FROM AN LLM SWARM TO A PDDL-EMPOWERED HIVE: PLANNING SELF-EXECUTED INSTRUCTIONS IN A MULTI-MODAL JUNGLE

Anonymous authors
Paper under double-blind review

ABSTRACT

In response to the call for agent-based solutions that leverage the ever-increasing capabilities of the deep models' ecosystem, we introduce HIVE– a comprehensive solution for selecting appropriate models and subsequently planning a set of atomic actions to satisfy the end-users' instructions. HIVE operates over sets of models and, upon receiving natural language instructions (i.e. *user queries*), schedules and executes explainable plans of atomic actions. These actions can involve one or more of the available models to achieve the overall task, while respecting end-users specific constraints. Notably, HIVE handles tasks that involve multi-modal inputs and outputs, enabling it to handle complex, real-world queries. Our system is capable of planning complex chains of actions while guaranteeing explainability, using an LLM-based formal logic backbone empowered by PDDL operations. We introduce the MUSE benchmark in order to offer a comprehensive evaluation of the multi-modal capabilities of agent systems. Our findings show that our framework redefines the state-of-the-art for task selection, outperforming other competing systems that plan operations across multiple models while offering transparency guarantees while fully adhering to user constraints.

1 INTRODUCTION

Within the past few years, the number of available models –either through commercial paywalls or open-sourced– has exploded both in terms of intrinsic performances and in terms of tasks handled by them, ranging from text generation (Achiam et al., 2023; Anthropic, 2023; Team et al., 2023) to more specific actions such as code generation (Becker et al., 2023; Dong et al., 2024) or image generation (Wang et al., 2023b; Zhu et al., 2023). This rapid growth has unlocked unprecedented potential for real-world applications, inspiring practitioners, especially in industry, to envision new use cases that leverage these powerful models (Liu et al., 2023c; Shen et al., 2024; Lu et al., 2024; Xing et al., 2024). However, if creativity and possibility have been *unleashed* by such a surge, implementing pipelines that involve multiple models remains a complex and largely manual (and often cumbersome) process, particularly when addressing tasks beyond the original design of these models. This often leads developers to create ad hoc modules to manage these complexities. In addition, a significant number of models available in the wild are either advanced proof-of-concept or very specialised ones, see *e.g.* the hundreds of thousands of models available on the HuggingFace platform[1]. As a consequence of this abundance, navigating through this jungle to **select** the appropriate models for a set of tasks has become very challenging. This complexity arises both in terms of performance and compatibility. Connecting models' input and output formats is complex, as the generated results are often difficult to control (Scholak et al., 2021; Qin et al., 2022). Moreover, **planning** and chaining tasks for real-world use-cases present a significant challenge too.

In this study, we present a comprehensive solution to tackle the aforementioned two challenges, *i.e.* **(I)** selecting appropriate models and then **(II)** planning a set of atomic actions to achieve the objectives in the end-users' instructions (i.e., *user queries*.) Our system, **HIVE**, takes natural language instructions (potentially involving multi-modal inputs and outputs) and can effectively schedule, execute and explain plans composed of atomic actions. These plans may involve one or more mod-

[1] 1 016 247 models on `https://huggingface.co/models` as of October 1st, 2024.

els, carefully orchestrated to accomplish the overall task while adhering to users-specific constraints, such as model size or licensing, to name a few.

One of our key contributions addresses the first challenge: the lack of *machine-understandable* interface that consolidates comprehensive information about available models. To bridge this gap, we propose a Capability Knowledge Graph (C-KG) which encompasses all the dimensions needed to sort models along the many needed dimensions required for automated planning and execution. For each model, *inter alia*, C-KG captures critical details such as supported tasks, performance metrics from state-of-the-art benchmarks, and minimal code snippets for inference. Additionally, to enable the planning of complex action sequences with guaranteed explainability, we developed a novel planning approach which, instead of relying solely on LLM reasoning capabilities, also employs formal logic to reach its conclusions. To achieve this, we took advantage of PDDL – a formal language widely used in robotics for defining planning problems (Aeronautiques et al., 1998), mapping the end-users' instructions with atomic actions, thereby enabling the conversion of natural language instructions into a PDDL problem space. This approach allows us to formally plan before executing the tasks using code snippets from the C-KG. As a result, it has enabled us to generate detailed reports that provide end-users with fine-grained and reliable explanations.

In the absence of standard publicly available benchmarks for solving real-world tasks, we introduce MuSE, a new evaluation benchmark of complex queries involving multi-modal inputs and outputs, to assess our proposed framework. Using MuSE, we reviewed the closest existing solutions, namely HuggingGPT (Shen et al., 2024) and ControlLLM (Liu et al., 2023c), which only tackle sub-problems of our broader objectives. The results indicate that HIVE not only surpasses these competitors but also consistently outperforms them across all benchmark dimensions. HIVE demonstrates a 30% higher accuracy in task selection and respects user constraints in 100% of cases, while being more reliable.

2 PRELIMINARIES

Planning Domain Definition Language (PDDL) (Aeronautiques et al., 1998) is a standardised language extensively used in the field of artificial intelligence (AI) planning to represent planning domains and problems. PDDL provides a formal syntax and semantics for defining the components of a planning task, including actions, predicates, objects, and their relationships. It enables the clear specification of the initial state, goal conditions, and permissible actions within a domain, facilitating the development and comparison of planning algorithms. In our work, PDDL plays a critical role in task decomposition and planning. By defining tasks as actions within PDDL domains, we leverage established planning techniques to generate coherent and feasible plans. The use of PDDL allows us to formally model complex tasks, ensuring that the system can reason about the preconditions and effects of actions within a well-defined framework.

Let $\mathbb{D} = \left\{d_1, d_2, \ldots, d_{|D|}\right\}$ be a set of PDDL domains. Each PDDL domain $d_j \in \mathbb{D}$ is associated with a set of PDDL actions s.t. $\mathbf{a}^{d_j} = \left\{a_1^{d_j}, a_2^{d_j}, \ldots, a_{A^j}^{d_j}\right\}$, where $j \in [1, |D|]$ and $A^j \in \mathbb{N}$ the number of actions included within the PDDL domain d_j. Furthermore, let $\mathbb{T}$ be a set of different tasks, consisting of all PDDL actions across the available PDDL domains $\in \mathbb{D}$, as follows: $\mathbb{T} = \bigcup_{j=1}^{|D|} \mathbf{a}^{d_j}$. Finally, we define $\mathbb{M} = \left\{m_1, m_2, \ldots, m_{|M|}\right\}$ as the set of all models available for completing a set of different tasks $\mathbb{T}$ or combinations thereof.

3 HIVE — GENERAL ARCHITECTURE

3.1 CAPABILITY KNOWLEDGE GRAPH

We extract *model cards*[2] directly from HuggingFace and incorporate an OpenIE extraction route for converting the textual descriptions from each model card into a structured representation. We align the models with Papers With Code[3], which enables us to collect information about how a particular

[2] https://huggingface.co/docs/hub/en/model-cards
[3] https://paperswithcode.com/

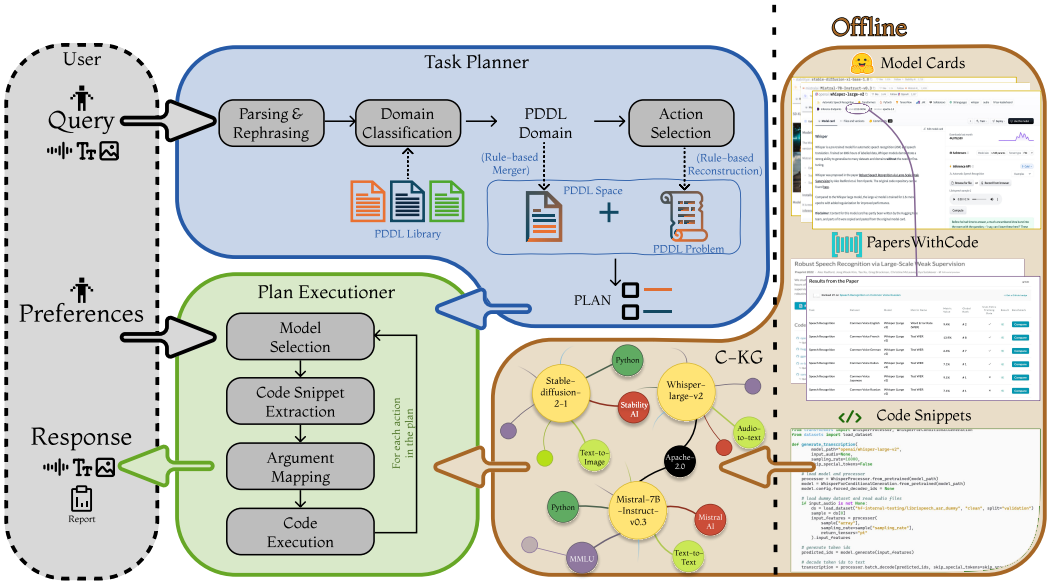

Figure 1: HIVE modular architecture.

model performs across different benchmarks. Our goal is to build a Capability Knowledge Graph, denoted as C-KG, which is a graph $G(\mathbb{M}, \mathbb{T}, E)$, such that each model $m_j \in \mathbb{M}$ is associated with one or more tasks from $\mathbb{T}$ along with its relevant performance, and different types of edges $\in E$ with various properties (e.g., number of parameters or supported languages). We search each model card collected by HuggingFace for potential arXiv[4] paper ID. We make use of the arXiv paper ID in order to bridge models from HuggingFace with their corresponding performances in different tasks and benchmarks. When we identify an arXiv paper ID within the knowledge base of Papers With Code, we extract all model versions[5] and their performances across the different benchmarks.

Each different model version is represented as a separate vertex within the C-KG. A model vertex $m_j \in$ C-KG is aligned with its model card *iff* the model's ID can be matched (e.g., `flan-t5-base`[6]) in both the original model card and the ID of at least one of the retrieved records from Paper With Code. This enables us to build a knowledge graph in which different models are associated with different benchmarks and tasks. Apart from the relevant performance scores in the various benchmarks and the models' properties extracted from HuggingFace, we extract from the model cards coding snippets that describe how each model can be loaded and executed. These execution snippets enable on-demand loading of a chosen model and execution of its inference step based on the provided parameters, if selected by model selection pipelines (see Section 3.2.2).

When processing the HuggingFace model card, we leverage a combination of keywords (see Appendix C) and regular expressions to identify coding blocks that showcase simplified examples of loading and executing a particular model m_j. After the coding blocks are extracted, we prompt (see Appendix C) an LLM that is proficient in code generation, such as DeepSeek-Coder[7], to generate a suitable Python function invoking m_j and running inference while taking into consideration input arguments of the originally extracted coding block. The resulting Python function, together with its signature (returned type, variable types and default values), is finally stored within the C-KG and connected with its corresponding model using an `execution` $\in E$ edge.

3.2 PLANNING MODEL ACTIONS

Having extracted and systematically structured the information pertinent to models associated with various tasks, we are now positioned to delineate the specific actions required to accomplish the objectives in the user query, as depicted in Figure 1.

[4] `https://arxiv.org/`

[5] arXiv papers may report results of different model variants, *e.g.* performance across different models sizes.

[6] `https://huggingface.co/google/flan-t5-base`

[7] `https://huggingface.co/deepseek-ai/DeepSeek-Coder-V2-Base`

3.2.1 TASK PLANNER

Parsing User Query User queries are often vague and unstructured, making it challenging for systems to understand the user's intent accurately. To overcome this, we introduce a parsing-rephrasing stage that bridges the gap between the ambiguous query and the system's structured requirements. This initial step sets the foundation for the subsequent stages, enabling the system to extract relevant information.

We parse an input user query $\mathbf{q}$ into distinct components: instruction ($\mathbf{i}$: `str`), input text ($\mathbf{t}$: `str`), question ($\mathbf{s}$: `str`), URL ($\mathbf{u}$: `str`), data ($\mathbf{x}$: `dict`), and categories ($\mathbf{g}$: `list`), as follows:

$$\mathcal{P}(\mathbf{q}) = \{\mathbf{i}, \mathbf{t}, \mathbf{s}, \mathbf{u}, \mathbf{x}, \mathbf{g}\} \tag{1}$$

with $\mathcal{P}$ the parsing function. The ability of LLMs to parse user queries into structured formats, like JSON, has been highlighted across multiple research efforts (Petroni et al., 2019; Wei et al., 2023). Using prompt engineering with a few-shot setting Brown (2020) (see Appendix C), we guide the LLM to convert an unstructured user input into structured data. Additionally, we ask the LLM to rewrite the user instruction to enhance clarity, simplifying complex directives and converting implicit information into explicit statements.

The instruction is crucial as it is used in the later stages to decompose the user query into smaller parts and determine objectives in the user query, $\mathbf{q}$. By transforming vague queries into well-defined components, our system becomes more robust to handle diverse and complex user inputs.

Task Decomposition Given the resulting instruction, after processing the input user query $\mathbf{q}$, we proceed to decompose it into smaller, manageable steps to identify a specific plan to attain the objectives (Wei et al., 2022; Yao et al., 2023b) and understand how each part of the instruction is associated with achievable goals within the system's capabilities. We utilise an LLM as a classifier (Zhang et al., 2024) in a few-shot example setting (Brown, 2020) (see Appendix C) to identify the relevant domains from the original set of PDDL domains, $\mathbb{D}$.

By providing the LLM with examples of instructions and their associated domains, we guide it to select the pertinent subset $\mathbb{D}^* \subseteq \mathbb{D}$ that aligns with the instruction. We prioritise *recall* in this classification step to ensure that all potentially relevant domains are considered, minimizing the risk of missing critical actions required to fulfil the user's objectives. This approach enhances the system's robustness by accounting for a wider range of possible actions.

Once we determine the subset of relevant PDDL domains, $\mathbb{D}^*$, we leverage the predefined PDDL structures of each classified domain. These domain structures are then mergedto create a unified PDDL domain file inclusive of all actions from $\mathbb{D}^*$, s.t.

$$\mathbf{a}^{\mathbb{D}^*} = \bigcup_{j=1}^{|D|} \mathbf{a}^{d_j} \;\middle|\; d_j \in \mathbb{D}^*. \tag{2}$$

This method ensures that the combined domain file encompasses all necessary actions while maintaining consistency and comprehensibility. Next, we exploit the parsed instruction $\mathbf{i}$ and the compiled set of actions from $\mathbb{D}^*$, $a^{\mathbb{D}^*}$, with an LLM to determine the specific actions required to achieve the instruction's objectives (see Appendix C), as follows: $\mathbf{a}_{\mathbf{i}}^{\mathbb{D}^*} \subseteq \mathbf{a}^{\mathbb{D}^*}$.

This allows to precisely map high-level user intents to concrete actions. Following this selection, the combined PDDL domain file (i.e. $\mathbf{a}^{\mathbb{D}^*}$) and the identified actions set (i.e. $\mathbf{a}_{\mathbf{i}}^{\mathbb{D}^*}$) help to reconstruct the corresponding PDDL problem. Finally, a Best First Width Search (Lipovetzky & Geffner, 2017) logical reasoner computes a detailed Plan of Actions ordering the actions $\in \mathbf{a}_{\mathbf{i}}^{\mathbb{D}^*}$ such that the system can execute step-by-step, ensuring coherent execution in line with the user's intent.

This hierarchical approach not only enhances the system's robustness in parsing and understanding diverse and complex user inputs but also guarantees the accuracy and feasibility of generated actionable plans by rigorously structuring them within established domain constraints.

3.2.2 MODEL SELECTION

Following the task planning, the next stage is the selection of appropriate models capable of performing the specified actions in the plan. Utilising the information from our C-KG (see Section 3.1), our goal is to identify a model combination $\mathbb{M}^\star \subseteq \mathbb{M}$ that would satisfy a set of conditions

$\mathbb{C} = \{c_1, c_2, \ldots, c_{|C|}\}$ imposed by the user while offering guarantees that the selected model combination will be suitable to generate an output $\mathbf{y}$ that addresses the original input query $\mathbf{q}$, s.t.

$$\mathbb{M}^\star = \arg\max_{\mathbb{M}} p\left(\mathbf{y}|\mathbf{q}, \mathbb{M}, \mathbb{C}\right) \tag{3}$$

For instance, if licensing is a concern, we filter out models that do not meet the required licensing terms. Similarly, if there are limitations on computational resources, we prioritise models that are efficient in size and cost. Additionally, we consider performance scores from relevant benchmarks to guide our selection. By analyzing these performance metrics, we can choose models that have demonstrated high effectiveness on tasks similar to those required, ensuring that the selected models are not only compliant with user constraints but also optimal for the specific tasks at hand.

By balancing these factors: model capabilities, licensing requirements, resource constraints, and performance metrics, we systematically select the most suitable model for each action. This ensures that the execution of the plan is aligned with both the technical requirements of the tasks and the practical considerations of the user. Ultimately, this careful selection enhances the system's efficiency and effectiveness, enabling it to perform complex tasks while adhering to user constraints.

3.2.3 Plan Execution

With the appropriate models selected for each action, we proceed to the plan execution phase. The execution involves retrieving the Python code snippets associated with `execute` relation with the chosen models, which are stored in the C-KG. More specifically, we map the arguments of the Python functions to the relevant components extracted from the parsed user query—$\{\mathbf{i}, \mathbf{t}, \mathbf{s}, \mathbf{u}, \mathbf{x}, \mathbf{g}\}$ (cf. Eq. 1) —using a complex similarity mapping algorithm. This mapping ensures that the models receive the correct inputs derived from the user's query, accurately capturing the user's intent. In cases where the Python code snippet for the selected model is not available, we employ a fallback strategy. We search for code snippets from other models that have been assigned to the same task and possess similar functionalities. This approach leverages the semantic and functional similarities between models within the same domain, allowing us to substitute models when necessary without compromising the action's intended outcome. By orchestrating the retrieval of code snippets and the fine-grained mapping of arguments, our system seamlessly transforms high-level plans into executable code, ensuring that the models operate on the intended data and parameters. This execution phase is critical as it bridges the gap between planning and action, which guarantees that each step of the plan is performed correctly and efficiently.

4 Experiments

4.1 Baselines

Addressing complex, multi-modal real-world tasks presents substantial challenges, and current solutions are limited. For our experiments, we compare our proposed method against the most relevant state-of-the-art techniques: HuggingGPT (Shen et al., 2024) and ControlLLM (Liu et al., 2023c). These represent significant advancements in integrating LLMs with task planning and execution frameworks.

We propose two innovative methods: HIVE and HIVE$_{\text{light}}$. HIVE leverages the advanced capabilities of ChatGPT for parsing user queries and decomposing tasks. To address the computational challenges associated with ChatGPT-based systems, we designed HIVE$_{\text{light}}$ as an efficient alternative that can be deployed on local servers. HIVE$_{\text{light}}$ employs InterLM2.5-7B-chat[8] Cai et al. (2024) for parsing user queries and Mistral-7B-Instruct-v0.3[9] for task decomposition, both of which have been subjected to 8-bit quantization. We selected a chat-oriented model for parsing, as conversational models excel at understanding subtle queries. Additionally, an instruction fine-tuned model was chosen for task decomposition due to its capability to deliver precise instruction clarity. By employing this dual-method setup, we ensure a thorough performance evaluation, positioning HIVE and HIVE$_{\text{light}}$ as strong competitors to existing state-of-the-art frameworks.

[8] https://huggingface.co/internlm/internlm2_5-7b-chat
[9] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

Table 1: Comparison of Task Selection, Flow of Tasks, and Output across all competitors.

Query Types	HuggingGPT			ControlLLM			HIVE light			HIVE		
	TS	FoT	O	TS	FoT	O	TS	FoT	O	TS	FoT	O
Single Task	0.47	0.47	0.83	0.74	0.74	0.74	0.80	0.80	0.72	0.88	0.88	0.79
Two Tasks	0.64	0.55	0.44	0.33	0.33	0.38	0.67	0.62	0.52	0.71	0.69	0.58
Three Tasks	0.42	0.42	0.30	0.36	0.36	0.33	0.57	0.43	0.33	0.67	0.67	0.46
Overall	0.57	0.51	0.53	0.43	0.43	0.47	0.69	0.64	0.55	**0.74**	**0.73**	**0.62**

4.2 MUSE — MULTI-MODAL SUB-TASK EXECUTION BENCHMARK

In the absence of standard publicly available benchmarks for solving real-world tasks, and recognizing that HuggingGPT (Shen et al., 2024) did not release their evaluation dataset, we developed a new benchmark to assess our proposed framework alongside state-of-the-art methods like HuggingGPT (Shen et al., 2024) and ControlLLM (Liu et al., 2023c). Although ControlLLM (Liu et al., 2023c) released their benchmark, it utilises a fine-tuned task decomposer, which could introduce bias if used for our evaluation. Therefore, to ensure a fair and unbiased comparison, we collaborated with experts from diverse linguistic backgrounds to create a set of 100 heterogeneous, real-world user queries (see Appendix Table 5) These queries are categorised into three types: Single-task, Two-task, and Three-task queries. In order to facilitate a fair comparison, we included only those tasks and models that are supported by all three systems. The benchmark is designed to cover various task domains, such as automatic speech recognition, question answering, and image generation, involving 15 models across different modalities[10]. This comprehensive benchmark enables us to rigorously evaluate the performance and generalisability of our framework.

Metrics. To assess our framework against state-of-the-art methods, we evaluate performance on three fronts, using binary metrics for simplicity and clarity:

- **Task Selection (TS)**: Determines whether the system accurately identifies the required tasks from the user's query. We assign a binary score—1 if the system selects all the tasks correctly, and 0 if it does not or if it selected irrelevant tasks. Correct task selection is crucial as it lays the foundation for successful execution and directly impacts the relevance of the final output.
- **Flow of Thought (FoT)**: We evaluate the logical sequence and integration of the selected tasks. A binary score is given based on whether the system establishes the correct flow—1 for a proper flow that respects task dependencies and order, and 0 for an incorrect sequence. This ensures that, especially in multi-task queries involving two or three tasks, the system processes tasks in an order that leads to the desired outcome.
- **Final Output (O)**: Assesses the correctness of the system's final response to the user's query. We adopt a binary evaluation—1 if the output fulfils the user's requirements, and 0 if it falls short. This includes evaluating the relevance of generated content, and the overall satisfaction of the user's intent. Note that we do not evaluate the *quality* of the output, that is we do not judge if the output is accurate but only focus on whether the expected task has been performed.

By employing these binary metrics for each query, we simplify the evaluation process while effectively capturing the essential aspects of each system's performance.

4.3 OVERALL RESULTS

The results of the experiment are presented in Table 1. The scores highlight the effectiveness of our proposed approach in handling complex multi-modal tasks. HIVE consistently outperforms the baseline methods in overall performance across all evaluation metrics: TS, FoT, and O. This performance is evident in both single-task and multi-task scenarios. Remarkably, HIVE light, which

[10]MUSE involves **10** domains leading to 10 distinct PDDL domain files covering **15** tasks. 67% of the queries are multi-modal `[text,image,audio]`, see also Appendix A and the `supplementary material`.

Table 2: Cross-Modality Performances.

In↓ Out→	Text		
	HuggingGPT	ControlLLM	HIVE light
Image	0.48	0.45	**0.52**
Audio	0.25	0.25	**0.67**

In↓ Out→	Image		
	HuggingGPT	ControlLLM	HIVE light
Text	0.36	0.18	**0.75**
Audio	0.50	0.25	**1.00**

In↓ Out→	Audio		
	HuggingGPT	ControlLLM	HIVE light
Text	0.33	0.14	**0.71**
Image	**0.80**	0.00	0.00

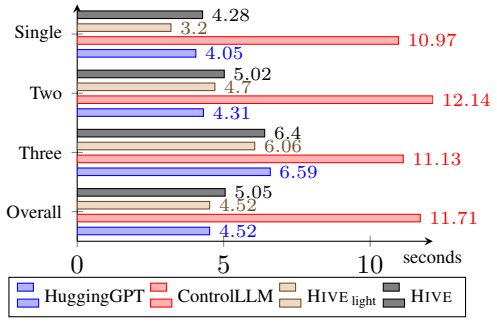

Figure 2: Average times (s) before execution.

is based on 8-bit quantised models with only 7 billion parameters, surpasses both HuggingGPT and ControlLLM in the overall evaluation. This underscores the effectiveness of our approach even with reduced computational resources.

In the case of single-task queries, all methods perform relatively well in generating correct final outputs. Though, HIVE stands out by achieving the highest performance in Task Selection and Flow of Thought, indicating a more accurate and coherent understanding and reasoning process. Conversely, HuggingGPT attains the highest Final Output performance but performs poorly in Task Selection and Flow of Thought metrics. This discrepancy arises because HuggingGPT tends to collect as many relevant tasks as possible, even if a query requires only a single task, leading to over-selection. Despite this over-selection, its strong ChatGPT backbone enables it to produce high-quality final outputs. Nevertheless, this approach may reduce the trustworthiness of the results, as it does not align precisely with the intended tasks (further discussed in Section 4.4.2). When it comes to multi-task queries, as expected, HIVE distinctly outperforms its competitors across all metrics. PDDL planning in HIVE light allows it to surpass HuggingGPT and ControlLLM, demonstrating proficiency in task selection, task flow management and execution. In contrast, ControlLLM struggles considerably with multi-task queries, exhibiting the weakest performance among the evaluated methods. While HIVE, HIVE light, and HuggingGPT employ prompt-based strategies to provide flexibility and adaptability, ControlLLM relies on a fine-tuned task decomposer. This approach limits its capacity to generalise to queries that deviate even slightly from its training set, leading to significant performance declines in multi-task scenarios.

4.4 DISCUSSIONS

4.4.1 CROSS-MODALITY PERFORMANCES

To gain a better understanding of the multi-modality of these systems' capabilities, we dig deeper into the Final Output (O) results from Table 1. We divide this investigation into three distinct parts based on the output modality and analyze the performance when the other two modalities are involved in the input (Table 2).

When it comes to text output, HIVE light demonstrates a substantial lead over its competitors when the input includes any image or audio. This showcases HIVE light's ability to integrate visual and auditory data to enhance text outputs. In the context of image output, HIVE light once again outperforms the other systems, illustrating its proficiency in converting textual and audio inputs into coherent visual responses. Lastly, although our system shows commendable performance in text-based audio generation, it falls short of achieving the desired objective in the image-to-audio scenario, indicating an area for potential improvement in future iterations.

4.4.2 TRUSTWORTHINESS

In order to review the connection between *justifications* (*i.e.* the conjunction[11] of TS and FoT) and *outputs* (O), we group in Figure 3 the results based on (justification, output) scores which can

[11] $\top$ iff TS AND FoT are both 1; $\bot$ in all other cases.

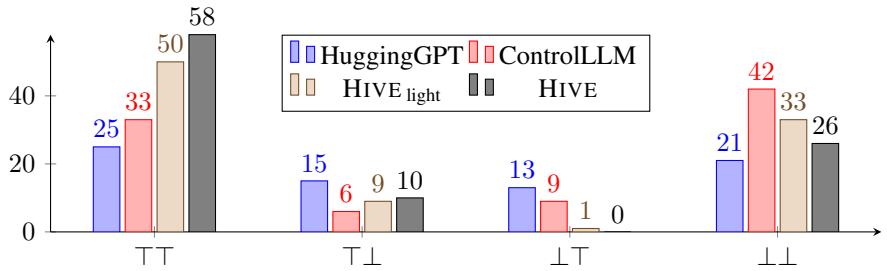

Figure 3: Correlations between *justifications* (TS AND FoT) and *outputs* (O).

respectively be correct ⊤ or incorrect ⊥. The failure cases `Err` are further discussed in Section 4.4.3. There are thereby four distinct cases. First, ⊤⊤ which corresponds to fully correct cases having both justifications and outputs, in this HIVE solutions (50+) outperform both HuggingGPT (25) and ControlLLM (33). Then, ⊤⊥ means that the plans were correct but the execution did not go through. In this category the four reviewed systems perform similarly, ranging from 6 to 15 cases. On the right-end side of Figure 3, the plain-wrong case ⊥⊥ witnesses ControlLLM as the "worst" system (see results in Section 4.3), having the highest score. Finally, the critical ⊥⊤ case indicates a lack of trustworthiness across all the baseline systems that may result in misleading outcomes. This case involves an incorrect plan or justification, despite the output being correct. Such cases have been recorded 9 times for ControlLLM and 13 times for HuggingGPT while being absent from HIVE and singleton in HIVE $_{\text{light}}$ showcasing the reliability of the results produced by HIVE.

Overall, this discussion allowed us to highlight two aspects. One, unlike other solutions, HuggingGPT exhibits results ranging from 13 to 25 in the four categories, meaning that it is hard to rely on it. Second, HIVE (s) tend not to fall in incoherent cases where results are correct without the corresponding plan being correct—in other words, when results are good their explanations can be trusted as well.

4.4.3 ROBUSTNESS

Since most of MUSE's queries require multiple models to interact together in a compatible manner, we noticed that sometimes the tested systems fail at dealing with either the *justification* or the *output* parts. In Table 3, we list the different cases encountered. The first point to be highlighted is that, among the four systems, HuggingGPT is the less robust one by far: 22 `Err` against 10, 7 and 6 for the other solutions. Second, even more critical, is that HuggingGPT, unlike the competition, is able to generate correct results (⊤) while failing (`Err`) in its plan construction. This exacerbates the fact that its justifications cannot be trusted, as the executed actions tend in many cases not to be consistent with the compiled plan, using GPT-3.5 at most places. This last finding is coherent with the ⊥⊤ discussion in Section 4.4.2.

4.4.4 LATENCY FOR PLANNING

Lastly, in this *discussion* section, we analyse the time performances (in seconds) of the systems to come up with a plan and select suitable models. As the chosen models may differ and no enforced rules such as "the quicker the better" (see Section 4.5 for discussion about model selection capabilities) were added, we measure the latencies up to the model selection stage.

Figure 2 presents these latencies according to the split already presented in Section 4.3 as per the number of tasks involved in the queries. First of all, HuggingGPT and HIVE (s) share the same orders of magnitude whereas ControlLLM is a magnitude slower (always 10+ seconds). Next, as expected, the more tasks within the query the slower the systems become until they reach an execution plan. On this, it is interesting to note that HuggingGPT's scaling law does not seem "linear" as the slope increases greatly between two- and three-task queries. This behaviour is compatible with the internal implementation of HuggingGPT: when other systems are rule-based (see the C-KG for HIVE (s) to select models), HuggingGPT needs to prompt (together with model descriptions) to select which models to use for each task.

Table 3: Failing case enumeration (`Err`), either as *justifications* (TS AND FoT) or *outputs* (O).

Failure Type	HuggingGPT	ControlLLM	HIVE light	HIVE
`Err`, $\top$	8	0	0	0
$\top$, `Err`	3	1	3	3
$\bot$, `Err`	3	1	1	0
`Err`, `Err`	8	8	3	3
Overall	22	10	7	6

4.5 TAKING INTO ACCOUNT USERS' CONSTRAINTS IN TERMS OF MODEL SELECTION

As depicted in Section 3.2.2, once a plan of actions is established, HIVE selects the *best* models to realise them. Obviously, depending on the circumstances, the definition of what is "best" may vary a lot, *e.g.* when resources are sparse, one may decide to use the smallest models possible even if the resulting quality is reduced, alternatively users might choose to select models based on their respective (recorded) results for specific benchmarks. In order to respect these various cases, HIVE allows users to specify selection criteria. In this Section[12], we review the capabilities of HIVE against HuggingGPT when users want to force some conditions of their own in the model selection. Since ControlLLM has one-to-one mappings of models for each task, it is *de facto* excluded.

Practically, we use the following query[13]: "`Transcribe the audio from .audio_1.wav and find entity tokens`". Regarding the task-model mappings, we let both HIVE and HuggingGPT have access to: openai/whisper-large-v2 and nvidia/parakeet-rnnt-1.1b (having respectively Apache-2.0 and CC-By-4.0 for licenses) for the ASR; and to dslim/bert-base-NER (MIT license) for NER. We first run the query without any constraints (control run, see Appendix B): both systems, HIVE and HuggingGPT, were able to transcribe the audio file and perform NER (even though HuggingGPT result set was empty). We then applied the following model selection constraints sequentially:

1. **License restrictions**: only use Openrail++ and Deepseek — HIVE returned nothing which was the expected behaviour as the available models were not having the requested licenses; on the other hand, HuggingGPT performed the task as in the control therefore infringing the restrictions (see Appendix B).
2. Uses the "**smallest possible**" model[14] — HIVE complied with the user choice and used the smaller models whereas HuggingGPT kept using openai/whisper-large-v2 as in the control run (see Appendix B).
3. Filter for the model having the **best results** at the `speech recognition on common voice english`[15] benchmark — Using the benchmark records from the C-KG, HIVE was able to select the correct model unlike HuggingGPT which chose models like in the control run (see Appendix Table B).

Overall, HIVE answered each time while properly taking into account the given constraints. While HuggingGPT failed every time, misleading even the users with regards to its justifications (refer to Section 4.4.2 for further justifications on this).

5 RELATED WORK

Automated Planning. The cognitive ability to organize and coordinate actions toward a specific goal is referred to as *planning*. While humans innately possess this capacity, machines lack such a capability. Automated planning has garnered significant interest from researchers across various domains, including robotics (Guo et al., 2023), autonomous vehicles (Ángel Madridano et al., 2021), and dialogue systems (Wang et al., 2023a). The methodologies employed to devise sequences of actions have evolved considerably, particularly in light of recent breakthroughs in deep learning.

[12] See also Appendix B for an extensive result description.
[13] A multimodal one involving two tasks: ASR and NER.
[14] We use the model disk footprint as a proxy for its size.
[15] Introduced in "Common Voice: A Massively-Multilingual Speech Corpus" Ardila et al. (2019).

Before the advent of large language models (LLMs), planning frameworks such as STRIPS (Fikes & Nilsson, 1971) or HTN (Erol et al., 1994) were developed to decompose tasks into a series of actions (or sub-tasks) leading to the desired outcomes (Sacerdoti, 1975). Building upon these frameworks, the Planning Domain Definition Language (PDDL) (Aeronautiques et al., 1998) emerged as a widely adopted standardized language for defining planning problems and domains. However, LLMs have superseded those frameworks to stand as a planner on their own (i.e. the *LLM-as-planner* paradigm). Multiple prompt engineering techniques (Liu et al., 2023b; Graux et al., 2024) were designed to leverage in-context learning aiming to directly generate the multi-step problem solutions. More specifically, the Chain-of-Thought (Wei et al., 2022) has revealed the promising reasoning capabilities of LLMs, and therefore new techniques were fashioned such as the self-consistency decoding strategy (Wang et al., 2022), Tree-of-Thought (Yao et al., 2023a), Program-of-Thought (Chen et al., 2023) or Graph-of-Thought (Yao et al., 2023c; Besta et al., 2024). However, LLMs are still struggling to produce acceptable and logical plans, especially as the complexity of the problem increases(Valmeekam et al., 2023; Xiao et al., 2024; Zheng et al., 2024) Thus, numerous initiatives have therefore sought to integrate problem-specific languages like PDDL along LLMs to maximize their effectiveness and leverage their full potential (Pallagani et al., 2023; Liu et al., 2023a; Oswald et al., 2024).

LLM-as-Agent. The genesis of large language models (LLMs) primarily stemmed from textual content, which initially narrowed the research focus to text generation. However, to address the diversity of real-world scenarios, significant efforts have been directed toward developing vision or speech LLMs, thereby aligning with a multi-modal paradigm (Zhu et al., 2023; Wu et al., 2023; Wang et al., 2023b). exemplify this trend. Additionally, to expand the capabilities of LLMs, there has been an increasing trend to integrate external tools with LLMs. Toolformer (Schick et al., 2024) pioneered the invocation of tool calls within generated sequences via special tokens giving rise to tool-augmented LLMs (Qin et al., 2023a;b; Guo et al., 2024; Qu et al., 2024). Then, ReAct (Yao et al., 2023b) introduced such intermediate tool calls during the reasoning process by incorporating intermediate outcomes within the prompt to better guide the final resolution of the problem. In contrast to ReAct, Reflexion (Shinn et al., 2023) adds verbal feedback on those intermediate results to further assess and verify outcomes, In the meantime, a plethora of fine-tuned LLMs tailored for specific tasks has become ubiquitous on platforms such as Hugging Face Hub (Wolf et al., 2019), alongside proprietary models such as GPT-4 (Achiam et al., 2023), Claude (Anthropic, 2023), and Gemini (Team et al., 2023) offering the opportunities to consider these LLMs as distinct agents. Indeed, the gathering of technical details for each parametric model stands as a critical component in the reporting and tracking efforts underlined by the use of *Model Cards* Mitchell et al. (2019). HuggingGPT (Shen et al., 2024), leverages such a large pool of LLMs using ChatGPT as the core controller. Following a similar approach, ControlLLM (Liu et al., 2023c) and Chameleon Lu et al. (2024) explore task planning via prompt engineering and integrate a more diverse pool of tools. While HuggingGPT, ControlLLM or Chameleon appoint appropriate models for each subtask, however, their model selection process remains sub-optimal as they do not identify the most accurate model. Thus, if these frameworks can fulfil their plans, the resulting performance may be unsatisfactory if the best agent is not utilized. To the best of our knowledge, our work represents the first attempt to address this gap.

6 Conclusion

Our research introduces HIVE, an innovative and comprehensive solution designed to navigate the complexities of model selection and task planning using a diverse set of deep learning models. By leveraging a Capability Knowledge Graph and an LLM-based formal logic planner, we transcend the limitations of the existing systems. HIVE stands out for its capability to plan and explain complex action chains while respecting user-specific constraints –thereby achieving both high performance and full transparency. Empirical evaluations on our newly designed benchmark reveal HIVE's superior performance, consistently outperforming competing platforms like HuggingGPT and ControlLLM. This breakthrough underscores HIVE's potential to redefine the state-of-the-art in task selection and planning, ultimately facilitating more efficient and user-friendly applications of advanced deep models. HIVE thus advances the handling of multi-modal tasks.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

Anthropic. Claude (oct 8 version). Accessed: 2023-10-08, 2023. URL https://www.anthropic.com/. Large language model.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. SpeechT5: Unified-modal encoder-decoder pre-training for spoken language processing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5723–5738, May 2022.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, pp. 500–506, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394314. doi: 10.1145/3545945.3569759. URL https://doi.org/10.1145/3545945.3569759.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. Internlm2 technical report, 2024.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020. URL https://arxiv.org/abs/2005.12872.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via chatgpt. *ACM Trans. Softw. Eng. Methodol.*, 33(7), September 2024. ISSN 1049-331X. doi: 10.1145/3672459. URL https://doi.org/10.1145/3672459.

Kutluhan Erol, James A Hendler, and Dana S Nau. *Semantics for hierarchical task-network planning*. Citeseer, 1994.

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

Damien Graux, Sébastien Montella, Hajira Jabeen, Claire Gardent, and Jeff Z Pan. [prompteng] first international workshop on prompt engineering for pre-trained language models. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 1311–1312, 2024.

Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. Recent trends in task and motion planning for robotics: A survey. *ACM Comput. Surv.*, 55(13s), jul 2023. ISSN 0360-0300. doi: 10.1145/3583136. URL https://doi.org/10.1145/3583136.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019a. URL https://arxiv.org/abs/1910.13461.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019b. URL http://arxiv.org/abs/1910.13461.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022a. URL https://arxiv.org/abs/2201.12086.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022b. URL https://arxiv.org/abs/2201.12086.

Nir Lipovetzky and Hector Geffner. Best-first width search: Exploration and exploitation in classical planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. doi: 10.1609/aaai.v31i1.11027. URL https://ojs.aaai.org/index.php/AAAI/article/view/11027.

B. Liu, Yuqian Jiang, Xiaohan Zhang, Qian Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *ArXiv*, abs/2304.11477, 2023a. URL https://api.semanticscholar.org/CorpusID:258298051.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), jan 2023b. ISSN 0360-0300. doi: 10.1145/3560815. URL https://doi.org/10.1145/3560815.

Zhaoyang Liu, Zeqiang Lai, Zhangwei Gao, Erfei Cui, Zhiheng Li, Xizhou Zhu, Lewei Lu, Qifeng Chen, Yu Qiao, Jifeng Dai, and Wenhai Wang. Controllm: Augment language models with tools by searching on graphs. *arXiv preprint arXiv:2305.10601*, 2023c.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 220–229, 2019.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34(1):423–431, May 2024. doi: 10.1609/icaps.v34i1.31502. URL https://ojs.aaai.org/index.php/ICAPS/article/view/31502.

Vishal Pallagani, Bharath Muppasani, Biplav Srivastava, Francesca Rossi, Lior Horesh, Keerthiram Murugesan, Andrea Loreggia, Francesco Fabiano, Rony Joseph, and Yathin Kethepalli. Plansformer tool: Demonstrating generation of symbolic plans using transformers. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 7158–7162. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/839. URL https://doi.org/10.24963/ijcai.2023/839. Demo Track.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 9538–9551. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3e25d1aff47964c8409fd5c8dc0438d7-Paper-Conference.pdf.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models, 2023a.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023b.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey. *arXiv preprint arXiv:2405.17935*, 2024.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021. URL https://arxiv.org/abs/2103.13413.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Earl D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, pp. 206–214, San Francisco, CA, USA, 1975. Morgan Kaufmann Publishers Inc.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC^2 Workshop*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 38975–38987. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7a92bcdede88c7afd108072faf5485c8-Paper-Datasets_and_Benchmarks.pdf.

Hongru Wang, Minda Hu, Yang Deng, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, Wai-Chung Kwan, Irwin King, and Kam-Fai Wong. Large language models as source planner for personalized knowledge-grounded dialogues. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9556–9569, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.641. URL https://aclanthology.org/2023.findings-emnlp.641.

Xinyu Wang, Bohan Zhuang, and Qi Wu. Switchgpt: Adapting large language models for non-text outputs. *arXiv preprint arXiv:2309.07623*, 2023b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*, 2023.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arxiv. *arXiv preprint arXiv:1910.03771*, 2019.

Chenfei Wu, Sheng-Kai Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *ArXiv*, abs/2303.04671, 2023. URL https://api.semanticscholar.org/CorpusID:257404891.

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. *arXiv preprint arXiv:2406.14884*, 2024.

Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. Understanding the weakness of large language model agents within a complex android environment. *arXiv preprint arXiv:2402.06596*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*, 2023c.

Yazhou Zhang, Mengyao Wang, Chenyu Ren, Qiuchi Li, Prayag Tiwari, Benyou Wang, and Jing Qin. Pushing the limit of llm capacity for text classification. *arXiv preprint arXiv:2402.07470*, 2024.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9006–9018, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.615.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Ángel Madridano, Abdulla Al-Kaff, David Martín, and Arturo de la Escalera. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 173: 114660, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.114660. URL https://www.sciencedirect.com/science/article/pii/S0957417421001019.

A MuSE Experimental Setup

Table 4 provides a comprehensive overview of the domains and tasks encompassed within the MuSE benchmark. All three competing systems have access to the models associated with each task listed in Table 4. It should be noted that both HuggingGPT and ControlLLM utilise ChatGPT as their backbone model, leveraging it for the execution of certain tasks. Since ControlLLM permits only one-to-one mappings, to maintain an unbiased benchmark, we assign one model per task (for details, please refer to the Supplementary Material). Moreover, to preserve the naturalness of the queries, we have refrained from making any grammatical or spelling corrections in the dataset.

Table 4: AI Tasks and Associated Models.

Domain	Task	Model
Audio	Automatic Speech Recognition	openai/whisper-large-v2 (Radford et al., 2022)
	Text to Speech	microsoft/speecht5_tts (Ao et al., 2022)
Image Generation	Text to Image	stabilityai/stable-diffusion-2-1(Rombach et al., 2022)
Image to Text	Image Captioning	Salesforce/blip-image-captioning-base(Li et al., 2022b)
	Object Detection	facebook/detr-resnet-101(Carion et al., 2020)
	Visual Question Answering	Salesforce/blip-vqa-base(Li et al., 2022a)
Image to Image	Depth Estimation	Intel/dpt-hybrid-midas(Ranftl et al., 2021)
Machine Translation	Translation from xx to yy	mistralai/Mistral-7B-Instruct-v0.1(Jiang et al., 2023)
Question Answering	Answer based on Context	distilbert/distilbert-base-uncased-distilled-squad(Sanh et al., 2019)
	Open QA	mistralai/Mistral-7B-Instruct-v0.1(Jiang et al., 2023)
	Table QA	Yale-LILY/reastap-large(Zhao et al., 2022)
Summarisation	Abstractive Summarisation	facebook/bart-large-cnn(Lewis et al., 2019b)
Text Generation	Text Generation	mistralai/Mistral-7B-Instruct-v0.1(Jiang et al., 2023)
Text Classification	Text Classification	facebook/bart-large-mnli (Lewis et al., 2019a)
Token Classification	Named Entity Recognition	dslim/bert-base-NER(Devlin et al., 2018)

Table 5: Small sample of queries from the MuSE benchmark.

Domain	Queries
"object_detection"	How many people is in the image? ./data/images/image_6.jpg
"image_to_text", "text_generation"	Write me a sonnet about this image ./data/images/image_11.jpg
"image_to_text", "image_generation"	Build a pictorial presentation of the flower after deciphering the audio ./data/audios/audio_4.wav
"audio", "question_answering"	Use audio transcription as context and answer 'what is the country starting the war whom the associated speech is about?' from ./data/audios/audio_6.wav
"audio", "summarisation", "image_generation"	Extract a summary from the audio transcript ./data/audios/audio_8.wav, and generate an image based on the summary

B SCENARIOS DETAILS FOR MODEL SELECTION VALIDATION

⇊ No Constraints (Cold run) ⇊

HIVE light

We, the people of the United States, in order to form a more perfect Union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

```
{
    "LOC": [
        "United␣States",
        "United␣States␣of␣America"
    ],
    "MISC": [
        "Constitution"
    ]
}
```

Comment: *Successfull!*

HuggingGPT

Based on the inference results, I have transcribed the audio file ".[Audio]" and obtained the following text: "We, the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves United States of America."
For the named entity recognition task, I have used the model "dslim/bert-base-NER." However, the inference result for this task is empty, indicating that no named entity tokens were found in the transcribed text.
Here is the complete workflow for your request:

1. **Automatic Speech Recognition (ASR) Task:**
   Model used: "openai/whisper-large-v2"
   Inference result: The audio file".[Audio]" was transcribed, and the resulting text is the one mentioned above.

2. **Named Entity Recognition (NER) Task:**
   Model used: "dslim/bert-base-NER"
   Inference result: No named entity tokens were found in the transcribed text.

Please note that the inference results are based on the models used, and there is a possibility of errors or incomplete detection.

Comment: *Unsuccessful: the system failed to pass the output from the previous task as input to the subsequent one, preventing the NER model from identifying any entities*

⇊ Specifically asked to use the smallest model available for each task ⇊

HIVE light

We, the people of the United States, in order to form a more perfect Union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America.

```
{
    "LOC": [
        "United␣States",
        "United␣States␣of␣America"
    ],
    "MISC": [
        "Constitution"
    ]
}
```

Comment: *Successful!*

HuggingGPT

Based on the inference results, I have transcribed the audio file ".[Audio]" and found the following named entity tokens:
"We the people of Unites States" – ORGANISATION
"the United States of America" – LOCATION
To accomplish this, I followed the following workflow:

1. **Task 1: Audio-to-Audio Transcription Model:** No suitable model found for this task
   Inference Result Error: "audio-to-audio" not found in available tasks

2. **Task 2: Automatic Speech Recognition (ASR) Task:**
   Model: I used the "openai/whisper-large-v2" model
   Inference result: "We, the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves United States of America."

3. **Task 3: Named Entity Recognition (NER) Task:**
   Model: I used the "dslim/bert-base-NER" model
   Inference result: No named entity tokens were found in the transcribed text.

I apologize for not being able to provide any named entity tokens in the transcribed text. If you have any further questions or need assistance with anything else, please let me know!

Comment: *Unsuccessful: the system failed in task selection and selected "audio-to-audio" as the first task, which is irrelevant to the user query. The system also failed to identify any entities*

⇓⇓ Specifically ask the systems to use `openrail++` & `deepseek` licensed models ⇓⇓

HIVE light

Comment: *Successful: as there were no models with `openrail++` and `deepseek` for ASR or NER, no model should be selected*

HuggingGPT

I apologize for any confusion caused. Based on the inference results, here is the detailed workflow for your request:

1. **Automatic Speech Recognition (ASR) Task:**

   Task: Transcribing the audio file "[Audio]".
   Model used: "openai/whisper-large-v2"
   Inference result: The transcribed text is as follows: "We, the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America."

2. **Named Entity Recognition (NER) Task:**

   Task: Finding named entity tokens in the transcribed text.
   Model used: "dslim/bert-base-NER".
   Inference result: Unfortunately, there is no available inference result for this task.

3. **Zero-shot Classification Task: Task:** Classifying named entity tokens.
   Model used: "gpt-3.5-turbo-0613"
   Inference result: I apologize, but as an AI language model, I am unable to access or transcribe audio directly. However, once the audio is transcribed, I can assist you with the zero-shot classification task. Could you please provide the transcription of the audio file, and I'll be happy to help you with the classification task using the provided text and image information?

4. **Text-to-Image Task: Task:** Generating images based on named entity tokens.
   Model used: "stability/stable-diffusion-v1"
   Inference result: Unfortunately, there was an error in the inference process for this task.

Please note that the inference results are based on the available models and their capabilities. If there are any specific requirements or modifications you would like to make, please let me know.

Comment: *Unsuccessful: first the system performed an incorrect selections of tasks and then also misguided the user by selecting models that do not aligned with the users license contraints*

⇓⇓ Specifically asked to choose the Automatic speech recognition model based on best performance on `speech recognition on common voice english` ⇓⇓

HIVE light

we the people of the united states in order to form a more perfect union establish justice insure domestic tranquillity provide for the common defense promote the general welfare and secure the blessings of liberty to ourselves and our posterity do ordain and establish this constitution for the united states of america

```
{
    "LOC": [
        "united␣States",
        "united␣States␣of␣
            ↪ america"
    ],
    "MISC": [
        "constitution"
    ]
}
```

Comment: *Successful: to test the model selection on benchmark performances, we added a "parakeet-rnnt" model in the mix as on this specific benchmark it performs better than the "whisper-large-v2".* HIVE$_{light}$ *successfully selects this for the ASR task*

HuggingGPT

Based on the Inference results, I will directly answer your request and provide a detailed workflow for your specific task. Here is the complete path or URLs of the files in the inference results:

1. **Transcribing the audio file "[Audio]" Model used:** "openai/whisper-large-v2"
   Inference Result: The transcribed text is as follows: "We, the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of liberty to ourselves United States of America."

2. **Finding named entity tokens in the transcribed text:**
   Model used: "dslim/bert-base-NER"
   Inference result: No named entity tokens were found in the transcribed text.

3. **Zero-shot classification for the task of "named entity tokens":**
   Model used: "gpt-3.5-turbo-0613"
   Inference result: As an AI language model, I don't have direct access to external files or the ability to process audio. Therefore, I am unable to perform zero-shot classification on the named entity tokens.

4. **Generating a response for the task of selecting models based on scores from "speech-recognition-on-common-voice-english":**
   Model used: "gpt-3.5-turbo-0613"
   Inference result: Based on the input the following response is generated:
   "To select models for speech recognition on Common Voice English, you can use the scores obtained from the speech recognition systems trained on the Common Voice English dataset. By evaluating the performance of different models using these scores, you can determine which models are most suitable for the task. This allows you to choose the best models for your specific speech recognition needs. Factors such as accuracy, robustness and language coverage should be considered when selecting these models. THe scores provide a valuable metric for comparing and evaluating different models, ultimately assisting in the decision-making process."

Please not that the information provided is based on the inference results from the models used. If you have any specific concerns or further questions, feel free to let me know!

Comment: *Unsuccessful: the system fails at the task selection state while also ignoring the requirements to selected model for the ASR based on the benchmark performance*

C HIVE'S PROMPTS

Usage Snippet Extraction **(§3.1)**

You are a python programming expert, mainly used to convert python code snippets into python functions. Follow the rules:
1. You also make sure all the function variables have default value.
2. There should always be input variable with default value that takes the input for the model.
3. Take Model path as a variable with default value of model name
4. Return the model response in the python function

Transform the code snippets in the text into one signed Python function including all the potential variables and default values for all. Only respond with code and in markdown format ‘‘‘python‘‘‘. {code}

Parsing & Rephrasing **(§3.2)**

Task Decomposition stage: The AI assistant can parse user input into multiple inputs and fill the relevant keys in the following `JSON`.

```
{"instruction": None, "input_text": None, "question": None,
    ↪ "url": None, "data_dict": {}, "categories": []}
```

Example 1
User: What is the date mentioned in this audio www.google.com/audio_file.mp3?
Response: {{”instruction”: ”Convert the audio to text and then answer the question”, ”url”: ”www.google.com/audio_file.mp3”, ”input_text”: ”What is the date?”}}
...

...

Example 5
User: Please transcribe the voice into text ./audio/audio_1.mp3 and classify the transcribed text into categories such as ’movie’, ’music’, ’painting’, or ’Other’.
Response: {”instruction”: ”Convert the audio to text, and perform text classification”, ”url”: ”./audio/audio_1.mp3”, ”categories”: [’movie’, ’music’, ’painting’, ’Other’]}

Based on the above example, parse the following:
User: {`USER_INPUT`}
Response:

Only return a `JSON`
The `JSON` keys are defined below:
instruction: textitwhat are the tasks asked in the question, it maybe one, two or three tasks. Try to find the implicit tasks as well
input_text: extract the original text or context in the input. Do not generate on your own
question: extract if there is any question asked
url: extract url passed in the user query
data_dict: extract dictionary passed in the user query
categories: extract ALL categories mentioned in the user query

Do not generate anything other than a parsable `JSON`

Domain Classification **(§3.2)**

You are professional in natural language processing task. Find which domains are related with the provided user query? You should pick domains from the following list {domains}. You MUST NOT output other domains not in the provdied list. Here are the examples:

Example 1: Answer the following questions in detail and give me a summarisation for the answer
Domains: question_answering; summarisation
...
Example 11: Summarise the transcript of the audio and find entities in it
Domains: audio; summarisation; token_classification

Provided Query: {query}. 'The order matters'
Domains:

Action Selection **(§3.2)**

You are an action selector. Given a **Task** and a list of **Actions**,
Select the crucial/mandatory actions. Follow the instructions below:
1. Pick the least amount of actions that can do the job in **Task**
2. Only select actions that are REQUIRED and NECESSARY for **Task**
3. Focus on precision of selection
4. Do not select EVALUATION or SCORE actions unless explicitly asked for in **Task**

Example 1
Task: I want to select a schema and then generate a SQL query for the a question
Actions: ["Schema-Selection", "generate_SQL", "execute_query", "validate_SQL"]
Can you understand the requirements of the Task and select necessary actions from the Actions. Do not give any explanations, only return a list and nothing else. Select at most three diverse, yet relevant actions.
Selected_Actions: ["Schema-Selection", "generate_SQL"]
...
...
Example 3
Task: Retrieve documents on renewable energy advancements and summarise the latest technologies
Actions: ["query_based_summarization", "rank_documents", "retrieve_most_relevant_document", "keyphrase_extraction", "summarization_evaluation", "get_extractive_summarization", "get_abstractive_summarization", "retrieve_multiple_documents"]
Can you understand the requirements of the Task and select necessary actions from the Actions. Do not give any explanations, only return a list and nothing else. Select at most three diverse, yet relevant actions.
Selected_Actions: ["retrieve_multiple_documents", "get_extractive_summarization"]

Task: {user_instruction}
Actions: {actions}
Can you understand the requirements of the Task and select necessary actions from the Actions. Do not give any explanations, only return a list and nothing else. Select at most three diverse, yet relevant actions.
Selected_Actions:

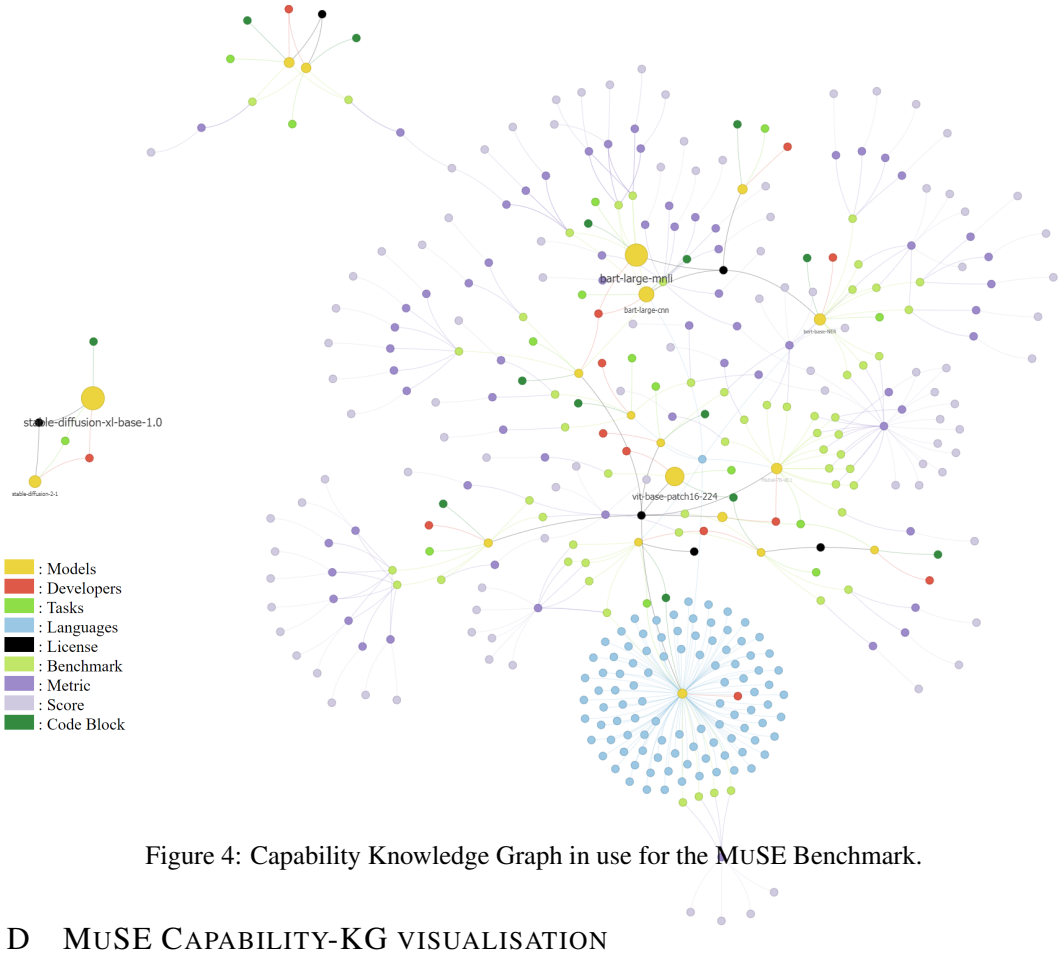

Figure 4: Capability Knowledge Graph in use for the MuSE Benchmark.

D MuSE Capability-KG visualisation

As described in Section 4.2, we introduced the MuSE benchmark in order to compare the performances of HuggingGPT, ControlLLM and Hive. The latter was declined in two sub-versions: Hive$_{\text{light}}$ having an 8-bit quantised 7B model for planning tasks and Hive relying on GPT-3.5 for the same action. In particular, MuSE comes with 100 multi-modal, multi-task, complex, natural-language queries[16].

In this Appendix, we provide, in Figure 4, a snapshot of the Capability Knowledge Graph which corresponds to the models and their associated pieces of data used by Hive to tackle the MuSE benchmark[17]. It is an excerpt, containing 461 triples involving 381 entities, of the complete C-KG which contains 125k triples for 39k distinct entities. The complete C-KG was formulated by extracting metadata from over 600,000 models listed on HuggingFace, with 26,806 models retained based on popularity metrics. In Figure 4, model nodes are depicted in yellow and are at the core of the knowledge graph, in the sense that it is around models that the C-KG building process was designed, starting from the HuggingFace model cards (see Section 3.1 for more details on this process).

Visually, this graph is twofold, indeed the two stable-diffusion models (xl-base-1.0 & 2-1) are disconnected from the rest of the graph. The remaining 13 models, on the other hand, are making a single piece of graph and organisation nodes (in red) or license nodes (in black) are often connection hubs. Interestingly, one could see that the main part of the C-KG is "bordered" by gray and purple nodes which corresponds to metrics and respective scores for the various benchmark (in bright green) data points retrieved from the `paperswithcode` API. Finally, the *image-to-text* block composed of blip-image-captioning-base and blip-vqa-base is somewhat separated from the main graph and the whisper-large-v2 which is completely surrounded by all the languages it covers (light blue nodes).

[16]See the `supplementary material` archive for all the query details and their associated data.

[17]A Web-interface to explore the MuSE C-KG is available as `supplementary material` too.