-specific preferences).

| LLM Agent | HuggingGPT (Shen et al., 2023) | | HuggingR$^{4*}$ | | HuggingR$^{4}$ | |
|---|---|---|---|---|---|---|
| | Workability | Reasonability | Workability | Reasonability | Workability | Reasonability |
| GPT-4o-mini | 65.52 | 49.21 | 84.77 | 75.56 | 92.03 | 82.46 |
| GPT-4o | 75.20 | 60.70 | 84.15 | 75.16 | 91.14 | 82.09 |
| GPT-4.1-mini | 66.44 | 50.79 | 85.00 | 75.79 | 91.04 | 82.38 |
| GPT-4.1 | 74.80 | 60.73 | 85.27 | 75.72 | 91.43 | 83.86 |
| GPT-5.4-mini | 70.47 | 56.50 | 85.13 | 77.26 | 91.83 | 83.96 |
| GPT-5.4 | 75.20 | 61.12 | 87.40 | 77.36 | **93.01** | **84.25** |
| Qwen3.6-flash | 73.40 | 58.96 | 84.05 | 73.81 | 91.24 | 81.49 |
| Claude-Sonnet-4 | 78.64 | 64.86 | 86.02 | 76.19 | 90.85 | 81.59 |
| Deepseek-v4-pro | 78.14 | 67.22 | 87.11 | 77.06 | 92.12 | 83.07 |
| Qwen2.5-7b | 56.59 | 41.83 | 82.78 | 73.82 | 85.73 | 76.31 |

Table 1: Performance comparison of different LLMs on the single-task dataset. We evaluate HuggingGPT and our HuggingR$^4$ across proprietary and open-source models; HuggingR$^{4*}$: the retrieval-only version without refinement.

**Multi-Task User Requests.** We leveraged GPT-4o to combine single-task requests into 13,383 multi-task scenarios (2–5 tasks each). Five domain experts manually verified all scenarios based on three criteria: (1) *domain consistency*, (2) *semantic coherence*, and (3) *technical feasibility*. Requests failing any criterion were revised or discarded, ensuring rigorous quality control.

### 4.2 Experimental setup

We evaluate HuggingR$^4$ across ten LLMs from the GPT, Claude, Qwen, and DeepSeek series. These models represent diverse architectures and varying scales of reasoning capabilities.

We set the Multi-Query count to 4 using `text-embedding-3-large`. For the sliding window, parameters $N$ and $k$ are set to 3 and 5, respectively, with a failure tracing threshold $\theta = 80\%$. To ensure statistical significance, we report results averaged over three independent runs. Complete parameter details are in Appendix D.1.

We employ HuggingGPT as our baseline, as it is a representative and the only query-driven model selection method. Following the official implementation, we apply a 100-character truncation limit to each model card.

Additionally, based on HuggingR$^4$, we introduce a pure retrieval variant, denoted as HuggingR$^{4*}$, which retains only the multi-stage retrieval mechanism and achieves maximal token savings by removing the refinement and Reflection.

### 4.3 Evaluation Metrics

We evaluate model selection quality using two metrics: *Workability* (task compatibility) and *Reasonability* (domain appropriateness).

- **Workability (Task Compatibility):** We define a model as *workable* if it satisfies three binary criteria: (1) task type matches the request, (2) input/output formats are compatible, and (3) the model is executable on Hugging Face Hub or locally. A model receives a Workability score of 1 if all criteria are met, 0 otherwise. This yields an average of 8.3 workable models per request.
- **Reasonability (Domain Appropriateness):** Among workable models, we define a model as *reasonable* if it satisfies two criteria: (1) trained on domain-relevant datasets (verified through model card metadata), and (2) achieves competitive performance (within top-5 reported metrics among candidate models). This yields an average of 2.1 reasonable models per request.

Detailed annotation guidelines and inter-annotator agreement are provided in Appendix C.

### 4.4 Results and Analysis

Table 1 compares HuggingR$^4$ against HuggingGPT across diverse LLMs. HuggingR$^4$ consistently outperforms the baseline across all models and metrics. Notably, it achieves peak performance on the latest GPT-5.4 (93.01% workability, 84.25% reasonability). Moreover, models like GPT-4o-mini exhibit massive absolute gains of 26.51 and 33.25 percentage points over HuggingGPT. Furthermore, the retrieval-only variant HuggingR$^{4*}$ demonstrates stable performance across backbones, as it relies primarily on Multi-Query Generation rather than deep reasoning capabilities.

While stronger LLMs generally elevate baseline results, framework effectiveness varies due to distinct reasoning patterns. For instance, despite a strong baseline, Claude-Sonnet-4 yields suboptimal gains due to over-analysis during reflection, leading to overly strict candidate filtering. In contrast, advanced reasoning models like Deepseek-v4-

| Method | Workability | Reasonability |
|---|---|---|
| *HuggingR$^{4*}$* | | |
| text-embedding-3-large | 84.77 | 74.56 |
| jina-embeddings-v3 | 84.81 | 73.75 |
| text-embedding-3-small | 83.04 | 70.93 |
| *HuggingR$^{4}$* | | |
| text-embedding-3-large | **92.03** | **82.46** |
| jina-embeddings-v3 | 90.62 | 80.67 |
| text-embedding-3-small | 90.12 | 79.95 |

Table 2: Performance of HuggingR$^{4*}$ and HuggingR$^{4}$ with different embedding models on GPT-4o-mini.

| Method | Workability | Reasonability |
|---|---|---|
| HuggingGPT | 68.08 | 51.44 |
| HuggingR$^{4*}$ | 78.27 | 69.86 |
| HuggingR$^{4}$ | **85.03** | **75.73** |

Table 3: Performance comparison on the multi-task data using GPT-4o-mini with text-embedding-3-large.

| Method (*w/ GPT-4o-mini*) | Work. | Reas. |
|---|---|---|
| Standard RAG | 65.94 | 49.40 |
| RAG + Cohere Rerank | 75.59 | 52.06 |
| ReAct (Yao et al., 2022) | 81.67 | 69.13 |
| Reflexion (Shinn et al., 2023) | 81.83 | 71.25 |
| **HuggingR$^{4}$ (Ours)** | **92.03** | **82.46** |

Table 4: Performance comparison with general-purpose retrieval and agent frameworks.

| Method | Input Type | # Tasks | # Models | Online |
|---|---|---|---|---|
| AutoMRM | Dataset | 1 | 11 | ✗ |
| TransferGraph | Dataset | 2 | 348 | ✗ |
| ModsNet | Dataset | 3 | 3,194 | ✗ |
| HuggingGPT | User Query | 24 | >10K | ✓ |
| **HuggingR$^{4}$** | **User Query** | **37** | **>10K** | ✓ |

Table 5: Comparison of HuggingR$^{4}$ with Other Paradigms

pro and GPT-5.4 seamlessly adapt to the iterative refinement. Conversely, our method drastically empowers smaller open-source models: on Qwen2.5-7B, it achieves 85.73% workability. These observations underscore the importance of balancing thorough analysis with reasoning efficiency.

We also evaluate the performance of our methods with different embedding models. As shown in Table 2, the performance of HuggingR$^{4}$ generally scales with the capacity of embedding models, while the consistently high performance across all models (90%+ workability) demonstrates the robustness of our approach to different semantic encoders. Notably, even with the smaller text-embedding-3-small, HuggingR$^{4}$ achieves 90.12% workability, only 1.91% lower than the best configuration, demonstrating that our method provides flexibility in balancing computational efficiency and performance. This robustness enables practitioners to select embedding models based on their specific deployment constraints without significantly compromising recommendation quality.

We further evaluate HuggingR$^{4}$ in multi-task scenarios using HuggingGPT's task planning. As shown in Table 3, HuggingR$^{4}$ yields 85.03% workability and 75.73% reasonability. While continuing to substantially outperform baselines, this represents a notable drop from our single-task results (92.03% and 82.46%, respectively). This performance gap stems primarily from the heightened complexity of task decomposition in multi-task scenarios, indicating that our orchestration and dependency management require future optimization.

To validate our architectural design, we compare HuggingR$^{4}$ against established retrieval and agent paradigms (Table 4). Standard RAG and RAG with Cohere Rerank struggle significantly (65.94% and 75.59% workability), proving that naive text matching is insufficient for navigating continuous model capability spaces. While general-purpose agents like ReAct (Yao et al., 2022) and Reflexion (Shinn et al., 2023) introduce reasoning and self-correction, elevating workability to roughly 81%, they still fall short in repository-scale selection. This highlights the absolute necessity of our domain-specific fine-grained refinement and sliding-window reflection mechanisms.

### 4.5 Comparison with Other Paradigms

Table 5 highlights fundamental differences between two paradigms. Data-driven methods (AutoMRM (Li et al., 2023b), TransferGraph (Li et al., 2024), ModsNet (Wang et al., 2023b)) rely on expert-extracted dataset meta-features and offline training, restricting them to narrow domains (typically 1–3 tasks). In contrast, our query-driven framework offers three key advantages: **(1) Usability:** Non-experts can express needs via natural language (*e.g.*, "Chinese movie review sentiment"); **(2) Adaptability:** It seamlessly integrates with rapidly evolving online repositories without retraining; and **(3) Scalability:** It comprehensively handles 37 diverse tasks across 10K+ models.

## 5 Ablation Study

### 5.1 Impact of Individual Modules

Table 6 presents our module ablation study. For HuggingR$^{4*}$, removing Semantic Distillation significantly degrades workability and reasonability

| Method | Workability | Reasonability |
|---|---|---|
| *HuggingR*$^{4*}$ | **84.77** | **75.56** |
| w/o Multi-Query | 82.94 | 71.63 |
| w/o Semantic Distillation | 81.36 | 71.26 |
| w/o Metadata Stream | 78.97 | 69.82 |
| *HuggingR*$^{4}$ | **92.03** | **82.46** |
| w/o Failure Tracing | 88.09 | 77.66 |
| w/o Self-reflection | 85.33 | 80.21 |
| w/o Sliding Window Strategy | 90.98 | 81.30 |

Table 6: Ablation study on individual module contributions using GPT-4o-mini with text-embedding-3-large.

| $N$ ($k=5$) | | | $k$ ($N=3$) | | |
|---|---|---|---|---|---|
| $N$ | Workability | Reasonability | $k$ | Workability | Reasonability |
| 1 | 88.98 | 79.82 | 3 | 90.98 | 81.30 |
| 2 | 91.54 | 81.99 | 4 | 91.73 | 82.14 |
| 3 | 92.03 | 82.46 | 5 | **92.03** | **82.46** |
| 4 | **92.35** | **83.07** | 6 | 91.63 | 81.79 |
| 5 | 92.32 | 82.48 | 7 | 91.34 | 81.99 |

Table 7: Performance impact of hyperparameters $N$ (window size) and $k$ (number of retrieved candidate models) in the sliding window strategy using GPT-4o-mini with text-embedding-3-large.

(-3.41 and -4.30 points), as it is vital for enhancing model card embedding quality. Disabling the Metadata stream causes even steeper drops (-5.80 and -5.74 points), highlighting its necessity for filtering relevant tasks to narrow the search space.

For the full HuggingR$^4$ framework, deactivating Failure Tracing yields the most substantial performance degradation (-4.80 points in reasonability). This highlights its critical role in catching undetected metadata mismatches and preventing error propagation. Furthermore, removing either Self-reflection or Sliding Window Strategy impairs performance. Collectively, these components empower the system to explore beyond initially retrieved candidates, facilitating the multi-round reasoning necessary for robust model selection.

### 5.2 Analysis of the Sliding Window Strategy

We further investigate the impact of the two key hyperparameters in the sliding window strategy: $k$ and $N$. As shown in Table 7, increasing $N$ consistently improves both Workability and Reasonability scores, with peak performance at $N$=4. However, larger $N$ values also lead to higher token due to more full model cards being processed in stage II. To strike a balance between performance and efficiency, we select $N$=3 as the default configuration. In contrast, varying $k$ shows a relatively stable trend, where performance improves up to $k$=5 but slightly degrades when $k$ exceeds 5. This indicates that an appropriate number of initial candidates is beneficial, but overly large candidate pools may introduce noise and reduce effectiveness.

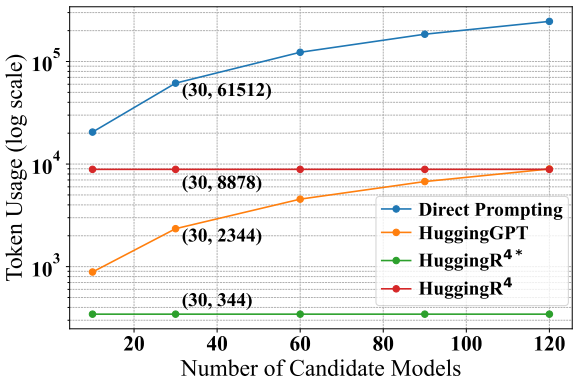


Figure 3: Token usage comparison across different numbers of candidate models using GPT-4o-mini with text-embedding-3-large (log scale).

### 5.3 Analysis of Token Usage

In practical applications, token consumption is a key efficiency indicator. A major advantage of our approach is that it avoids increasing token usage as the number of candidate models grows, making it highly scalable. As shown in Figure 3, both Direct Prompting and HuggingGPT consume significantly more tokens when handling more models, due to their reliance on directly embedding large model cards into the prompt. In contrast, HuggingR$^4$'s token consumption is independent of the number of candidate models, as they operate on retrieval results. Notably, HuggingR$^{4*}$ achieves high model selection performance while consuming only a minimal amount of tokens. Due to its multi-step reasoning process, HuggingR$^4$ requires slightly more tokens. However, it achieves superior performance when handling a larger number of candidate models. Therefore, our proposed methods not only excel in model selection capability but also provide significant advantages in token consumption.

## 6 Conclusion

We present HuggingR$^4$, a progressive reasoning framework for selecting optimal AI models from massive repositories, alongside ModelSelectBench, the first rigorously annotated benchmark for model select. Extensive experiments demonstrate that HuggingR$^4$ significantly outperforms existing baselines by accurately capturing fine-grained user intents. Crucially, its multi-stage design completely decouples token consumption from the candidate pool size, delivering state-of-the-art accuracy with exceptional scalability and efficiency.

## Limitations

The primary limitation of HuggingR$^4$ is the lack of adaptive policy optimization. In future work, we plan to employ agentic reinforcement learning (RL) to dynamically learn and optimize these model selection policies.

## References


Arash Ahmadi, Sarah Sharif, and Yaser M Banad. 2025. Mcp bridge: A lightweight, llm-agnostic restful proxy for model context protocol servers. *arXiv preprint arXiv:2504.08999*.

Antonio Contreras Ortiz, Ricardo Rioda Santiago, Daniel E Hernandez, and Miguel Lopez-Montiel. 2025. Multiclass evaluation of vision transformers for industrial welding defect detection. *Mathematical and Computational Applications*, 30(2):24.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. 2025. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*.

Tiantian Gan and Qiyao Sun. 2025. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation. *arXiv preprint arXiv:2505.03275*.

Walter Gerych, Yara Rizk, Vatche Isahagian, Vinod Muthusamy, Evelyn Duesterwald, and Praveen Venkateswaran. 2024. Who knows the answer? finding the best model and prompt for each query using confidence-based search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18065–18072.

Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2023. Laf: labeling-free model selection for automated deep neural network reusing. *ACM Transactions on Software Engineering and Methodology*, 33(1):1–28.

Ziyang Huang, Wangtao Sun, Jun Zhao, and Kang Liu. 2025. Improve rule retrieval and reasoning with self-induction and relevance reestimate. *arXiv preprint arXiv:2505.10870*.

Likith Anoop Kadiyala, Omer Mermer, Dinesh Jackson Samuel, Yusuf Sermet, and Ibrahim Demir. 2024. The implementation of multimodal large language models for hydrological applications: A comparative study of gpt-4 vision, gemini, llava, and multimodal-gpt. *Hydrology*, 11(9):148.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320.

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

Mikhail V Koroteev. 2021. Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023a. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Yiyan Li, Si Zhang, Rongxing Hu, and Ning Lu. 2021. A meta-learning based distribution system load forecasting model selection framework. *Applied Energy*, 294:116991.

Zhaotian Li, Binhang Qi, Hailong Sun, and Xiang Gao. 2023b. Automrm: A model retrieval method based on multimodal query and meta-learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1228–1237.

Ziyu Li, Hilco Van Der Wilk, Danning Zhan, Megha Khosla, Alessandro Bozzon, and Rihan Hai. 2024. Model selection with model zoo via graph learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1296–1309. IEEE.

Zhuoran Liu, Danpei Zhao, Bo Yuan, and Zhiguo Jiang. 2025. Rescueadi: Adaptive disaster interpretation in remote sensing images with autonomous agents. *IEEE Transactions on Geoscience and Remote Sensing*.

Daohan Lu, Sheng-Yu Wang, Nupur Kumari, Rohan Agarwal, Mia Tang, David Bau, and Jun-Yan Zhu. 2023. Content-based search for deep generative models. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12.

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A Burke. 2025. Scalemcp: Dynamic and auto-synchronizing model context protocol tools for llm agents. *arXiv preprint arXiv:2505.06416*.

Mi Luo, Fei Chen, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Jiashi Feng, and Zhenguo Li. 2020. Metaselector: Meta-learning for recommendation with user-level adaptive model selection. In *Proceedings of The Web Conference 2020*, pages 2507–2513.

Vicent Sanz Marco, Ben Taylor, Zheng Wang, and Yehia Elkhatib. 2020. Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1):1–28.

Meriem Mastouri, Emna Ksontini, and Wael Kessentini. 2025. Making rest apis agent-ready: From openapi to model context protocol servers for tool-augmented llms. *arXiv preprint arXiv:2507.16044*.

Linghan Meng, Yanhui Li, Lin Chen, Zhi Wang, Di Wu, Yuming Zhou, and Baowen Xu. 2021. Measuring discrimination to boost comparative testing for multiple deep learning models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 385–396. IEEE.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

Yaorui Shi, Shihan Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. 2025. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. *arXiv preprint arXiv:2505.11277*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Ben Taylor, Vicent Sanz Marco, Willy Wolff, Yehia Elkhatib, and Zheng Wang. 2018. Adaptive selection of deep learning models on embedded systems. *arXiv preprint arXiv:1805.04252*.

Yongqi Tong, Sizhe Wang, Dawei Li, Yifan Wang, Simeng Han, Zi Lin, Chengsong Huang, Jiaxin Huang, and Jingbo Shang. 2024. Optimizing language model's reasoning abilities with weak supervision. *arXiv preprint arXiv:2405.04086*.

Kapil Wanaskar, Gaytri Jena, and Magdalini Eirinaki. 2025. Multimodal benchmarking and recommendation of text-to-image generation models. *arXiv preprint arXiv:2505.04650*.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2023a. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*.

Mengying Wang, Sheng Guan, Hanchao Ma, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alp Sehirlioglu, and Yinghui Wu. 2023b. Selecting top-k data science models by example dataset. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2686–2695.

Song Wang, Li Sun, Wei Fan, Jun Sun, Satoshi Naoi, Koichi Shirahata, Takuya Fukagai, Yasumoto Tomita, Atsushi Ike, and Tetsutaro Hashimoto. 2017. An automated cnn recommendation system for image classification tasks. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 283–288. IEEE.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Junde Wu, Jiayuan Zhu, and Yuyuan Liu. 2025. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv preprint arXiv:2502.04644*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Erxin Yu, Jing Li, Ming Liao, Qi Zhu, Boyang Xue, Minghui Xu, Baojun Wang, Lanqing Hong, Fei Mi, and Lifeng Shang. 2025. Self-error-instruct: Generalizing from errors for llms mathematical reasoning. *arXiv preprint arXiv:2505.22591*.

Yige Yuan, Teng Xiao, Shuchang Tao, Xue Wang, Jinyang Gao, Bolin Ding, and Bingbing Xu. 2025. Incentivizing strong reasoning from weak supervision. *arXiv e-prints*, pages arXiv–2505.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Wenling Zhang, Yixiao Li, Zhaotian Li, Hailong Sun, Xiang Gao, and Xudong Liu. 2024b. Modelgalaxy: A versatile model retrieval platform. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2771–2775.

James Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Qizhe Xie. 2023. Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*.

Zhi Zhou, Lan-Zhe Guo, Peng-Xiao Song, and Yu-Feng Li. 2024. You only submit one image to find the most suitable generative model. *arXiv preprint arXiv:2412.12232*.

# A Methodology Implementation Details

## A.1 Methodology Prompt

Our framework structures the LLM as a multi-stage reasoning assistant. Given user query $q$ and optional task label, the system prompt orchestrates:

**Stage I: Iterative Retrieval.** The system generates targeted search queries and applies filtering tools across multiple rounds, adaptively refining the strategy (broadening if too few results, narrowing if too many) until obtaining $|\mathcal{M}_{\text{reason}}| \leq N$ candidates. The prompt prohibits inferring capabilities from model names, enforcing metadata validation.

**Stage II: Comparative Refinement.** With $|\mathcal{M}_{\text{reason}}| \leq N$ candidates, a refinement tool performs comparison across: (1) task performance, (2) efficiency, (3) compatibility, and (4) community adoption. The LLM selects the optimal model $m^*$ through pairwise analysis with justification.

**Stage III: Meta-Cognitive Reflection.** A reflection agent adversarially audits $\mathcal{M}_{\text{reflect}}$ against requirements (language, dataset, size, license, hardware, special constraints). If verified, the system returns `MODEL_NAME`; if any check fails, it outputs `UNCERTAIN` with failure trace, triggering fallback Stage I with augmented query. The prompt as showcased below:

> You are an expert reasoning assistant for selecting models on HuggingFace. Your task is to **progressive reasoning**, iteratively filtering out the most appropriate model based on the user's input: {user query}.
> **Stage I: Iterative Reasoning and Dual-Stream Retrieval**
> - Carefully analyze the user's requirements and generate targeted search queries. - Use available filtering tools iteratively (e.g., filter by task, language, size, license).
> - Refine your search strategy based on intermediate results, if you get too many results, add more constraints; if you get zero results, broaden your search.
> - Continue retrieval cycles until you obtain $N$ or fewer candidate models.
> - Critical: Do NOT assume model capabilities based solely on model names. Always validate against model metadata.
> **Stage II: Fine-Grained Semantic Refinement**
> - Once you have $\leq \{N\}$ candidate models, use refinement tool to provide a selection of the one best model.
> - Analyze each candidate across multiple dimensions: Task alignment and performance benchmarks. Efficiency (inference speed, memory usage). Compatibility (frameworks, hardware). Community adoption and maintenance status.
> - Select the best model based on this comprehensive analysis.
> **Stage III: Meta-Cognitive Reflection**
> - Verify the selected model satisfies all user criteria: language, dataset compatibility, model size, type, and special requirements
> - If the model meets all these criteria, and you have information to recommend this model, return the final result: `MODEL_NAME`.
> - If any of the criteria are not satisfied, system output: `UNCERTAIN`.

## A.2 Iterative Reasoning and Dual-Stream Retrieval Prompts

**Semantic Distillation of Model Cards.** Given that raw model cards often contain verbose documentation, redundant metadata, and heterogeneous formatting that can hinder retrieval efficiency, we perform semantic distillation on all model cards before framework execution. Specifically, we apply a transformation $\Psi : C(m_i) \rightarrow \tilde{C}(m_i)$ that removes non-informative artifacts (e.g., raw URLs, redundant citations, organizational boilerplate) while canonicalizing technical specifications and extracting essential fields. This preprocessing step maximizes the signal-to-noise ratio by retaining only the core information needed for model selection, such as model purpose, capabilities, and key technical descriptors, while eliminating elements that contribute to prompt bloat without aiding the reasoning process:

> **Prompt:** You are a helpful assistant specialized in simplifying HuggingFace model cards. Your task is to extract only the fields relevant to model selection: id, downloads, likes, pipeline_tag, task, meta, language, datasets, and description. For the description field, simplify the content by keeping only the most essential information that helps users understand the model's purpose and use case. Output only a clean, valid JSON object with the selected and simplified fields.

**Dual-Stream Adaptive Retrieval Prompt.** To enable flexible and precise model filtering from the large-scale repository while accounting for the inherent sparsity of repository metadata, we instruct the LLM to utilize a dual-stream retrieval mechanism in the main prompt. The system autonomously selects between two retrieval streams based on query analysis: direct stream and metadata stream. The LLM autonomously selects and combines these tools based on query analysis, enabling adaptive retrieval strategies.

The retrieval prompts are structured as follows:

> **Tool 1: Direct Similarity Retrieval**
> - Used to find models similar to the user's query.
> **Input:** The retrieval query:
>
> ```
> <|begin_similarity_query|>
> Retrieval Query
> <|end_similarity_query|>
> ```
>
> **Output:** The top-$k$ most relevant models:
>
> ```
> <|begin_similarity_result|>
> model 1, model 2 ...
> <|end_similarity_result|>
> ```

> **Tool 2.1: Metadata Language Retrieval**
> - Filters models based on the specified language.
> **Input:** The ISO language code of user query:
>
> ```
> <|begin_language_query|>
> ISO language code
> <|end_language_query|>
> ```
>
> **Output:** The Top-k models:
>
> ```
> <|begin_language_result|>
> model 1, model 2 ...
> <|end_language_result|>
> ```
>
> **Tool 2.1: Metadata Dataset Retrieval**
> - *Filters models based on the dataset requirements.*
> **Input:** Describe the required dataset:
>
> ```
> <|begin_dataset_query|>
> Describe the required dataset
> <|end_dataset_query|>
> ```
>
> **Output:** The Top-k models:
>
> ```
> <|begin_dataset_result|> model 1,
> model 2 ... <|end_dataset_result|>
> ```

> **Special Case:** If some model cards are missing dataset labels, and no similarity search results are returned for the dataset retrieval, mark this dataset retrieval as untrustworthy.

**Multi-Query Augmentation.** To enhance query diversity and improve retrieval coverage, we generate multiple search queries from different perspectives before performing the model retrieval. This multi-query approach allows us to capture various aspects of the user's intent and increases the likelihood of finding relevant models through different semantic angles. In our implementation, we set $K = 4$ and use the following prompt to transform the user query into four distinct search queries:

> **Prompt:** You are a helpful assistant in generating multiple search queries based on a single query entered by a user. This will allow us to use the user query to more accurately find the most similar model via Hugging Face model card embeddings. The generated queries should characterize the model itself (e.g., inputs to the model, uses of the model, datasets, kinds of languages, etc.) as needed by the user. You only need to output these $K$ queries, each on a new line, with no other information.
> Generate multiple search queries related to: {question}
> **Output:** (4 queries)

The prompt instructs the LLM to focus on different model characteristics such as input types, use cases, training datasets, and supported languages. The output consists of four queries, each presented on a separate line without additional formatting or explanations. These diversified queries are then used in parallel during the reasoning-retrieval phase to maximize the breadth of candidate model discovery.

### A.3 Fine-Grained Semantic Refinement Prompts

**Refinement Tool Prompt.** In the refinement step, the system obtains complete model cards for the candidate models identified in the coarse-grained retrieval phase. This enables more informed model selection by providing comprehensive descriptions rather than the simplified representations used in the initial retrieval.

The refinement process utilizes a specialized tool that fetches detailed model information from

the vector database. This tool takes the candidate model list from Stage I and returns their complete model cards, allowing the LLM to make more nuanced decisions based on full specifications, capabilities, and usage guidelines:

> **Tool 4: Get a detailed description of the model:**
> - Used to get the full model cards.
> **Input:** Output model list of stage I:
>
> ```
> <|begin_descriptions_query|>
> model 1, model 2, model 3
> <|end_descriptions_query|>
> ```
>
> **Output:** Complete description of these models:
>
> ```
> <|begin_descriptions_result|>
> {description 1}, {description 2},
> {description 3}
> <|end_descriptions_result|>
> ```

### A.4 Meta-Cognitive Reflection Prompts

**Reflection Failure Handling.** When the reflection module determines that the models selected in Stage II are inadequate for the user's request, the main prompt outputs `UNCERTAIN`, triggering a backtrack to Step 1 with an expanded search scope. The system employs the following prompt to guide the LLM through this recovery process:

> **Prompt:** Now the system has completed a full turn. The self-reflection module believes that {model 1} is not capable of doing this job. Please return to the Reasoning and Retrieval stage and re-screen other models. The system has automatically updated your previous query results. Please go through Stage I.

This mechanism ensures that when initial model selections prove insufficient, the framework can systematically explore alternative candidates while maintaining the context of previously evaluated models, thereby improving the overall success rate of model discovery.

### A.5 Detailed Workflow of the Sliding Window Strategy

To further elucidate the dynamic context management of HuggingR$^4$, Figure 4 provides a visual representation of our sliding window strategy. This mechanism is fundamentally designed to decouple token consumption from the massive size of the community repository, resolving the prompt bloat issue inherent to existing query-driven methods.

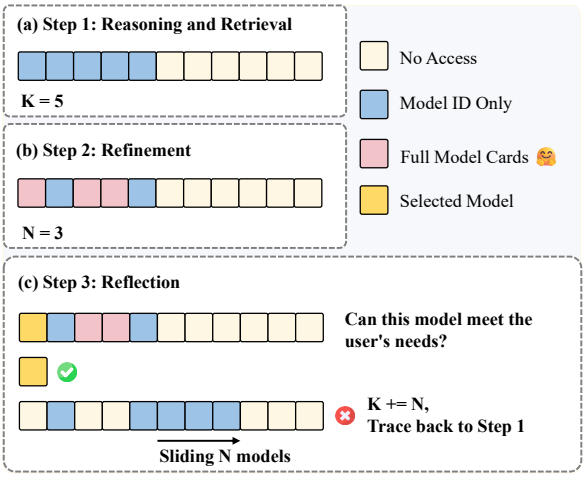


Figure 4: The sliding window strategy treats each model as a window and ranks them in descending order of similarity scores. The color of each window indicates the system's access level to the corresponding model card during selection.

As depicted in the figure, the repository is conceptualized as a continuous array of candidate models, ranked from left to right in descending order of their initial semantic similarity scores. Each block represents a candidate model, and its color strictly encodes the system's access privilege to the corresponding metadata during the decision process.

The strategy operates through a multi-stage state transition, efficiently balancing computational overhead and selection accuracy:

**Step 1: Coarse-Grained Reasoning and Retrieval.** In the initial phase, the system identifies the Top-$K$ candidates (e.g., $K = 5$). To strictly control token consumption, the LLM is only granted access to the *Model IDs* and highly distilled metadata of these candidates (represented by the blue windows). The vast majority of the repository remains unaccessed (cream windows), preventing massive context overload.

**Step 2: Fine-Grained Refinement.** Once the initial metadata screening isolates $N$ or fewer candidate models, the access privilege is elevated: the system fetches the *full model cards* for these candidates (represented by the pink windows). By granting full access only to this highly restricted subset, the LLM can perform an in-depth, comparative analysis of technical nuances (*e.g.*, dependencies, quantization formats) without exploding the token count.

**Step 3: Reflection and State Transition.** After the refinement stage, the LLM selects the single most optimal model (dark yellow window). Crucially, the system then executes a "zero-trust" reflection to verify if this specific model can genuinely satisfy the user's intricate requirements.

- *Success Path:* If the selected model passes the reflection audit (indicated by the green checkmark), the search terminates successfully, and the model is outputted.
- *Failure and Sliding Path:* If the model fails the audit (indicated by the red cross), a backtracking event is triggered. To prevent the LLM from entering a hallucination loop or repeatedly selecting the same suboptimal candidates, the currently evaluated models (pink/yellow windows) are frozen and marked as "exhausted". The retrieval window then dynamically slides forward along the ranked list by $N$ positions ($K \leftarrow K + N$). The system traces back to Step 1, exposing a fresh, unexplored partition of candidate models to the agent.

This recursive sliding mechanism is the cornerstone of HuggingR$^4$'s robustness. By evaluating the repository via sequential, carefully managed windows rather than a global one-shot prompt, the framework can gracefully recover from initial retrieval biases. It guarantees a systematic exploration of the continuous capability space, ensuring optimal model selection while maintaining a strictly bounded token consumption per iteration.

# B Data Construction and Details

## B.1 Model Cards

This section describes the construction process of model cards for the 1,110 HuggingFace models in our dataset. Each model card is systematically organized to capture essential metadata and functional descriptions that enable effective model selection and retrieval.

**Model Card Structure.** Following a similar approach to HuggingGPT, we construct model cards by extracting and organizing information from the HuggingFace model hub. Each card contains structured metadata including model type, supported tasks, performance metrics, and detailed descriptions. The standardized format ensures consistent information representation across all models in our database.

This is the JSON format of the specific model card data:

**JSON:**

```
{
  "id": "Model's ID",
  "task": "Model's task label",
  "downloads": "Model downloads",
  "likes": "Number of model likes",
  "meta": {
    "language": Supported languages,
    "datasets": Datasets used",
    "license": "Model's license",
    "tags": ["List of model tags"],
    ...
  },
  "description": "Model description"
  ...
}
```

**Task Categories Coverage.** Our model cards encompass 37 distinct task categories, covering a comprehensive range of machine learning applications from natural language processing to computer vision and audio processing. Table 11 provides a complete overview of all supported tasks in our dataset, demonstrating the diversity and breadth of models available for selection.

**Single-Task User Requests.** We systematically constructed 1,016 single-task user requests based on the characteristics of selected model cards from the HuggingFace repository. The construction process involved analyzing the capabilities and specializations of various models across different domains and task types. For each task category, we generated approximately 30 user requests, ensuring comprehensive coverage of different use cases, complexity levels, and domain-specific requirements. This approach allows us to evaluate our framework's performance across diverse scenarios that users might encounter in practice.

The user requests were crafted to vary in specificity and complexity, ranging from straightforward task descriptions (e.g., "I need a model for sentiment analysis") to more nuanced requirements that include domain constraints, performance preferences, or specific technical requirements (e.g., "I need a lightweight sentiment analysis model optimized for financial text that can run on mobile devices").

**Data Structure.** Each single-task user request in our dataset follows a structured format:

> **JSON:**
>
> ```
> {
>  "request": "User Request",
>  "Task_label": {
>  "workable": [Workable models],
>  "reasonable": [Reasonable models]
>  }
> }
> ```

### B.2 Multi-Task User Requests

To better reflect real-world scenarios where users often have complex, multi-faceted requirements, we constructed multi-task user requests by leveraging GPT-4o to combine single-task requests into comprehensive scenarios. The ground truth labels for these multi-task requests are directly derived from the corresponding Single-Task User Requests labels:

> **Prompt:** I would like to generate a multi-task user request. Below are several single-task user requests. Combine them into a natural, multi-task user request.
> 1. The order of the tasks is not fixed; you can arrange the tasks in a way that aligns with a human query. 2. The combined request must be coherent and meaningful, with each task relying on the results of the previous task(s). 3. If it is not possible to form a natural request, output 'NO'.
> Below are the user requests to combine:
> {user requests 1}-{task 1}, ...

The generated multi-task requests underwent rigorous review and refinement by domain experts to ensure quality and coherence. Through this process, we curated 13,383 multi-task user requests.

## C Annotation Protocol

This appendix provides detailed annotation guidelines, inter-annotator agreement analysis, and example annotations for our evaluation metrics.

### C.1 Workability Annotation Guidelines

A model is considered *workable* if it satisfies all of the following criteria:

1. **Task Type Match:** The model's task type (extracted from the `pipeline_tag` field in model metadata) matches the user request.
2. **Input/Output Compatibility:** The model accepts the required input format and produces the expected output format.
3. **Executability:** The model is available and executable on Hugging Face Hub or can be deployed locally. This is verified through:
   - API availability check
   - Checkpoint availability
   - No reported execution errors in model card

**Annotation Decision Tree:**

- If any criterion is not satisfied- Workability: 0
- If all criteria are satisfied- Workability: 1

### C.2 Reasonability Annotation Guidelines

Among workable models, a model is considered *reasonable* if it satisfies the following criteria:

1. **Domain-Relevant Training Data:** The model is trained on datasets relevant to the user's domain. This is verified through the `datasets` field in model cards.
2. **Competitive Performance:** The model achieves competitive performance within its domain, defined as ranking within the top-5 reported metrics among candidate models.

**Annotation Decision Tree:**

- If the model is not workable- Reasonability: 0
- If the model is workable but fails either criterion- Reasonability: 0
- If the model is workable and satisfies both criteria-Reasonability: 1

## D Additional Results

### D.1 Experimental setup

This section provides additional implementation details that complement the experimental configuration described in the main paper.

**LLM Configuration.** For the reasoning LLM component in our HuggingR$^4$ framework, we maintain default parameter settings including temperature, Top-K, and Top-P values to preserve the model's natural reasoning capabilities. In contrast, for all other LLM components involved in the pipeline, we set the temperature to 0 to ensure deterministic and consistent outputs across experimental runs.

**Vector Database and Retrieval Framework.** We implement our vector database using Chroma, which provides efficient storage and retrieval of model embeddings. The retrieval operations are orchestrated through the LangChain framework, enabling seamless integration between the reasoning components and the vector database queries.

These additional parameters, combined with the configuration details provided in Section 4.2 of the main paper (Multi-Query number $K = 4$, text-embedding-3-large embedding model, sliding window parameters $N = 3$ and $k = 5$, failure tracing threshold $\theta$=80%), constitute the complete experimental setup for our evaluation.

### D.2 Tool Selection vs. Model Selection

While both traditional API tool selection and community model selection involve choosing appropriate solutions for user tasks, they operate under fundamentally different paradigms. Traditional tool selection handles discrete, non-overlapping function categories (weather API and email API) with complete structured metadata, where selection reduces to categorical matching. In contrast, community model selection faces four critical challenges:

- **Massive capability overlap:** Thousands of models target identical tasks (e.g., 6,117 sentiment analysis models on HuggingFace), making differentiation challenging.
- **Subtle differentiation:** Models vary in domain adaptation, architecture, training data, and language support, requiring fine-grained comparative reasoning.
- **Continuous quality spectrum:** Models offer varying degrees of suitability rather than binary applicability.

These differences manifest across three critical dimensions. **First, search space characteristics.** Tool selection operates on 10 to 100 curated tools with minimal overlap, enabling exhaustive enumeration. Model selection navigates over 100,000 models with massive redundancy, requiring multi-stage progressive filtering. **Second, context dependency.** Tool performance is largely context-independent: a weather API returns accurate data regardless of application domain. In contrast, model performance is highly context-dependent. For instance, an object detector trained on everyday scenes (e.g., COCO) may achieve strong performance on common objects but struggle with domain-specific targets like satellite imagery or industrial defects, where specialized models demonstrate superior accuracy. **Third, selection criteria.** Tool selection uses binary applicability checks: Does the tool belong to the target category? Are required parameters available? This reduces to boolean logic. Model selection requires multi-dimensional ranking that balances task alignment, domain relevance, language compatibility, architectural suitability, and deployment constraints under incomplete information, necessitating context-aware scoring with uncertainty handling.

| Threshold $\theta$ | Workability (%) | Reasonability (%) |
|---|---|---|
| 0% | 88.09 | 77.66 |
| 40% | 90.45 | 80.41 |
| 60% | 91.73 | 81.69 |
| **80%** | **92.03** | **82.46** |
| 100% | 91.24 | 81.99 |

Table 8: Ablation study on the failure tracing threshold $\theta$ using HuggingR$^4$ with GPT-4o-mini and text-embedding-3-large.

| Module / Variant | Avg. Tokens per Query |
|---|---|
| Multi-Query Generation | 204 |
| Semantic Distillation | 1,204 |
| Metadata Stream | 361 |
| Self-reflection | 1,067 |
| **HuggingR$^4$ (Total Pipeline)** | **10,464** |

Table 9: Step-wise breakdown of average token consumption per query.

### D.3 Token Consumption Breakdown

Table 9 details our token consumption. While Distillation and Reflection require more tokens to process detailed capabilities, modules like Multi-Query Generation incur negligible overhead. Crucially, our sliding window limits full model card access, bounding the total usage and completely decoupling it from the repository size.

### D.4 Failure Tracing and Backtracking.

To evaluate the effectiveness of the failure tracing mechanism, we conduct ablation studies by varying the similarity threshold $\theta$ that triggers the traceback process. Table 8 presents the results across different threshold settings. We observe a consistent improvement trend as $\theta$ increases from 0% to 80%, with performance peaking at $\theta = 80\%$ (92.03% workability and 82.46% reasonability). Disabling the mechanism ($\theta = 0\%$) yields the lowest performance (88.09%/77.66%), while an overly strict threshold ($\theta = 100\%$) leads to slight degradation (91.24%/81.99%), indicating that moderate thresh-

| Method | Workability | Reasonability |
|---|---|---|
| Concatenation Retrieval | **84.77** | **75.56** |
| Separate Retrieval (Top-1) | 83.07 | 74.11 |
| Separate Retrieval (Top-3) | 83.66 | 73.72 |
| Separate Retrieval (Top-5) | 84.65 | 73.62 |

Table 10: Comparison of Retrieval Strategies on HuggingR$^{4*}$ with GPT-4o-mini and text-embedding-3-large.

olds best balance failure detection and false positive avoidance.

### D.5 Multi-Query Retrieval Strategy

As mentioned in the main text, we employ multiple queries from different perspectives using LLMs before performing retrieval to achieve better results. However, unlike previous methods, we do not perform separate retrievals for each generated query. Instead, we concatenate these queries and conduct a single retrieval operation to achieve cost savings.

As shown in Table 10, we conducted additional experiments to validate our approach. We compare our concatenation-based retrieval strategy against separate retrieval methods with different top-$k$ configurations using Reciprocal Rank Fusion (RRF) for result merging.

The results demonstrate that our concatenation approach achieves the best performance across both metrics. For the separate retrieval methods, we observe an interesting trade-off: as the top-$k$ value increases, workability improves while reasonability decreases. This phenomenon occurs because larger $k$ values expand the retrieval scope, capturing more potentially relevant models and improving workability. However, this broader scope introduces more noise and less precise candidates, which negatively impacts the subsequent fine-grained refinement process, leading to decreased reasonability scores.

### D.6 Adaptability to Other Repositories

Though evaluated on HuggingFace, HuggingR$^4$ readily adapts to unstructured platforms like GitHub. By integrating schema extraction into our offline Semantic Distillation step, the LLM can parse raw code or READMEs into standardized metadata (*e.g.*, tags) in a single pass. This one-time preprocessing strictly incurs zero online latency, seamlessly mapping unstructured data into our searchable space without compromising efficiency.

### D.7 Qualitative Results

To demonstrate the effectiveness of HuggingR$^4$ in real-world, we present three representative examples that showcase different aspects of our framework's capabilities.

**Example #1.** As shown in Table 12, HuggingR$^4$ demonstrates its ability to understand domain-specific requirements. The system effectively recognizes that the user needs a sentiment analysis model specifically optimized for social media content. Through the reasoning-retrieval phase, our framework identifies that tweet sentiment analysis requires models trained on informal language, hashtags, and social media conventions, which differ significantly from traditional text sentiment analysis. The retrieved candidate models are then refined based on their training data sources, with preference given to models that explicitly mention Twitter or social media datasets in their descriptions.

**Example #2.** As shown in Table 13, this case exemplifies HuggingR$^4$'s capability to recognize multi-modal requirements and domain specificity simultaneously. The system correctly identifies that the user requires a computer vision model capable of document layout analysis, specifically tailored for academic papers. Through progressive reasoning, our framework understands that academic documents have unique structural elements that require specialized recognition capabilities. The refinement phase successfully filters models based on their training on academic document datasets and their ability to distinguish between different layout components typical in research papers.

**Example #3.** As shown in Table 14, This example particularly highlights the power of our reflection mechanism. Initially, the first round of retrieval returned suboptimal results, focusing primarily on generic text-to-speech models without emotional expressiveness capabilities. However, our reflection module identified this mismatch between user requirements and retrieved candidates. Recognizing the need for models with emotional speech synthesis capabilities, the system performed a sliding window backtrack and initiated a second retrieval round with refined search criteria. In the second iteration, HuggingR$^4$ successfully identified models specifically designed for emotional speech generation, demonstrating how the reflection component enables self-correction through explicit error analysis and iterative improvement in model selection.

| Task | Args | #Candidate Models | #Single-Task Requests |
|---|---|---|---|
| text-classification | text | 30 | 31 |
| token-classification | text | 30 | 30 |
| text2text-generation | text | 30 | 30 |
| summarization | text | 30 | 32 |
| translation | text | 30 | 18 |
| conversational | text | 30 | 14 |
| text-generation | text | 30 | 11 |
| sentence-similarity | text | 30 | 31 |
| tabular-classification | text | 30 | 30 |
| object-detection | image | 30 | 42 |
| image-classification | image | 30 | 30 |
| image-to-text | image+text | 30 | 31 |
| text-to-image | text+image | 30 | 19 |
| image-to-image | image | 30 | 38 |
| visual-question-answering | image+text | 30 | 22 |
| document-question-answering | text | 30 | 30 |
| image-segmentation | image | 30 | 32 |
| depth-estimation | image | 30 | 12 |
| text-to-video | text+image | 30 | 28 |
| text-to-speech | text+audio | 30 | 30 |
| automatic-speech-recognition | audio | 30 | 34 |
| audio-to-audio | audio | 30 | 33 |
| audio-classification | audio | 30 | 36 |
| tabular-regression | text | 30 | 25 |
| time-series-forecasting | text | 30 | 30 |
| voice-activity-detection | audio | 30 | 18 |
| text-to-audio | text+audio | 30 | 20 |
| table-question-answering | text | 30 | 19 |
| question-answering | text | 30 | 30 |
| keypoint-detection | image | 30 | 24 |
| image-to-3d | image | 30 | 20 |
| text-to-3d | text-image | 30 | 22 |
| video-classification | video | 30 | 30 |
| visual-document-retrieval | image+text | 30 | 30 |
| audio-text-to-text | audio+text | 30 | 33 |
| image-text-to-text | image+text | 30 | 40 |
| video-text-to-text | video+text | 30 | 31 |

Table 11: Overview of the task list, including arguments, candidate model pool sizes, and the number of single-task requests for each task.

**Example #1**

**Question:**
Could you perform a sentiment analysis on the tweets provided in ./tweets.txt?

**Model Output by HuggingR$^4$**

To perform sentiment analysis on tweets, we need to select a model that is specifically designed for sentiment analysis, ideally trained or suitable for analyzing social media data like tweets. Let's go through the selection process step by step.
**Initial Similarity Retrieval**
Since the user wants to perform sentiment analysis on tweets, I will start with a similarity search query focusing on sentiment analysis related to social media or tweets.

Direct Retrieval

<|begin_similarity_query|> perform sentiment analysis on tweets <|end_similarity_query|>
<|begin_similarity_result|>
[finiteautomata/bertweet-base-sentiment-analysis, cardiffnlp/twitter-roberta-base-sentiment, cardiffnlp/twitter-roberta-base-sentiment-latest, j-hartmann/emotion-english-distilroberta-base, siebert/sentiment-roberta-large-english]
<|end_similarity_result|>

**Analyzing the Similarity Results**
The models returned from the Retrieval search are as follows:
1. finiteautomata/bertweet-base-sentiment-analysis
2. cardiffnlp/twitter-roberta-base-sentiment
3. cardiffnlp/twitter-roberta-base-sentiment-latest
4. j-hartmann/emotion-english-distilroberta-base
5. siebert/sentiment-roberta-large-english
These models are designed for sentiment analysis, and several of them are specifically trained on Twitter data, which is ideal for the user's requirement.
Next, let's check if these models are specifically trained on tweet datasets to filter for the best ones further.
**Dataset Filtering**
Considering the user's need for sentiment analysis on tweets, I'll filter for models that mention a relevant dataset related to social media or tweets.

Metadata Retrieval

<|begin_dataset_query|>
models designed for sentiment analysis on Twitter or social media datasets
<|end_dataset_query|>
<|begin_dataset_result|>
This retrieval is invalid. Please refer to other search results.
<|end_dataset_result|>

**Candidate Analysis**
Since the dataset filtering did not yield trustworthy results, I will proceed with analyzing the candidates based on their known capabilities and descriptions from the initial similarity results.
I have the following candidates from the similarity retrieval:
1. finiteautomata/bertweet-base-sentiment-analysis 2. cardiffnlp/twitter-roberta-base-sentiment‘ 3. cardiffnlp/twitter-roberta-base-sentiment-latest
I'll now get detailed descriptions of these models to see which one would best fit the user's need for analyzing the sentiment in tweets.

Refinement Tool

<|begin_descriptions_query|>
[finiteautomata/bertweet-base-sentiment-analysis, cardiffnlp/twitter-roberta-base-sentiment, cardiffnlp/twitter-roberta-base-sentiment-latest]
<|end_descriptions_query|>
<|begin_descriptions_result|>
[Model card 1: ..., model card 2: ..., model card 3: ...]
<|end_descriptions_result|>

All three models are capable of performing sentiment analysis on tweets. However, the ‘cardiffnlp/twitter-roberta-base-sentiment-latest‘ model stands out due to the following reasons:
- It is trained on a significantly larger dataset of tweets (124 million).
- It has more downloads and likes, indicating greater usage and community trust.
- It provides a comprehensive classification of sentiments.
After a module of Reflection, based on the analysis of the descriptions and the specific needs for sentiment analysis on tweets, I recommend:
\boxed{cardiffnlp/twitter-roberta-base-sentiment-latest}.

Table 12: An illustrative example demonstrating how our proposed HuggingR$^4$ framework processes a user request.

**Example #2**

**Question:**
I have an image of a research paper. Can you identify the layout elements such as images and text within it?

**Model Output by HuggingR$^4$**

To address the user's request for identifying layout elements such as images and text within an image of a research paper, we need to identify a model suitable for document layout analysis or optical character recognition (OCR) focused on document structures.
**Identifying the model requirements**
- Type of request: Layout element identification in an image of a research paper.
- Constraints: The model should be capable of handling layout elements like images and text.
**Using the Direct Retrieval tool**
Let's start by retrieval for models that are relevant to document layout analysis.

Direct Retrieval

`<|begin_similarity_query|>`
Identify layout elements such as images and text in an image of a research paper. `<|end_similarity_query|>`
`<|begin_similarity_result|>`
[omoured/YOLOv10-Document-Layout-Analysis, TahaDouaji/detr-doc-table-detection, keremberke/yolov8s-table-extraction, Aryn/deformable-detr-DocLayNet, keremberke/yolov8m-table-extraction]
`<|end_similarity_result|>`

**Dataset Filtering**
Next, we'll look to filter these models based on their suitability for datasets related to document layout analysis. This usually includes datasets with annotated layouts that include text and images.
Since we're targeting layout identification in research papers, I will describe the required dataset more specifically.

Metadata Retrieval

`<|begin_dataset_query|>`
Models trained on document layout datasets with elements such as text and images, specifically those suitable for research papers.
`<|end_dataset_query|>`
`<|begin_dataset_result|>`
[omoured/YOLOv10-Document-Layout-Analysis, TahaDouaji/detr-doc-table-detection, keremberke/yolov8s-table-extraction, Aryn/deformable-detr-DocLayNet, keremberke/yolov8m-table-extraction]
`<|end_dataset_result|>`

**Detailed Descriptions of the Candidate Models**
Since we still have more than three candidates, I will get detailed descriptions of these models to narrow it down further.

Refinement Tool

`<|begin_descriptions_query|>`
[omoured/YOLOv10-Document-Layout-Analysis, TahaDouaji/detr-doc-table-detection, keremberke/yolov8s-table-extraction]
`<|end_descriptions_query|>`
`<|begin_descriptions_result|>`
[Model card 1: ..., model card 2: ..., model card 3: ...]
`<|end_descriptions_result|>`

**Final Selection**
- omoured/YOLOv10-Document-Layout-Analysis is the most suitable model since it is specifically designed for document layout analysis, capable of detecting not only text but also images, making it well-suited to the user's needs.
- The other models focus primarily on table detection, which may not fully address the broader requirement of identifying various layout elements in a research paper.
**Self-Reflection**
The selected model, omoured/YOLOv10-Document-Layout-Analysis, meets all the user's requirements as it focuses on document layout analysis and is trained on a dataset relevant to research paper layouts.
After a module of self-reflection, here is my final answer:
`\boxed{omoured/YOLOv10-Document-Layout-Analysis}`.

Table 13: An illustrative example demonstrating how our proposed HuggingR$^4$ framework processes a user request.

**Example #3**

**Question:**
Could you help me generate a speech clone that conveys specific emotions from the text: ./emotion_text.txt?

**Model Output by HuggingR$^4$**

**Analyze the User's Request**
- The user wants to generate a speech clone capable of conveying specific emotions based on text found in ./emotion_text.txt. This indicates a requirement for a model that can perform emotion recognition or emotion synthesis in speech.
To find relevant models, I'll start with a similarity search related to generating speech that conveys emotions.

Direct Retrieval

<|begin_similarity_query|>Speech synthesis that conveys emotions. <|end_similarity_query|>
<|begin_similarity_result|>
[SWivid/F5-TTS, SparkAudio/Spark-TTS-0.5B, HKUSTAudio/Llasa-3B, coqui/XTTS-v2, metavoiceio/metavoice-1B-v0.1]
<|end_similarity_result|>

**Analyze Results**
The similarity retrieval returned 5 models. Now, I'll check if any of these models provide capabilities specific to synthesizing speech that conveys emotions.
**Filtering by Dataset**
Next, some of these models might have used datasets related to emotional speech synthesis. I'll filter based on that.

Metadata Retrieval

<|begin_dataset_query|>
Models that utilize datasets for emotional speech synthesis.
<|end_dataset_query|>
<|begin_dataset_result|>
This retrieval is invalid. Please refer to other search results.
<|end_dataset_result|>

The dataset filtering returned no valid models, indicating that I cannot trust this retrieval for emotional speech synthesis datasets. Therefore, I'll proceed directly with the candidate models obtained from the similarity search without additional filtering.
**Retrieve Descriptions of Candidate Models**
I will now get the detailed descriptions of the returned models to determine their suitability for generating emotional speech.

Refinement Tool

<|begin_descriptions_query|>
[SWivid/F5-TTS, SparkAudio/Spark-TTS-0.5B, HKUSTAudio/Llasa-3B]
<|end_descriptions_query|>
<|begin_descriptions_result|>
[Model card 1: ..., model card 2: ..., model card 3: ...]
<|end_descriptions_result|>

Given this analysis, I cannot confidently recommend a model that fully meets all requirements for generating emotionally expressive speech.
**Reflection**
Proceeding with the self-reflection on the models, since none explicitly state the ability to express specific emotions.
Now system have completed a full turn, The Reflection module believes that [SWivid/F5-TTS, SparkAudio/Spark-TTS-0.5B, HKUSTAudio/Llasa-3B] are not capable of doing this job. Please return to the thinking-retrieval stage and re-screen other models. The system has automatically updated your previous query results for you. Please go through the Step 1.

Direct Retrieval

<|begin_similarity_query|>Speech synthesis that conveys emotions. <|end_similarity_query|>
<|begin_similarity_result|>
[coqui/XTTS-v2, metavoiceio/metavoice-1B-v0.1, amphion/MaskGCT, lj1995/GPT-SoVITS, OuteAI/OuteTTS-0.1-350M]
<|end_similarity_result|>

Now, I will get detailed descriptions of these selected models for a final evaluation.

Refinement Tool

<|begin_descriptions_query|>
[coqui/XTTS-v2, metavoiceio/metavoice-1B-v0.1, lj1995/GPT-SoVITS]
<|end_descriptions_query|>
<|begin_descriptions_result|>
[Model card 1: ..., model card 2: ..., model card 3: ...]
<|end_descriptions_result|>

Considering the explicit emphasis on emotional synthesis and ease of use in various speech scenarios, I will select metavoiceio/metavoice-1B-v0.1 as the best fit due to its dedicated focus on emotional speech and strong specification in processing long-form text, which aligns well with the user's goal of generating expressive speech from a text file.
After a module of self-reflection, here is my final answer: \boxed{metavoiceio/metavoice-1B-v0.1}.

Table 14: An illustrative example demonstrating how our proposed HuggingR$^4$ framework processes a user request.