HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face

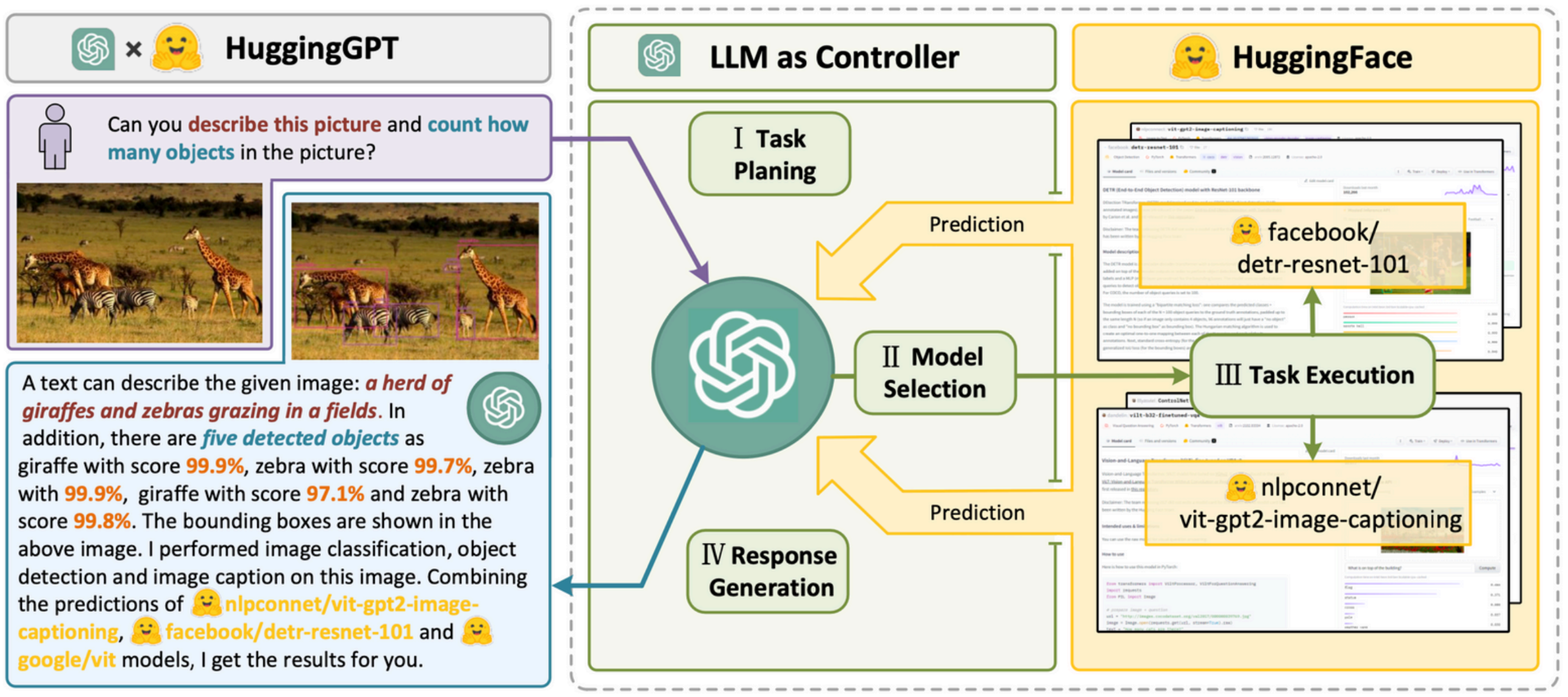
Yongliang Shen[1,2,*], **Kaitao Song**[2,*,†], **Xu Tan**[2],
Dongsheng Li[2], **Weiming Lu**[1,†], **Yueting Zhuang**[1,†]
Zhejiang University[1], Microsoft Research Asia[2]
{syl, luwm, yzhuang}@zju.edu.cn, {kaitaosong, xuta, dongsli}@microsoft.com

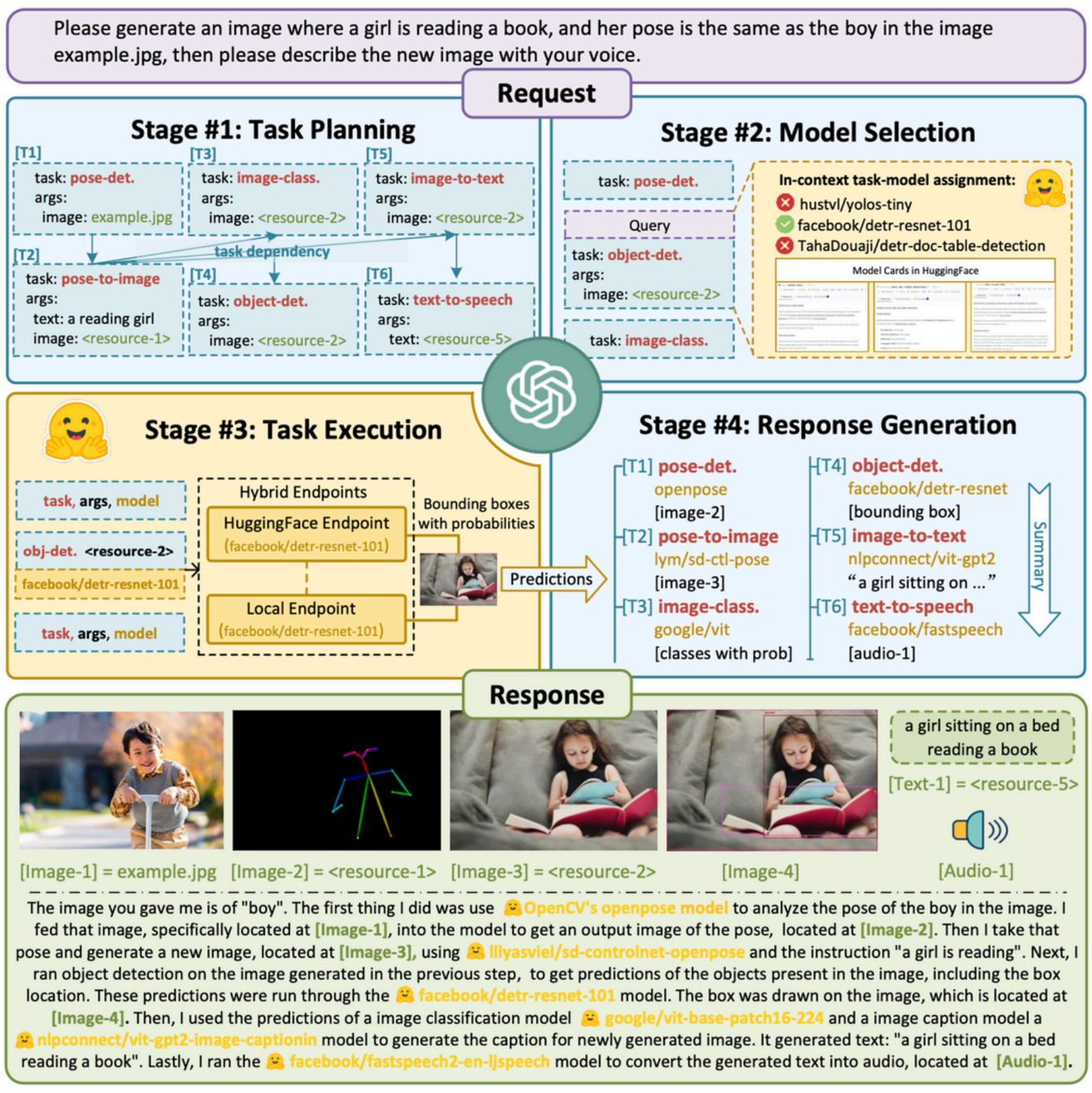
https://github.com/microsoft/JARVIS

简介

HuggingGPT提出以语言作为通用接口，赋予LLM调度外部专家模型的能力。一个LLM完成所有的AI任务时不可能的，如何设计LLM和其他专家模型交互的Agent？本文将LLM看作“中枢大脑”负责任务规划、模型选择与协作执行。通过将Hugging Face开源社区中模型的功能描述（天然的function description）融入prompt，LLM 能够理解并调用这些模型以完成具体子任务，从而构建一个开放、可扩展的多模型协作系统——一切由 prompt engineering 驱动。

万能的PROMPT ENGINEERING啊





	Prompt
Task Planning	#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: [{"task": task, "id", task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]. The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag "<resource>-task_id" represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: {{ *Available Task List* }}. Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: {{ *Demonstrations* }}. To assist with task planning, the chat history is available as {{ *Chat Logs* }}, where you can trace the user-mentioned resources and incorporate them into the task planning stage.
Demonstrations	
Can you tell me how many objects in e1.jpg?	[{"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "e1.jpg" }}]
In e2.jpg, what's the animal and what's it doing?	[{"task": "image-to-text", "id": 0, "dep":[-1], "args": {"image": "e2.jpg" }}, {"task":"image-cls", "id": 1, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task":"object-detection", "id": 2, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task": "visual-quesrion-answering", "id": 3, "dep":[-1], "args": {"text": "what's the animal doing?", "image": "e2.jpg" }}]
First generate a HED image of e3.jpg, then based on the HED image and a text “a girl reading a book”, create a new image as a response.	[{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "e3.jpg" }}, {"task": "pose-text-to-image", "id": 1, "dep": [0], "args": {"text": "a girl reading a book", "image": "<resource>-0" }}]
Model Selection	Prompt
	#2 Model Selection Stage - Given the user request and the call command, the AI assistant helps the user to select a suitable model from a list of models to process the user request. The AI assistant merely outputs the model id of the most appropriate model. The output must be in a strict JSON format: {"id": "id", "reason": "your detail reason for the choice"}. We have a list of models for you to choose from {{ *Candidate Models* }}. Please select one model from the list.
	Candidate Models
	{"model_id": model id #1, "metadata": meta-info #1, "description": description of model #1} {"model_id": model id #2, "metadata": meta-info #2, "description": description of model #2} . . . . {"model_id": model id #K, "metadata": meta-info #K, "description": description of model #K}
Response Generation	Prompt
	#4 Response Generation Stage - With the input and the inference results, the AI assistant needs to describe the process and results. The previous stages can be formed as - User Input: {{ *User Input* }}, Task Planning: {{ *Tasks* }}, Model Selection: {{ *Model Assignment* }}, Task Execution: {{ *Predictions* }}. You must first answer the user's request in a straightforward manner. Then describe the task process and show your analysis and model inference results to the user in the first person. If inference results contain a file path, must tell the user the complete file path. If there is nothing in the results, please tell me you can't make it.