Agents - HuggingGPT

Rowel Atienza, PhD
University of the Philippines
github.com/roatienza
2023

Why Agents

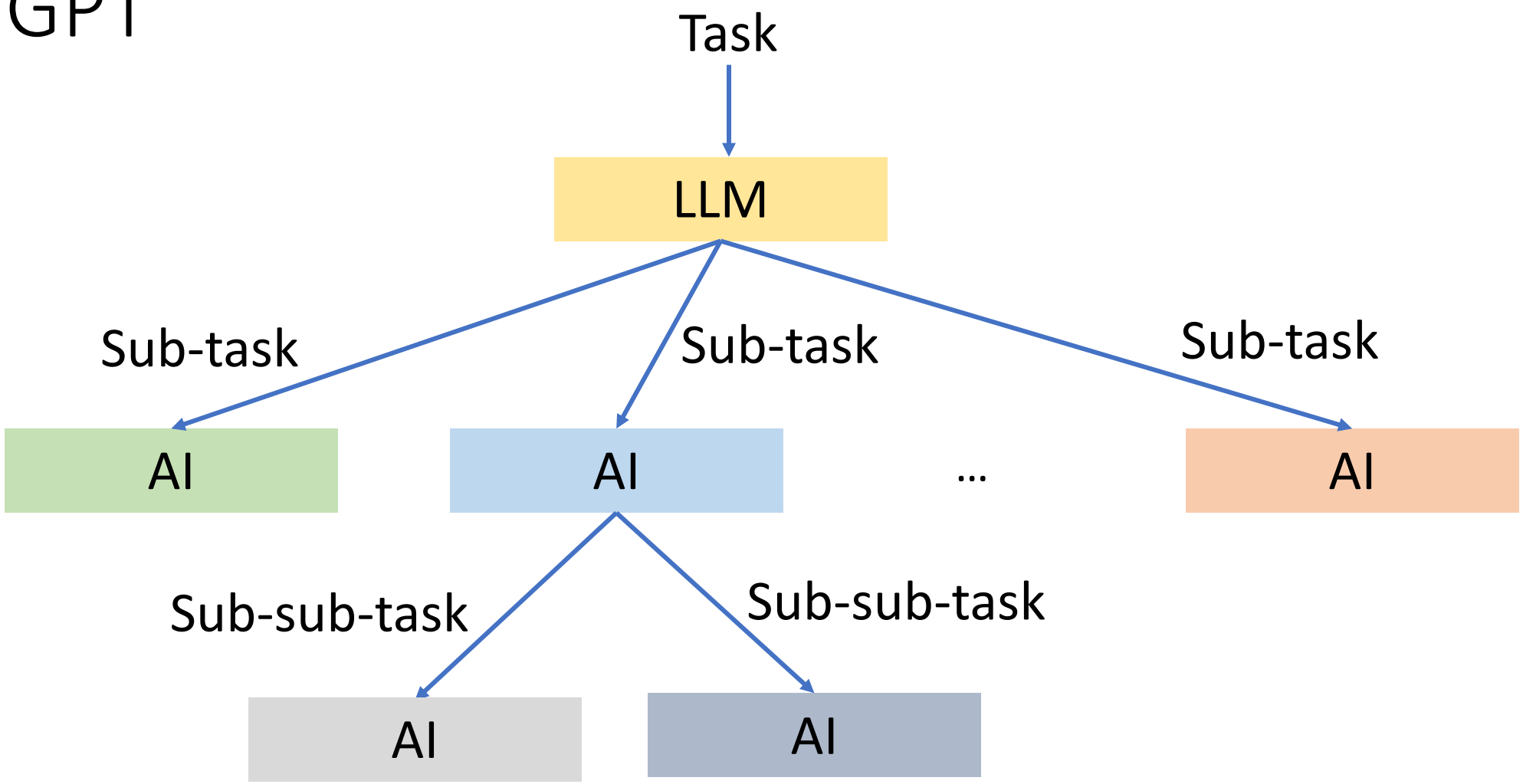
- Agents can autonomously solve tasks with the help of tools and AI models

Why HuggingGPT (HGPT)

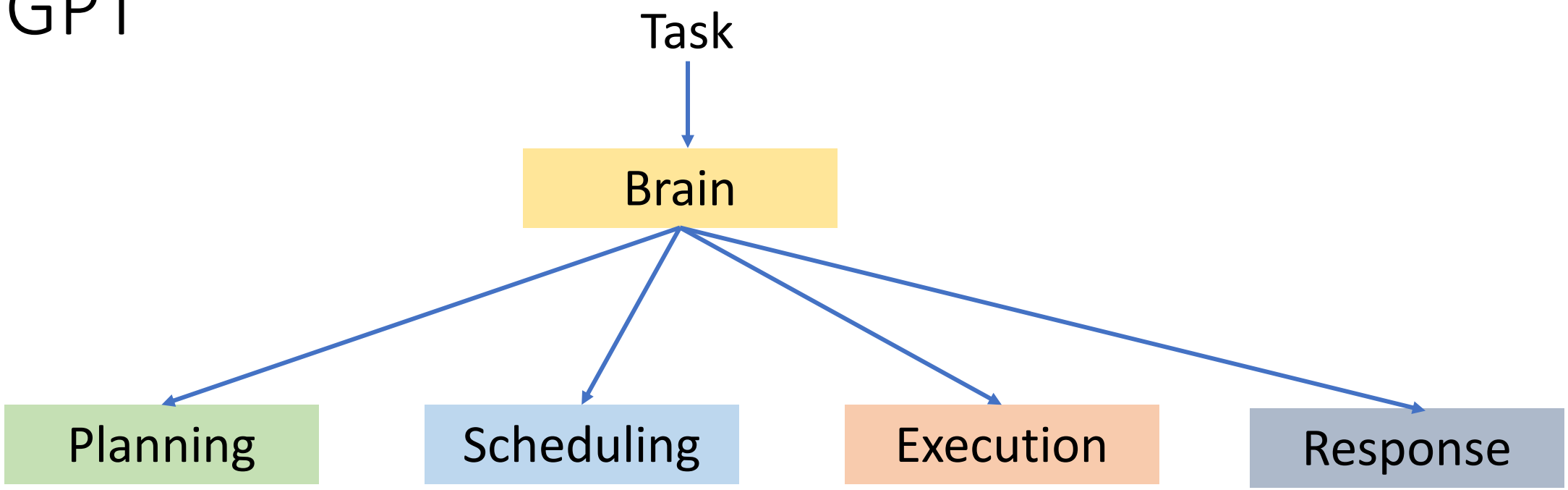
- A framework for executing tasks using available tools and AI models on HuggingFace and public domains (eg github)

Shen, Yongliang, et al. "Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface." *arXiv preprint arXiv:2303.17580* (2023).

HGPT


Task
LLM
Sub-task
Sub-task
Sub-task
AI
AI
…
AI
Sub-sub-task
Sub-sub-task
AI
AI

HGPT



HGPT

- **Task Planning**: Using LLM to analyze the requests of users to understand their intention and disassemble them into possible solvable tasks via prompts.
- **Model Selection**: To solve the planned tasks, LLM selects expert models that are hosted on HuggingFace or local inference server based on model descriptions.
- **Task Execution**: Invoke and execute each selected model and return the results to LLM
- **Response Generation**: Finally, using LLM to integrate the prediction of all models and generate answers for users.

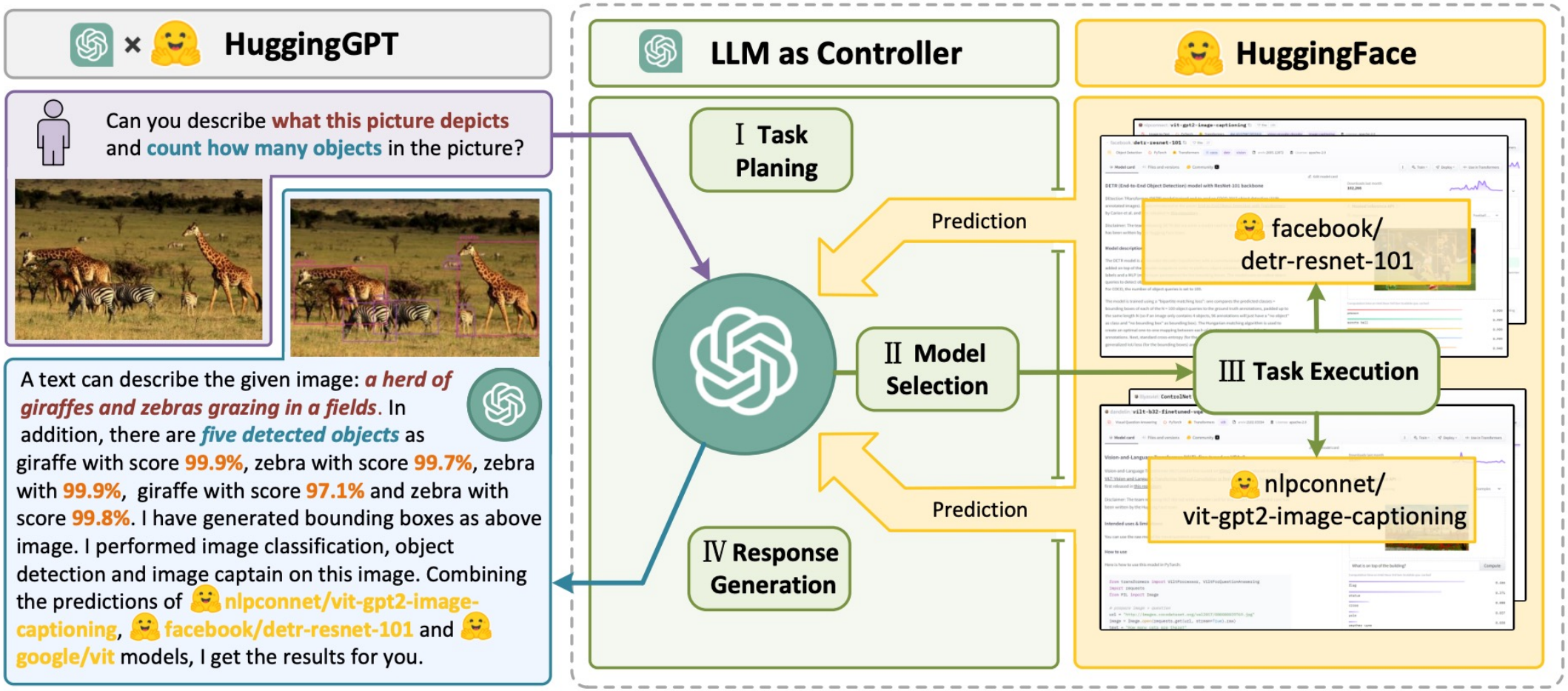
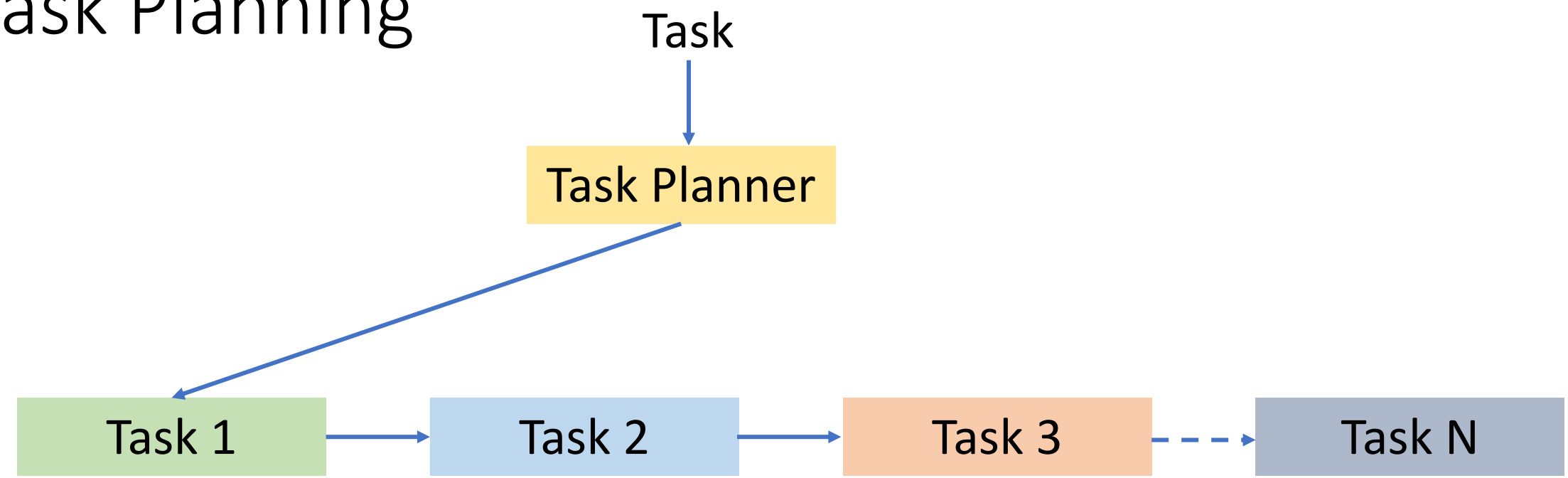


Figure 1: *Language serves as an interface for LLMs (e.g., ChatGPT) to connect numerous AI models (e.g., those in Hugging Face) for solving complicated AI tasks.* In this concept, an LLM acts as a controller, managing and organizing the cooperation of expert models. The LLM first plans a list of tasks based on the user request and then assigns expert models to each task. After the experts execute the tasks, the LLM collects the results and responds to the user.

Task Planning

Task Planning



Task Planner Prompts

- Specification-based Instruction – template-based, slot-filling
- Demonstration-based Parsing – In-context learning

Specification-based Instruction

- Task ID provides a unique identifier for task planning
- Task types cover different tasks in language, visual, video, audio, etc.
- Task dependencies define the pre-requisite tasks required for execution.
- Task arguments contain the list of required arguments for task execution.

Task Arguments

Task	Args
Text-cls	text
Token-cls	text
Text2text-generation	text
Summarization	text
Translation	text
Question-answering	text
Conversational	text
Text-generation	text
Tabular-cls	text

Table 1: NLP tasks.

Task	Args
Image-to-text	image
Text-to-image	image
VQA	text + image
Segmentation	image
DQA	text + image
Image-cls	image
Image-to-image	image
Object-detection	image
Controlnet-sd	image

Table 2: CV tasks.

Task	Args
Text-to-speech	text
Audio-cls	audio
ASR	audio
Audio-to-audio	audio

Table 3: Audio tasks.

Task	Args
Text-to-video	text
Video-cls	video

Table 4: Video tasks.

Demonstration-based Parsing

- Input-output examples
- Demonstrate dependencies between tasks
- Show order of execution

Model Selection

Model Selection

- Select the appropriate model for each task using model description in HuggingFace
- In-context Model Assignment based on prompt

Task Execution

Task Execution

- On HuggingFace or local endpoint, run the inference and return the result to the LLM
- Models that do not have dependencies can be parallelized

Hybrid Endpoints

- Local endpoints – fast but few
  - Higher priority
- HuggingFace endpoints – slow but many
  - Lower priority

Resource Dependency

- Future resources are hard to predict in spite of task planning
- Solution – track all resources generated by tasks
  - Future tasks can reuse these resources if needed

Response Generation

Response Generation

- Integrate all information from the previous three stages (task planning, model selection, and task execution) into a concise summary including:
  - the list of planned tasks
  - selected models for the tasks,
  - inference results of the models.
- LLMs convert inference results into a human readable/understandable form

Experiments

Experimental Setup

- **gpt-3.5-turbo**
- `temperature = 0`
  - Predictable results
- `logit_bias = 0.2`
  - Adhere to format

Task Planning Prompt

Task Planning	Prompt	
	#1 Task Planning Stage - The AI assistant can parse user input to several tasks: [{"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 that the current task relies on. A special tag "<resource>-task_id" refers to the generated text image, audio and video in the dependency task with id as task_id. The task MUST be selected from the following options: {{ Available Task List }}. There is a logical relationship between tasks, please note their order. If the user input can't be parsed, you need to reply empty JSON. Here are several cases for your reference: {{ Demonstrations }}. The chat history is recorded as {{ Chat Logs }}. From the chat logs, you can find the path of the user-mentioned resources for your task planning.	
	Demonstrations	
	Look at /exp1.jpg, Can you tell me how many objects in the picture?	[{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg" }}, {"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg" }}]
	In /exp2.jpg, what's the animal and what's it doing?	[{"task": "image-to-text", "id": 0, "dep":[-1], "args": {"image": "/exp2.jpg" }}, {"task":"image-classification", "id": 1, "dep": [-1], "args": {"image": "/exp2.jpg" }}, {"task":"object-detection", "id": 2, "dep": [-1], "args": {"image": "/exp2.jpg" }}, {"task": "visual-question-answering", "id": 3, "dep":[-1], "args": {"text": "What's the animal doing?", "image": "/exp2.jpg" }}]
	Given an image /exp3.jpg, first generate a hed image, then based on the hed image and a prompt: a girl is reading a book, you need to reply with a new image.	[{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/examples/boy.jpg" }}, {"task": "openpose-control", "id": 1, "dep": [-1], "args": {"image": "/examples/boy.jpg" }}, {"task": "openpose-text-to-image", "id": 2, "dep": [1], "args": {"text": "a girl is reading a book", "image": "<resource>-1" }}]

Model Selection Prompt

	Prompt
Model Selection	#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-infos #1, "description": description of model #1} {"model_id": model id #2, "metadata": meta-infos #2, "description": description of model #2} [Select top-K candidate models with the same task type as the current task] {"model_id": model id #K, "metadata": meta-infos #K, "description": description of model #K}

Response Prompt

	Prompt
Response Generation	#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.

Query: generate a video with the title "An astronaut is walking in space" and dub it.

Response: Sure. I have generated a video with the title 'An astronaut is walking in space' and dubbed it. The workflow I used is as follows: First, I used the text-to-video model ' 🤗 damo-vilab/text-to-video-ms-1.7b' to generate a video from the text. The inference result is the generated video ' /videos/5696.mp4'. Second, I used the text-to-speech model ' 🤗 facebook/tts_transformer-es-css10' to generate an audio from the text. The inference result is the generated audio ' /audios/3f9e.flac '. Finally, I combined the video and audio to generate the dubbed video. I hope this answers your question. Is there anything else I can help you with?

Task Planning:

1: {"task": "text-to-video", "id": 0, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}
2: {"task": "text-to-speech", "id": 1, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}

Execution Results:

1. {"generated video": "/videos/5696.mp4"} **2.** {"generated audio": "/audios/3f9e.flac"}

Query: read the image example.jpg for me.

Response: Sure. I have read the image example.jpg for you. The inference result is **'INVOICE NO : 18301/102/T0305'**. The model used for this task is microsoft/trocr-base-printed, which is best suited for the task of image-to-text as it is fine-tuned on the SROIE dataset which is specifically designed for this task. Then I have converted the text to audio, the generated audio is /audios/da5s.wav and the model used for this task is facebook/fastspeech2-en-ljspeech, which is a FastSpeech 2 text-to-speech model which is suitable for the task of text-to-speech. Is there anything else I can help you with?

Task Planning:

1: {"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "example.jpg"}}
2: {"task": "text-to-speech", "id": 1, "dep": [0], "args": {"text": "<resource-1>"}

Execution Results:

1. {"generated text": "INVOICE NO : 18301/102/T0305"} **2.** {"generated audio": "/audios/da5s.wav"}

Code demo