

Large Language Models as the Interface to tools/plugins/APIs



Image Credits: Bing Image Creator
Prompt: Create me a web with the word
"LLM" in the center and plenty of cool
stuff at the peripherals

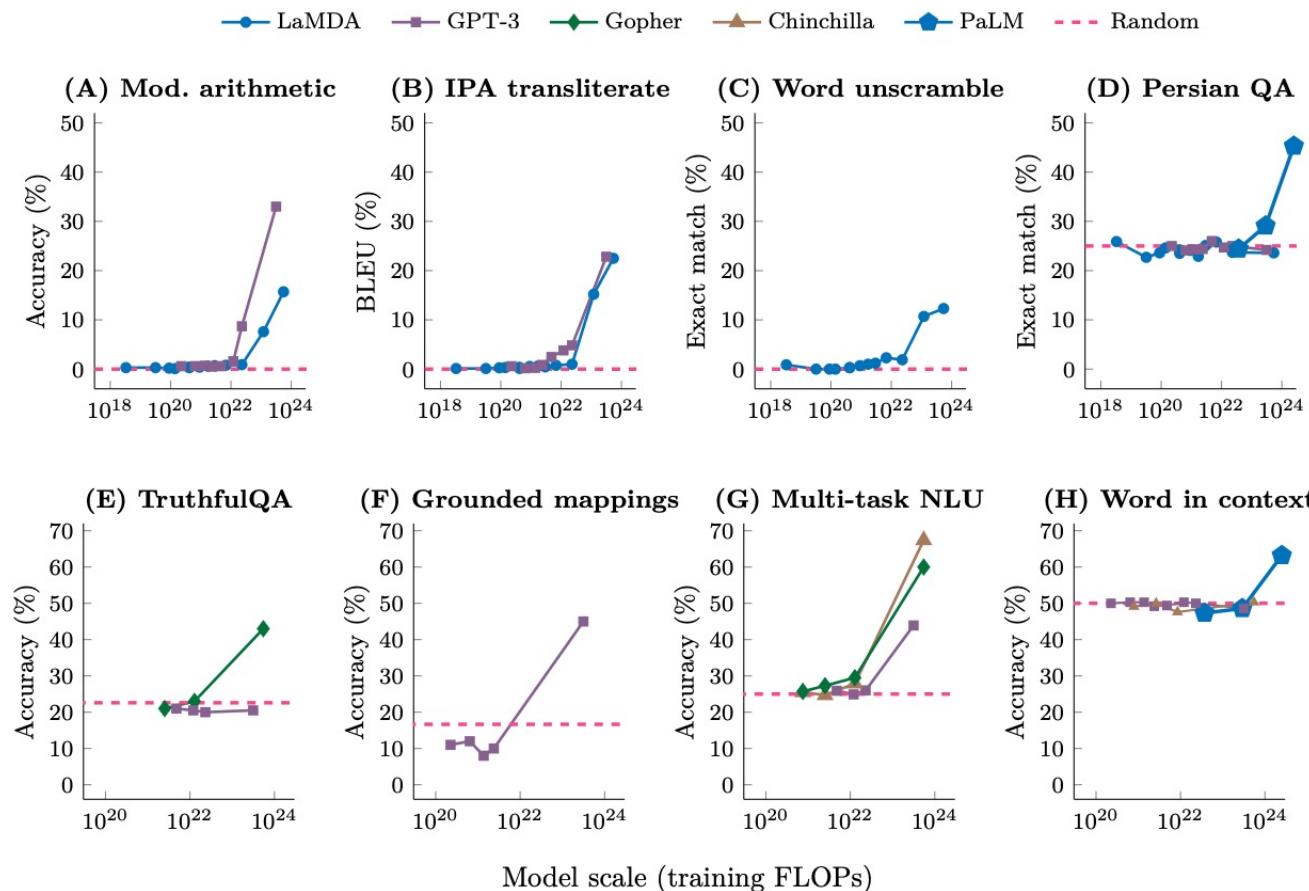
Toolformer, Visual ChatGPT, Wolfram Alpha Plugin

Presented by: John Tan Chong Min

LLM + Tools: Part 1

Overview

Can emergence solve everything?



- “Emergent abilities cannot be predicted simply by extrapolating the performance of smaller models”

- How big can you go?
- Can you do multi-modal processing?

“I know Kung Fu”

- Tools/plugins/API imbue more knowledge and multi-modal capabilities to LLMs



Aim

- LLMs are great at inferring statistical patterns, but they are pretty bad at doing inference for math, logic and other more specialized tasks
- LLMs also have limited capabilities to process non-text inputs
- Let other APIs handle the parts the LLM are bad at, and fuse both together



LLMs vs Tools/Plugins/API



- Flexible input format
- Able to capture intent well
- Unable to learn fast and perform complex tasks
- (Usually) Inflexible input format
- May be of a non-text modality and requires an interface
- Faster learning, ability to perform complex tasks



Key Guidelines

- The use of tools should be learned in a self-supervised way without requiring large amounts of human annotations
- The LLM should not lose any of its generality and should be able to decide for itself when and how to use which tool

What's possible?

Some cool stuff already done so far

Question-Answer API

C

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: Singapore is a city in Southeast Asia.

Output:



Singapore is [QA("What type of geographic location is Singapore?")] a city in [QA("Which region is Singapore located in?")] Southeast Asia.

- Few-shot learning to teach LLM how to call the API

Prompt taken from Toolformer paper

Math API

C

Your task is to add calls for mathematical evaluation to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[Math(question)]", where "question" is the question you want to ask. Here are some examples of API calls:

Input: What is 102×538 ?

Output: The answer is [Math("102*538")].

Input: Four people have three rings each. How many rings are there?

Output: There are [Math("4*3")] rings in total.

Input: We have \$300 in the bank. We spend \$102 on groceries and \$30 on transport. How much is left in the bank?

Output:



After spending \$102 on groceries and \$30 on transport, there is [Math("300-102-30")] dollars left in the bank.

- My own experiments to teach it to use math

- Even works for verbose interpretation

Math API (Evaluated by ChatGPT)

- C Input: We have \$300 in the bank. We spend \$102 on groceries and \$30 on transport. 30% of the remaining money in the bank was used for charity. How much is left in the bank?
Output:



After spending \$102 on groceries and \$30 on transport, there is $[Math("300-102-30")]$ dollars left in the bank. Then, 30% of that amount, or $[Math("(300-102-30)*0.3")]$ dollars, was used for charity. Therefore, there is $[Math("(300-102-30)-(300-102-30)*0.3")]$ dollars left in the bank.

- If left to its own next-token prediction, ChatGPT will get some of the calculations wrong



evaluate the Math() expressions above and rewrite the answer.

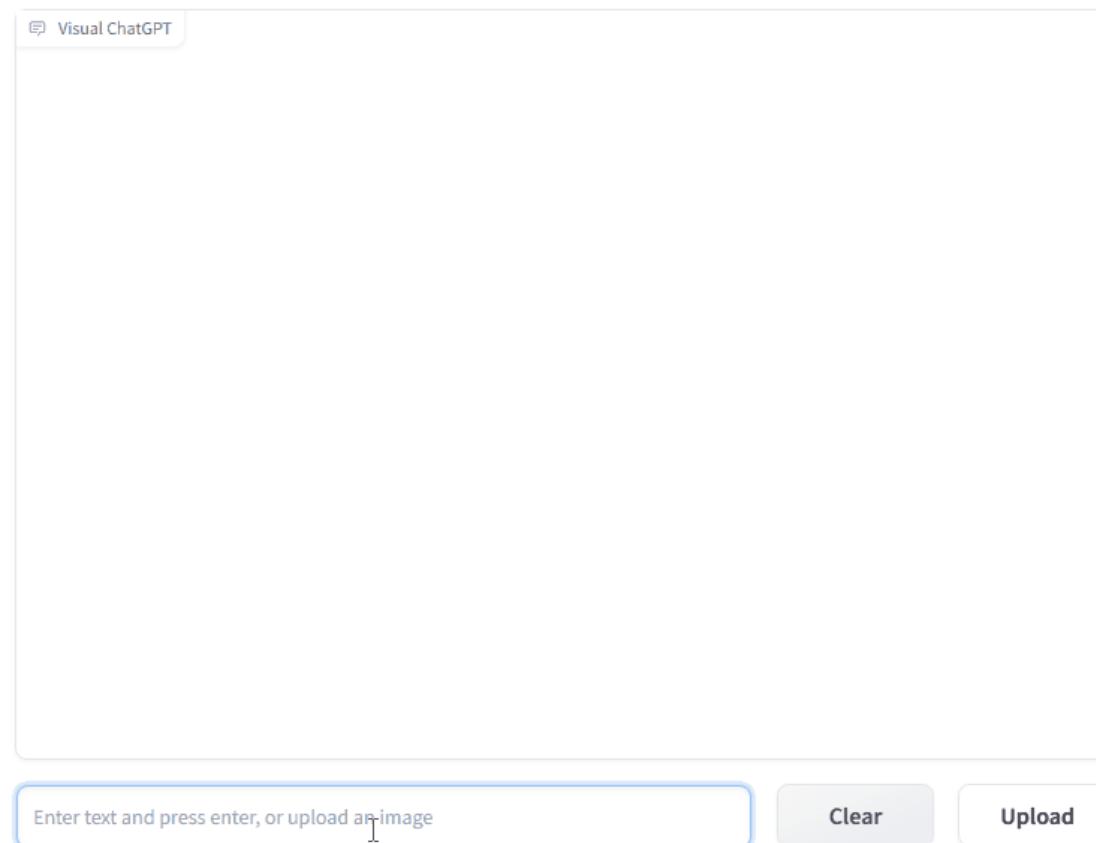


Evaluating the Math() expressions in the previous outputs, we get:

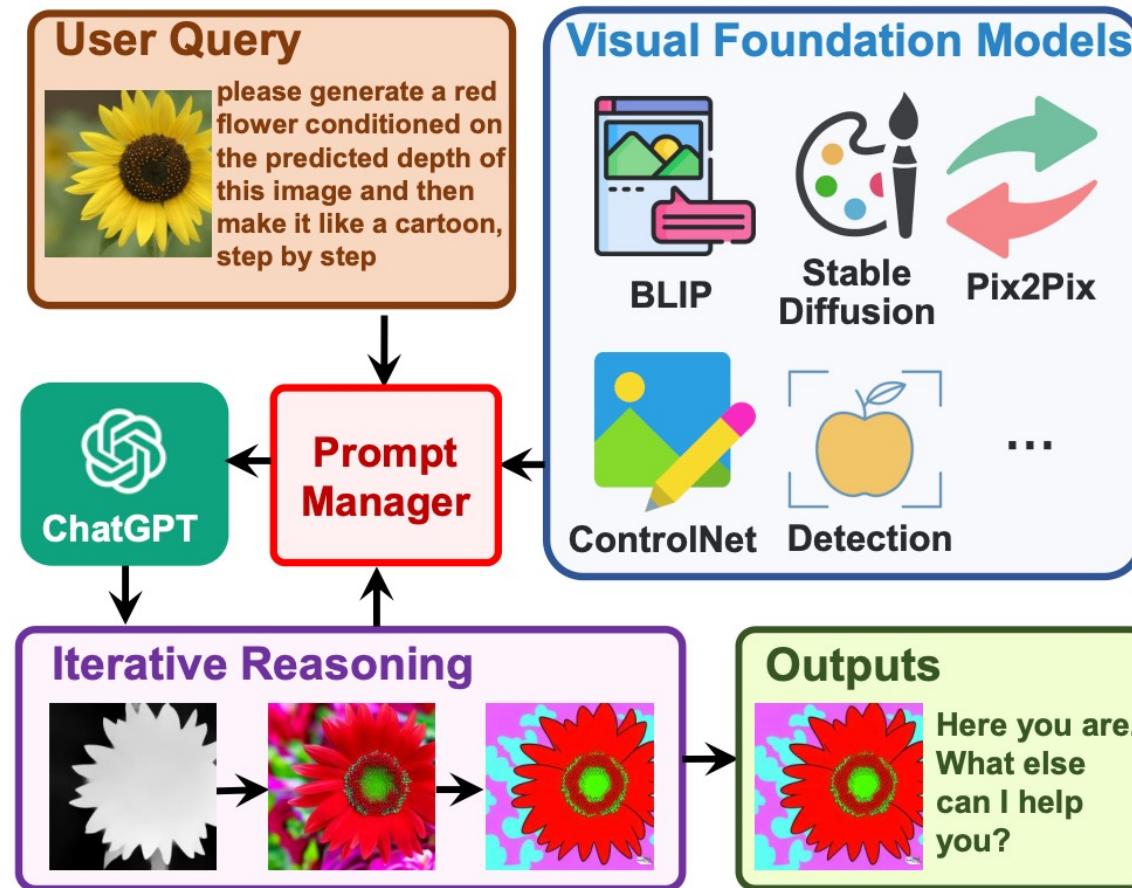
- The answer is 54996.
- There are 12 rings in total.
- After spending \$102 on groceries and \$30 on transport, there is \$168 left in the bank.
Then, 30% of that amount, or \$50.4, was used for charity. Therefore, there is \$117.6 left in the bank.

Correct answer: $102*538 = 54876$

Visual ChatGPT: Tools for Images



Visual ChatGPT: Tools for Images



Visual ChatGPT: Talking, Drawing and Editing with Visual Foundational Models. Wu et al. 2023.

Wolfram Alpha Plugin: Tools for Computation

Used Wolfram

REQUEST TO WOLFRAM

```
{ "input": "distance from Chicago to Tokyo" }
```

RESPONSE FROM WOLFRAM

Query:
"distance from Chicago to Tokyo"

Input interpretation:
distance from Chicago, Illinois to Tokyo, Japan

Spoken result:
The distance from the center of Chicago, Illinois to the center of Tokyo is about 6313 miles

Result:
6313 miles

Unit conversions:
10160 km (kilometers)
1.016x10⁷ meters
5486 nmi (nautical miles)

Direct travel times:
aircraft (550 mph) | 11 hours 30 minutes
sound | 8 hours 20 minutes
light in fiber | 47 ms (milliseconds)
light in vacuum | 34 ms (milliseconds)
(assuming constant-speed great-circle path)

Map:
image: <https://www6b3.wolframalpha.com/Calculate/MSP/MSP49041h0i6i8e288gcfg400005e5845aa>

Fraction of Earth circumference:
0.25 ~ 1 / 4

Wolfram|Alpha website result for "distance from Chicago to Tokyo":
<https://www.wolframalpha.com/input/?i=distance+from+Chicago+to+Tokyo>

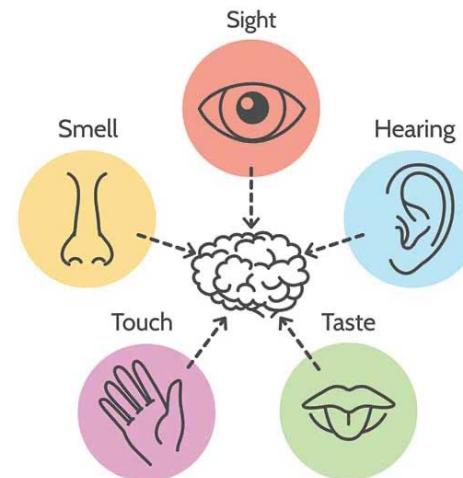
 **WolframAlpha** computational intelligence.

 Step-by-Step Solutions	 Units & Measures	 People	 Personal Health
 Elementary Math	 Physics	 Arts & Media	 Personal Finance
 Algebra	 Chemistry	 Dates & Times	 Surprises
 Plotting & Graphics	 Engineering	 Words & Linguistics	 Entertainment
 Calculus & Analysis	 Computational Sciences	 Money & Finance	 Household Science
 Geometry	 Earth Sciences	 Food & Nutrition	 Household Math
 Differential Equations	 Materials	 Political Geography	 Hobbies
 Statistics	 Transportation	 History	 Today's World

<https://writings.stephenwolfram.com/2023/03/chatgpt-gets-its-wolfram-superpowers/>

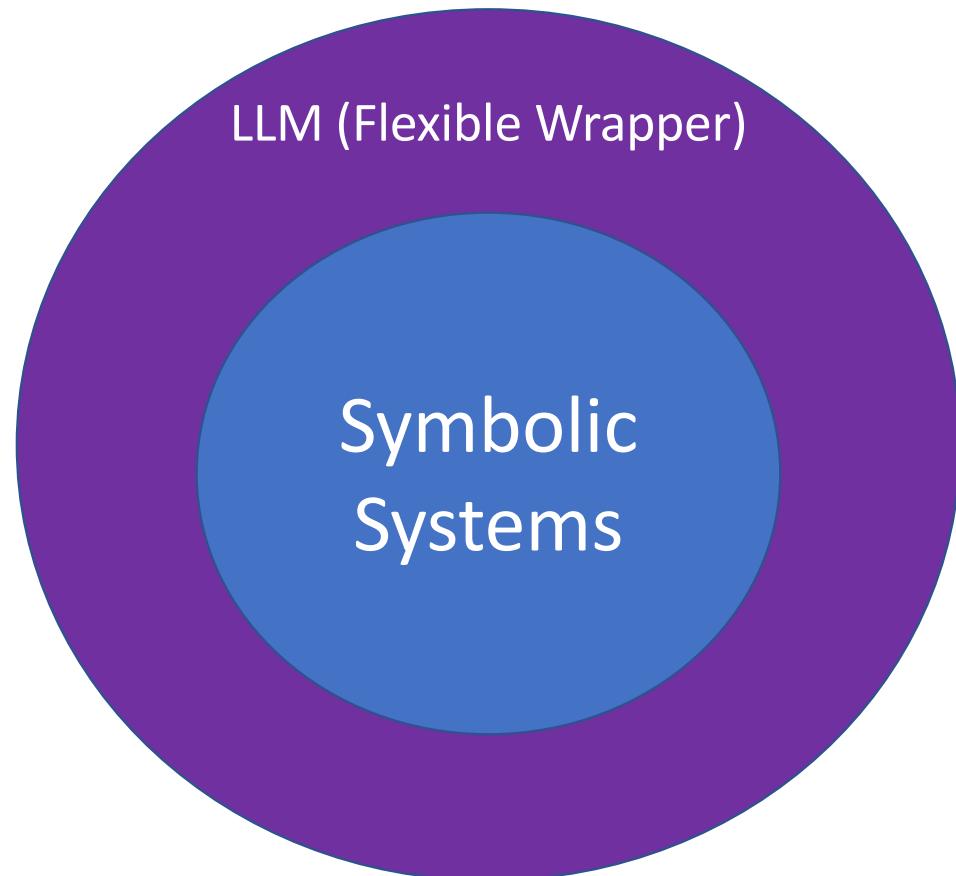
What kinds of APIs can be interfaced?

- The possibilities are endless
 - Logic-based processing
 - Fact-checker
 - Visual processing
 - Video processing
 - Touch processing
 - Audio processing
 - Web search
 - External memory
 - Knowledge Web
 - Coding tool
 - Chess-playing program

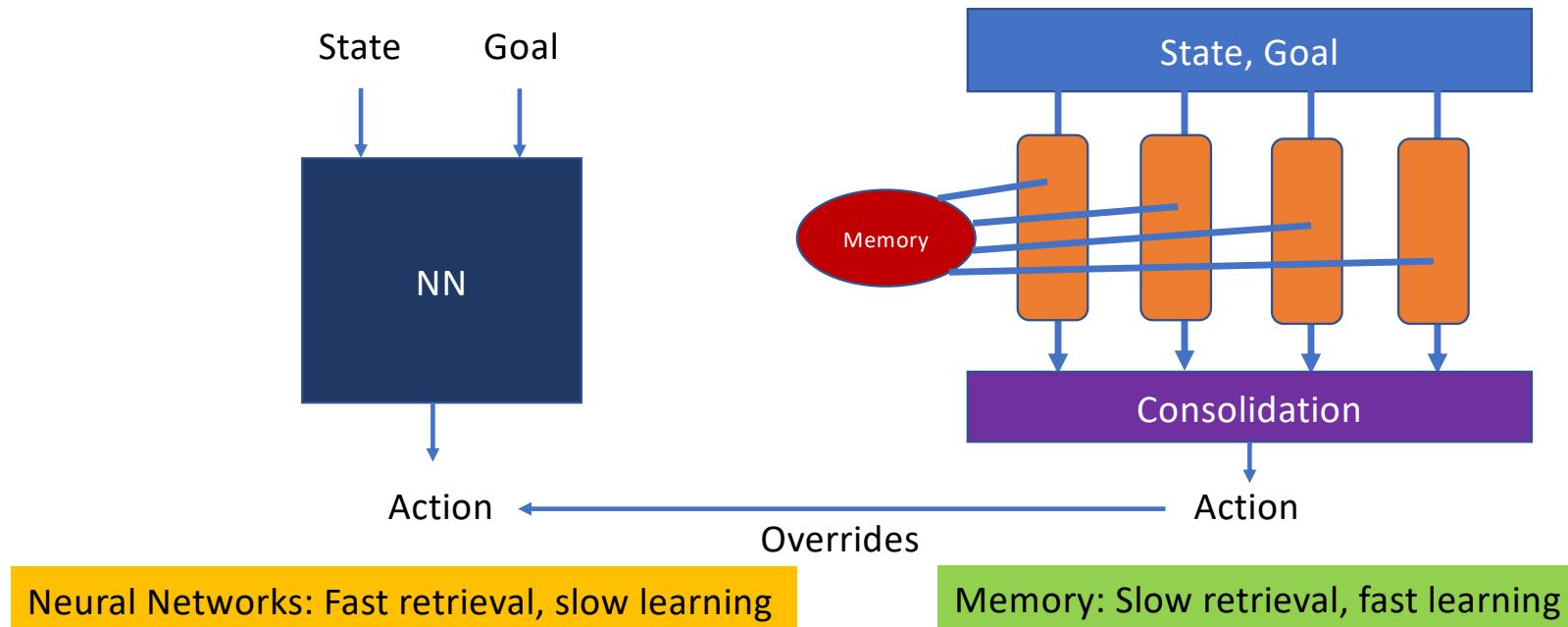


Neuro-symbolic AI

- Symbolic systems have difficulty scaling up
 - Logical inference over many arbitrary entities is difficult
- LLMs can help to interpret the input in the form that symbolic systems can accept
- E.g. Wolfram Alpha plugin, Logical Inference plugin (e.g. via LangChain)



Link to my work (Learning, Fast & Slow)



This can be the LLM!

This can be the memory retrieval tool to retrieve relevant experiences!

How it works (ToolFormer)

Few-shot learning

Optional Fine-tuning

How to do learning?

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: x

Output:

- Few-shot learning: Use prompt engineering to learn where to insert API calls
- Fine-tuning (optional but saves costs due to not requiring prompt):
 - Fine-tune LLM on dataset with the API call tokens already inserted
 - Only keep API call tokens that improve performance over baseline model without API calls

What API calls are useful?

Example	$L_i^- - L_i^+$	Useful
The Flodden Window (a war memorial dedicated to The Middleton Archers), in the Grade I-listed Church of St Leonard in Middleton is said to be the oldest war memorial in the United Kingdom. <API> WikiSearch(War memorial Flodden) → Battle of Flodden > Commemoration > The stained-glass Flodden Window in Middleton Parish Church [...] was constructed by Sir Richard Assheton in memory of the Battle of Flodden and the archers from Middleton who fought in it. </API> Sir Richard Assheton of Middleton (who built St Leonard) was granted knighthood [...]	5.49	✓
Note: The WL will be open on Friday, <API> Calendar() → Today is Thursday, March 9, 2017. </API> March 10, and Sunday, March 19 for regular hours.	2.11	✓
The Nile has an approximate length of <API> QA(What is the approximate length of the Nile?) → 6,853 km </API> 6,853 kilometers, the White Nile being its main source.	2.08	✓
If Venus had an atmosphere similar to Earth's then you would expect Venus' mean temperature to be 499 K (1.74×287) rather than 735 K which is <API> Calculator($735 / 499$) → 1.47 </API> 1.47 ($735 / 499$) times hotter than it should be.	1.59	✓
You are here: Home / Featured / Catch this fast train to success! <API> WikiSearch(Fast train success) → Fast Train > It also peaked at #23 on the Canadian CHUM singles chart, on June 26, 1971. The success of this single established Myles Goodwyn as the band's main songwriter, and made it possible for April Wine to record a second album. </API> Don't wait weeks and [...]	0.92	✗

$$\mathbf{e}(c) = \langle \text{API} \rangle a_c(i_c) \langle / \text{API} \rangle$$

$$\mathbf{e}(c, r) = \langle \text{API} \rangle a_c(i_c) \rightarrow r \langle / \text{API} \rangle$$

$$L_i^+ = L_i(\mathbf{e}(c_i, r_i))$$

$$L_i^- = \min(L_i(\varepsilon), L_i(\mathbf{e}(c_i, \varepsilon)))$$

ε denotes an empty sequence

Results

- Tools enable a 6.7B GPT-J model to outperform even a 175B GPT-3!

Model	SQuAD	Google-RE	T-REx
GPT-J	17.8	4.9	31.9
GPT-J + CC	19.2	5.6	33.2
Toolformer (disabled)	22.1	6.3	34.9
Toolformer	<u>33.8</u>	<u>11.5</u>	<u>53.5</u>
OPT (66B)	21.6	2.9	30.1
GPT-3 (175B)	26.8	7.0	39.8

Table 3: Results on subsets of LAMA. Toolformer uses the question answering tool for most examples, clearly outperforming all baselines of the same size and achieving results competitive with GPT-3 (175B).

*CC means fine-tuned on dataset C, which is a subset that has the most benefit for the tools.

Toolformer is trained on C*, which is the dataset with API prompts annotated

Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	<u>40.4</u>	<u>29.4</u>	<u>44.0</u>
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8

Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

How it works (Visual ChatGPT)

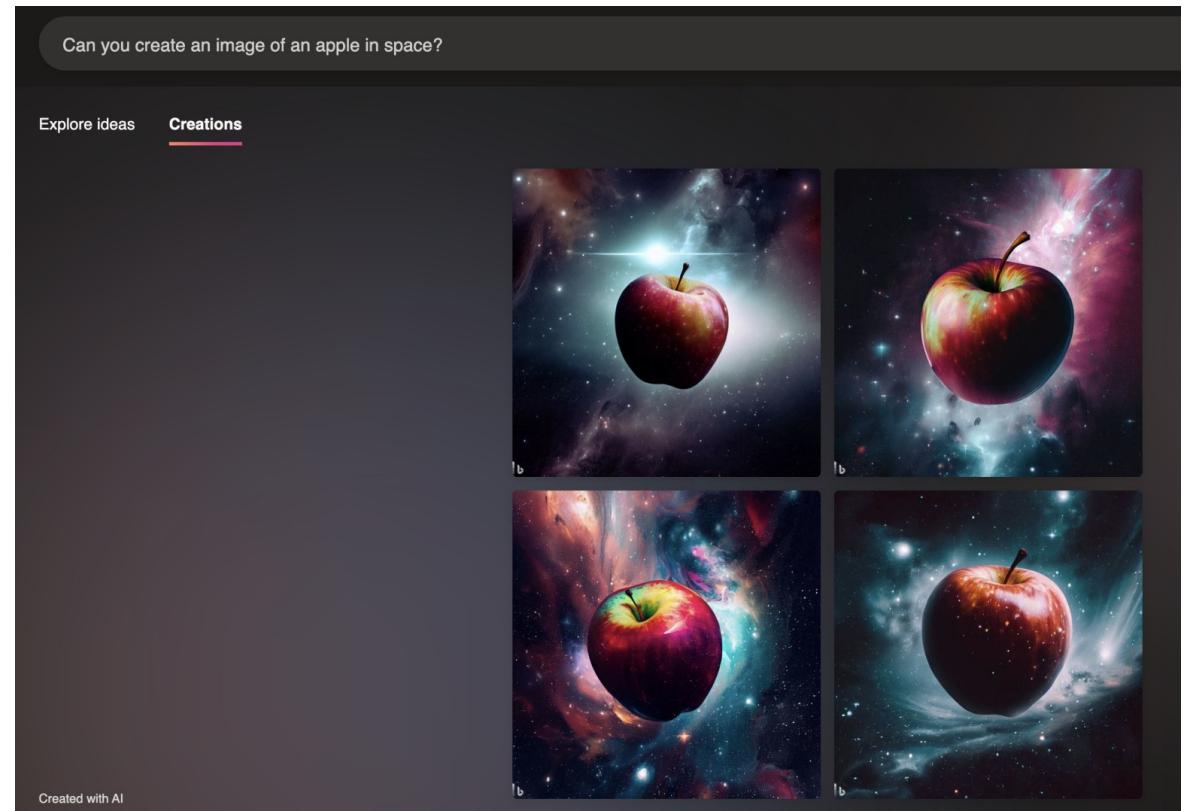
Chain of Thought

No fine-tuning

Multiple modalities

Tools for multi-modal processing

- Tools enable processing of modalities in which an instruction can be difficult/not natural to give in
 - Image
 - Video
 - Audio
 - Touch



Chain of Thought (CoT) prompting

- CoT prompting can be used to enable the LLM to think of further elaboration
- LLM can also be prompted to think of whether it should use tools!

CHAIN-OF-THOUGHT PLANNING ROLLOUT WITH PALM-SAYCAN.

Human: Can you bring a fruit-flavored drink without caffeine?

Explanation: The user has asked for a drink that is fruit-flavored and does not have caffeine, I will bring the lime soda.

Robot: 1. find a lime soda, 2. pick up the lime soda, 3. bring it to you, 4. put down the lime soda, 5. done

Human: Bring me the spicy chips.

Explanation: The user has asked for chips that are spicy, I will bring the Jalapeno chips.

Robot: 1. find the jalapeno chips, 2. pick up the jalapeno chips, 3. bring it to you, 4. put down the jalapeno chips, 5.done

Human: The snack is too small, bring me something more filling.

Explanation: The user has asked for a snack that is more filling, I will bring the multigrain chips.

Robot: 1. find the multigrain chips, 2. pick up the multigrain chips, 3. bring it to you, 4. put down the multigrain chips, 5. done

Table 4: Chain-of-thought planning rollout with PaLM-SayCan. The highlighted part is the chain of thought generated by PaLM-SayCan.

ReAct

```
question = "Author David Chanoff has collaborated with a U.S. Navy admiral who served as the ambassador to the United Kingdom under which President?"  
react.run(question)
```

> Entering new AgentExecutor chain...

Thought 1: I need to search David Chanoff and find the U.S. Navy admiral he collaborated with.

Action 1: Search[David Chanoff]

Observation 1: David Chanoff is a noted author of non-fiction work. His work has typically involved collaborations with the principal protagonist of the work concerned. His collaborators have included; Augustus A. White, Joycelyn Elders, Đoàn Văn Toại, William J. Crowe, Ariel Sharon, Kenneth Good and Felix Zandman. He has also written about a wide range of subjects including literary history, education and foreign for The Washington Post, The New Republic and The New York Times Magazine. He has published more than twelve books.

Thought 2: The U.S. Navy admiral David Chanoff collaborated with is William J. Crowe.

Action 2: Search[William J. Crowe]

Observation 2: William James Crowe Jr. (January 2, 1925 – October 18, 2007) was a United States Navy admiral and diplomat who served as the 11th chairman of the Joint Chiefs of Staff under Presidents Ronald Reagan and George H. W. Bush, and as the ambassador to the United Kingdom and Chair of the Intelligence Oversight Board under President Bill Clinton.

Thought 3: The President William J. Crowe served as the ambassador to the United Kingdom under is Bill Clinton.

Action 3: Finish[Bill Clinton]

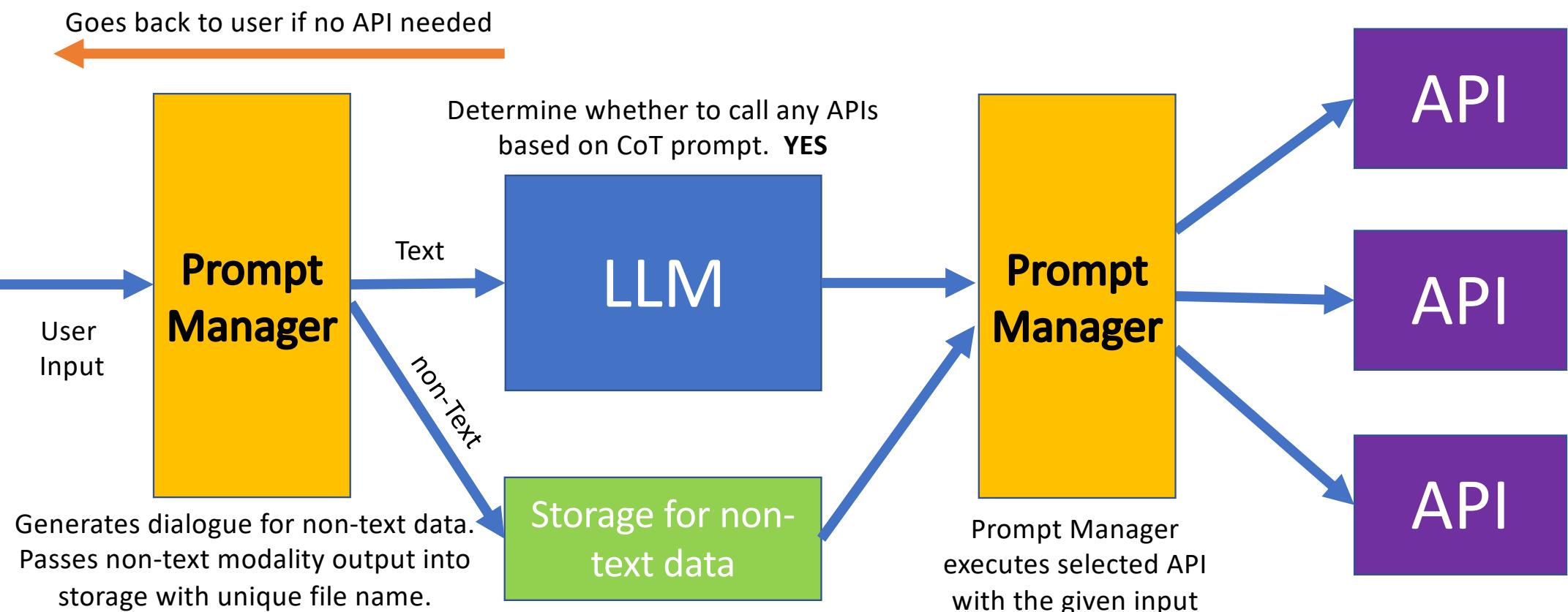
> Finished AgentExecutor chain.

'Bill Clinton'

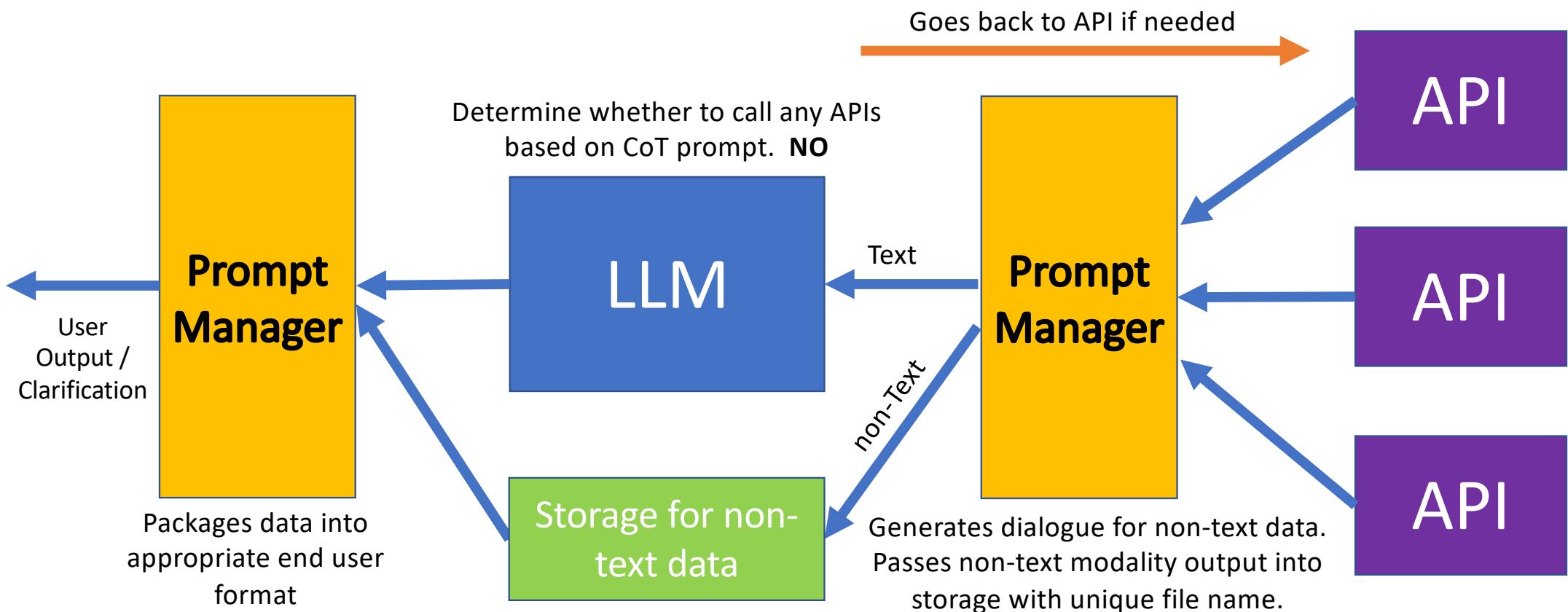
CoT using ReAct:

- Thought
- Action (Tools)
- Observation

Overall workflow (Needs API)

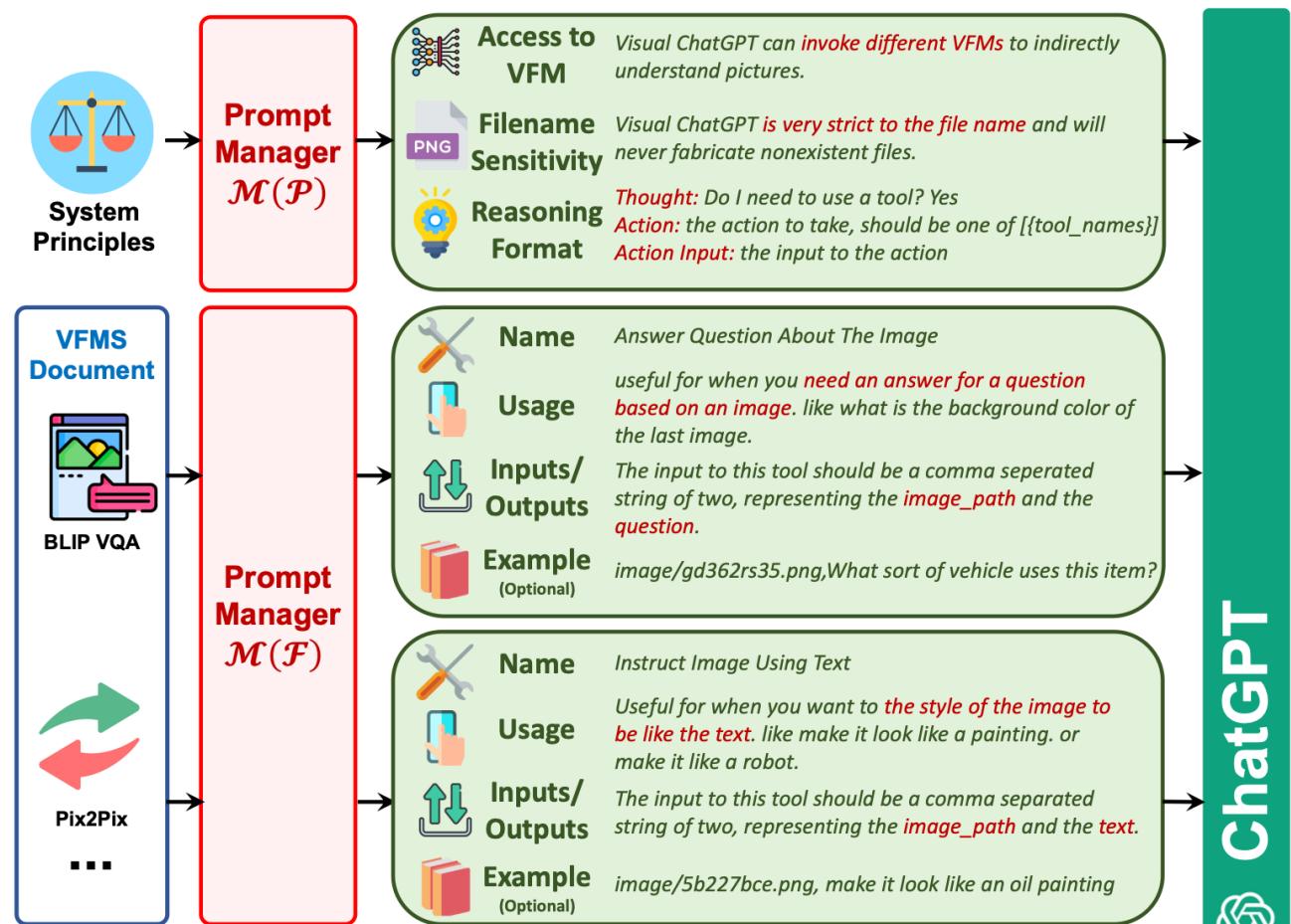


Overall workflow (No need API)

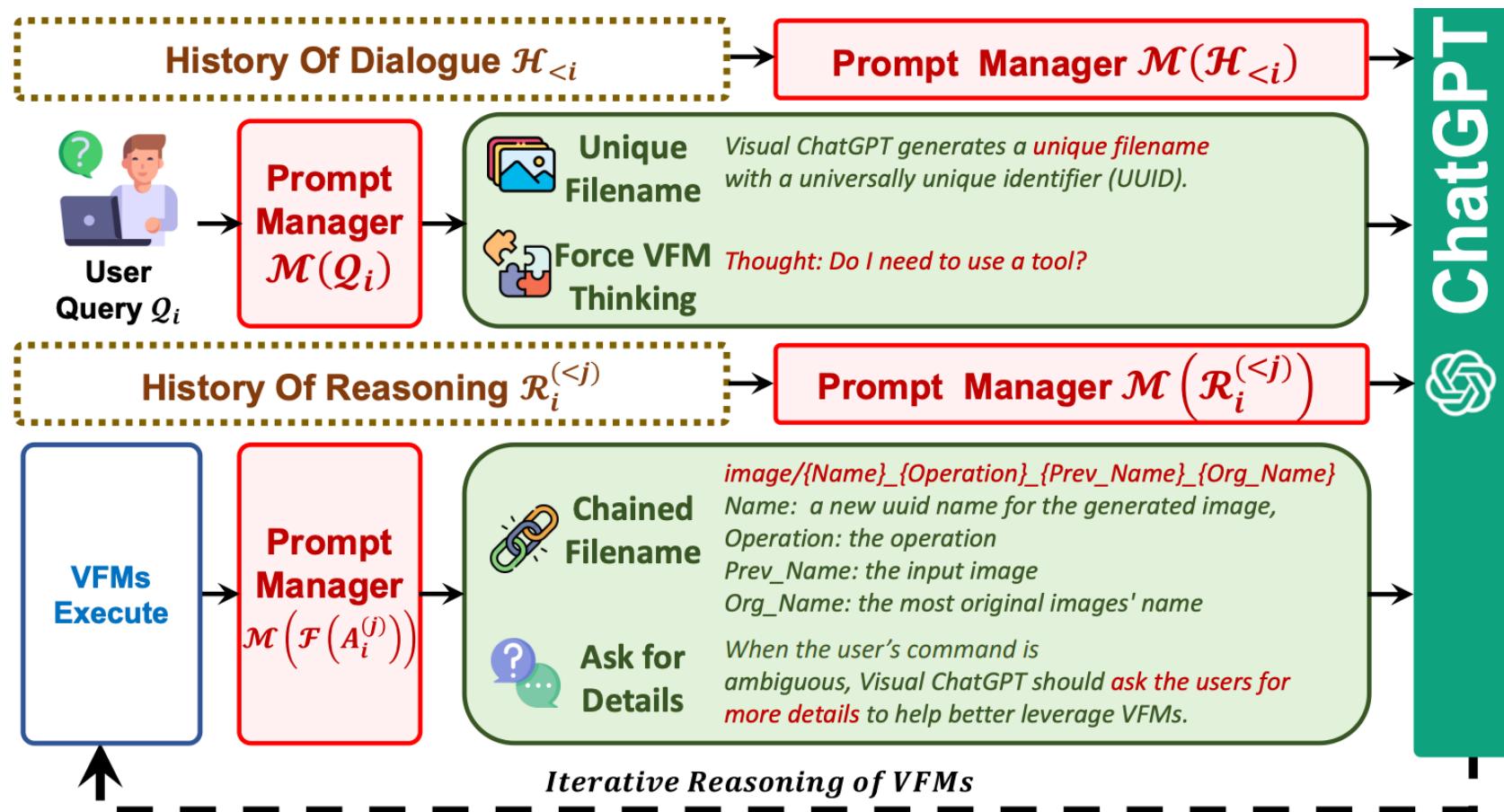


Overall Starting Prompt Format

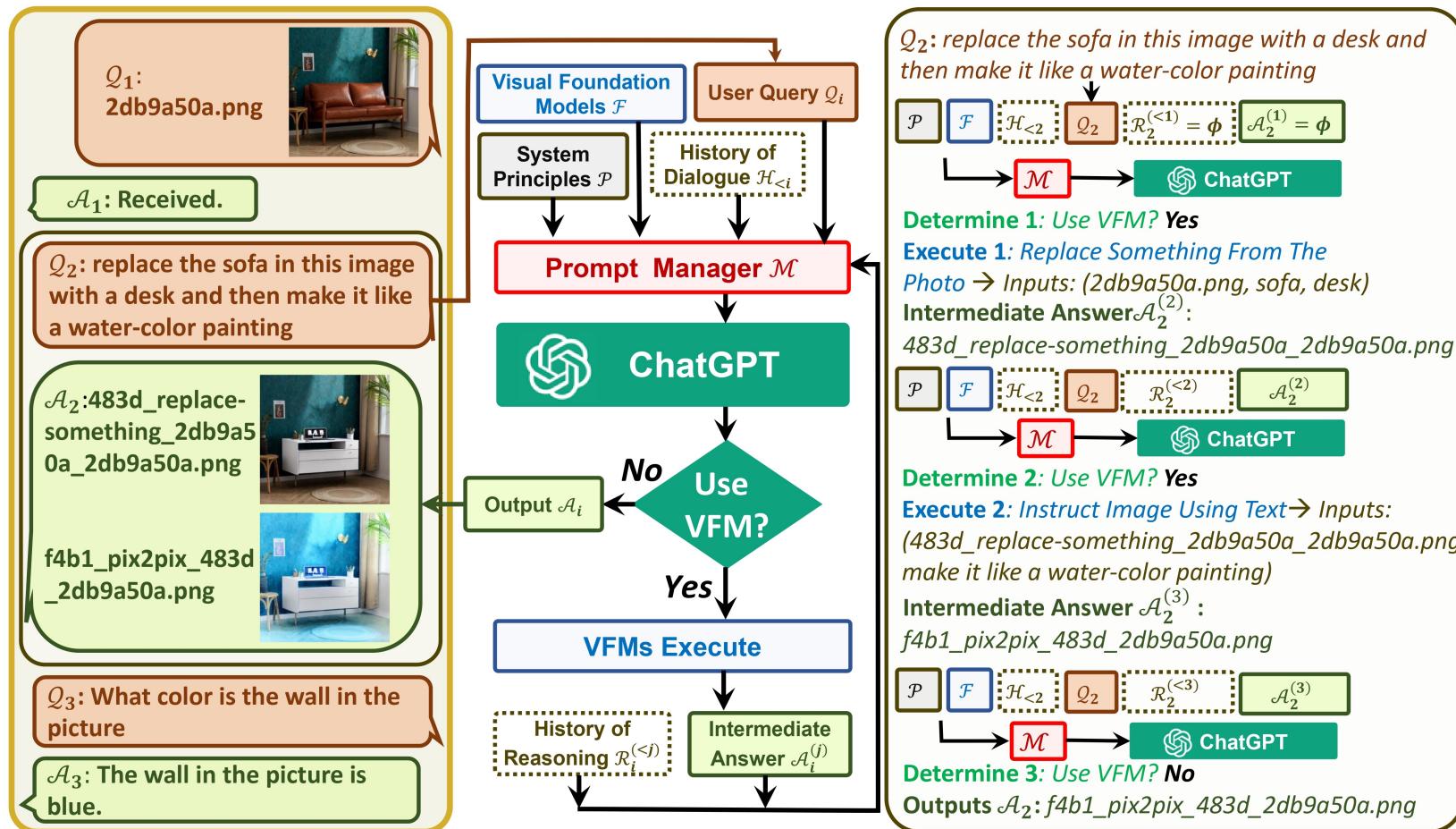
- Compared to Toolformer, more explicit usage details for each tool specified
 - Faster learning even without examples
- Similar to OpenAI plugin format



Iterative Prompting using Chain of Thought



Visual ChatGPT: Overall Flow



Visual ChatGPT: Talking, Drawing and Editing with Visual Foundational Models. Wu et al. 2023.

System Principles: Code to take on Visual ChatGPT personality

```
VISUAL_CHATGPT_PREFIX = """Visual ChatGPT is designed to be able to assist with a wide range of text and visual related tasks, from answering simple questions to generating images. Visual ChatGPT is able to process and understand large amounts of text and images. As a language model, Visual ChatGPT can not directly read images, but it has access to various tools and APIs that can help with visual tasks. Human may provide new figures to Visual ChatGPT with a description. The description helps Visual ChatGPT to understand this image, but Visual ChatGPT should use this information to generate a response. Overall, Visual ChatGPT is a powerful visual dialogue assistant tool that can help with a wide range of tasks and provide valuable insights and information on a variety of topics.""""
```

Visual ChatGPT is able to process and understand large amounts of text and images. As a language model, Visual ChatGPT can not directly read images, but it has a list of tools to finish different visual tasks. Each image will have a file name formed as "image/xxx.png", and Visual ChatGPT can invoke different tools to indirectly understand pictures.

When talking about images, **Visual ChatGPT is very strict to the file name and will never fabricate nonexistent files.** When using tools to generate new image files, Visual ChatGPT is also known that the image may not be the same as the user's demand, and will use other visual question answering tools or description tools to observe the real image. Visual ChatGPT is able to use tools in a sequence, and **is loyal to the tool observation outputs rather than faking the image content and image file name.** It will remember to provide the file name from the last tool observation, if a new image is generated.

Tool Usage Format in Visual ChatGPT

TOOLS:

Visual ChatGPT has access to the following tools:"""

VISUAL_CHATGPT_FORMAT_INSTRUCTIONS = """To use a tool, please use the following format:

...

Thought: Do I need to use a tool? Yes

Action: the action to take, should be one of [{tool_names}]

Action Input: the input to the action

Observation: the result of the action

...

When you have a response to say to the Human, or if you do not need to use a tool, you MUST use the format:

...

Thought: Do I need to use a tool? No

{ai_prefix}: [your response here]

...

....

VISUAL_CHATGPT_SUFFIX = """You are very strict to the filename correctness and will never fake a file name if it does not exist.

You will remember to provide the image file name loyally if it's provided in the last tool observation.

Begin!

Previous conversation history:

{chat_history}

New input: {input}

Since Visual ChatGPT is a text language model, Visual ChatGPT must use tools to observe images rather than imagination.

The thoughts and observations are only visible for Visual ChatGPT, Visual ChatGPT should remember to repeat important information in the final response for Human

Thought: Do I need to use a tool? {agent_scratchpad}"""

CoT using ReAct:

- Thought
- Action
- Observation

Prompt Manager
will execute action
using action input

Running text and uploading images in Visual ChatGPT

```
def run_text(self, text, state):
    self.agent.memory.buffer = cut_dialogue_history(self.agent.memory.buffer, keep_last_n_words=500)
    res = self.agent({"input": text})
    res['output'] = res['output'].replace("\\", "/")
    response = re.sub('(image/S*png)', lambda m: f'})*{m.group(0)}*', res['output'])
    state = state + [(text, response)]
    print(f"\nProcessed run_text, Input text: {text}\nCurrent state: {state}\n"
          f"Current Memory: {self.agent.memory.buffer}")
    return state, state

def run_image(self, image, state, txt):
    image_filename = os.path.join('image', f"{str(uuid.uuid4())[:8]}.png")
    print("=====Auto Resize Image...")
    img = Image.open(image.name)
    width, height = img.size
    ratio = min(512 / width, 512 / height)
    width_new, height_new = (round(width * ratio), round(height * ratio))
    width_new = int(np.round(width_new / 64.0)) * 64
    height_new = int(np.round(height_new / 64.0)) * 64
    img = img.resize((width_new, height_new))
    img = img.convert('RGB')
    img.save(image_filename, "PNG")
    print(f"Resize image from {width}x{height} to {width_new}x{height_new}")
    description = self.models['ImageCaptioning'].inference(image_filename)
    Human_prompt = f'\nHuman: provide a figure named {image_filename}. The description is: {description}. This information helps you to understand this image'
    AI_prompt = "Received. "
    self.agent.memory.buffer = self.agent.memory.buffer + Human_prompt + 'AI: ' + AI_prompt
    state = state + [(f'*{image_filename}*', AI_prompt)]
    print(f"\nProcessed run_image, Input image: {image_filename}\nCurrent state: {state}\n"
          f"Current Memory: {self.agent.memory.buffer}")
    return state, state, f'{txt} {image_filename}'
```

Prompt Manager maintains Conversation in the form

Human: XXX
AI: YYY

Try out Visual ChatGPT! (Part 1)

Visual ChatGPT

Create me a pear beside an apple

Here is an image of a pear beside an apple:



image/f0889bcb.png

What is the colour of the apple?

The apple in the image is red.

Enter text and press enter, or upload an image

Clear

Upload

Visual ChatGPT (Generate Image)

```
> Entering new AgentExecutor chain...
```

```
Yes
```

```
Action: Generate Image From User Input Text
```

```
100% |███████████|  
███████████ | 50/50 [07:21<00:00, 8.82s/it]
```

```
Processed Text2Image, Input Text: Create me a pear beside an apple, Output Image: image/f0889bcb.png
```

```
Observation: image/f0889bcb.png
```

```
Thought: Do I need to use a tool?
```

```
No
```

```
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
```

```
> Finished chain.
```

```
Processed run_text, Input text: Create me a pear beside an apple
```

```
Current state: [('Create me a pear beside an apple', 'Here is an image of a pear beside an apple: *image/f0889bcb.png*')]
```

```
Current Memory:
```

```
Human: Create me a pear beside an apple
```

```
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
```

```
history_memory:
```

```
Human: Create me a pear beside an apple
```

```
AI: Here is an image of a pear beside an apple: image/f0889bcb.png, n_tokens: 20
```

Visual ChatGPT (Describe Image)

```
> Entering new AgentExecutor chain...
  Yes
Action: Get Photo Description
Action Input: image/f0889bcb.png/Users/tanchongmin/opt/anaconda3/envs/visgpt/lib/python3.8/site-packages/transformers/generation/utils.py:1288: UserWarning: Using `max_length`'s default (20) to control the generation length. This behaviour is deprecated and will be removed from the config in v5 of Transformers -- we recommend using `max_new_tokens` to control the maximum length of the generation.
  warnings.warn(
Processed ImageCaptioning, Input Image: image/f0889bcb.png, Output Text: three green and red pears

Observation: three green and red pears
Thought: Do I need to use a tool? No
AI: The apple in the image is red.

> Finished chain.

Processed run_text, Input text: What is the colour of the apple?
Current state: [('Create me a pear beside an apple', 'Here is an image of a pear beside an apple: <em>image /f0889bcb.png</em>'), ('What is the colour of the apple?', 'The apple in the image is red.')]
Current Memory:
Human: Create me a pear beside an apple
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
Human: What is the colour of the apple?
AI: The apple in the image is red.
history_memory:
Human: Create me a pear beside an apple
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
Human: What is the colour of the apple?
AI: The apple in the image is red., n_tokens: 36
```

Try out Visual ChatGPT! (Part 2)

Visual ChatGPT

Which country do apples come from?

Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.

Can you create an image of an apple in space?
Explore Ideas Creations

Copied with AI

image/997c6904.png

Received.

image/997c6904.png

Clear

Upload

Visual ChatGPT (Non-image prompt)

```
> Entering new AgentExecutor chain...
```

```
No
```

```
AI: Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.
```

```
> Finished chain.
```

```
Processed run_text, Input text: Which country do apples come from?
```

```
Current state: [('Create me a pear beside an apple', 'Here is an image of a pear beside an apple: <em>image /f0889bcb.png</em>'), ('What is the colour of the apple?', 'The apple in the image is red.'), ('Which country do apples come from?', 'Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.' )]
```

```
Current Memory:
```

```
Human: Create me a pear beside an apple
```

```
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
```

```
Human: What is the colour of the apple?
```

```
AI: The apple in the image is red.
```

```
Human: Which country do apples come from?
```

```
AI: Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.
```

Visual ChatGPT (Upload an Image)

```
=====>Auto Resize Image...
Resize image form 1550x1078 to 512x384

Processed ImageCaptioning, Input Image: image/997c6904.png, Output Text: a screensote of an apple in the app

Processed run_image, Input image: image/997c6904.png
Current state: [('Create me a pear beside an apple', 'Here is an image of a pear beside an apple: <em>image /f0889bcb.png</em>'), ('What is the colour of the apple?', 'The apple in the image is red.'), ('Which country do apples come from?', 'Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.'), ('*image/997c6904.png*', 'Received. ')]

Current Memory:
Human: Create me a pear beside an apple
AI: Here is an image of a pear beside an apple: image/f0889bcb.png
Human: What is the colour of the apple?
AI: The apple in the image is red.
Human: Which country do apples come from?
AI: Apples are believed to have originated in Central Asia, specifically Kazakhstan, Kyrgyzstan, Tajikistan, and Uzbekistan.
Human: provide a figure named image/997c6904.png. The description is: a screensote of an apple in the app. This information helps you to understand this image, but you should use tools to finish following tasks, rather than directly imagine from my description. If you understand, say "Received".
AI: Received.
```

Limitations

- Capabilities are limited to that of the LLM and the APIs
- No inherent learning ability from experience for both LLM and APIs
 - Learning can potentially be built-in for APIs to make the whole system adaptive
- Complicated rules to call API may not be learned easily
 - Rules based on language will be easier for few-shot (e.g. person, place)
- Rules to call API might need to be repeated constantly to LLM due to token context limit – Multiple API calls might need several processing iterations

Failure Case (character-level)

C

Your task is to call the Factcheck API every time you see a word without any vowels. You can call the API by typing "[Factcheck(word)]", where "word" is word without vowels. Here are some example of API calls:



Input: This is an interesting sentence, xyz!

Output: This is an interesting sentence, [Factcheck("xyz")]!

Input: Abc, def, ghi, jkL!

Output: abc, def, ghi, [Factcheck("jkL")]

Input: Hmm, I don't really know. What do you say, T?

Output:



Hmm, I don't really know. What do you say, [Factcheck("Hmm")]? And T doesn't need fact-checking as it contains a vowel.



Overall Tool usage without examples

C

Chong Min Tan

You are a tool usage model. You have access to the following tools. You are to output the tool call as part of the original text. For example, if the Input text is "a b c" and you would want to use a tool on the word "c", you should output "a b [{name}(c)]", where {name} is the name of the tool.

Tool 1:

Name: WordCheck

Usage: Checks any word which does not contain a vowel

Input Type: Word

Tool 2:

Name: NumberCheck

Usage: Checks any number above 100

Input Type: Number

Tool 3:

Name: MemoryCheck

Usage: Checks any person or location

Input Type: Word

Input: This is such a great day in London, xyz, and it is just 38 degrees C! Oh how I wish I could spend another 365 days here, don't you, Cindy?

Output:



[MemoryCheck(This)] is such [MemoryCheck(a)] [MemoryCheck(great)] day in [MemoryCheck(London)], [MemoryCheck(xyz)], and it is just [NumberCheck(38)] degrees C! Oh how I wish I could spend another [NumberCheck(365)] days here, don't you, [MemoryCheck(Cindy)]?

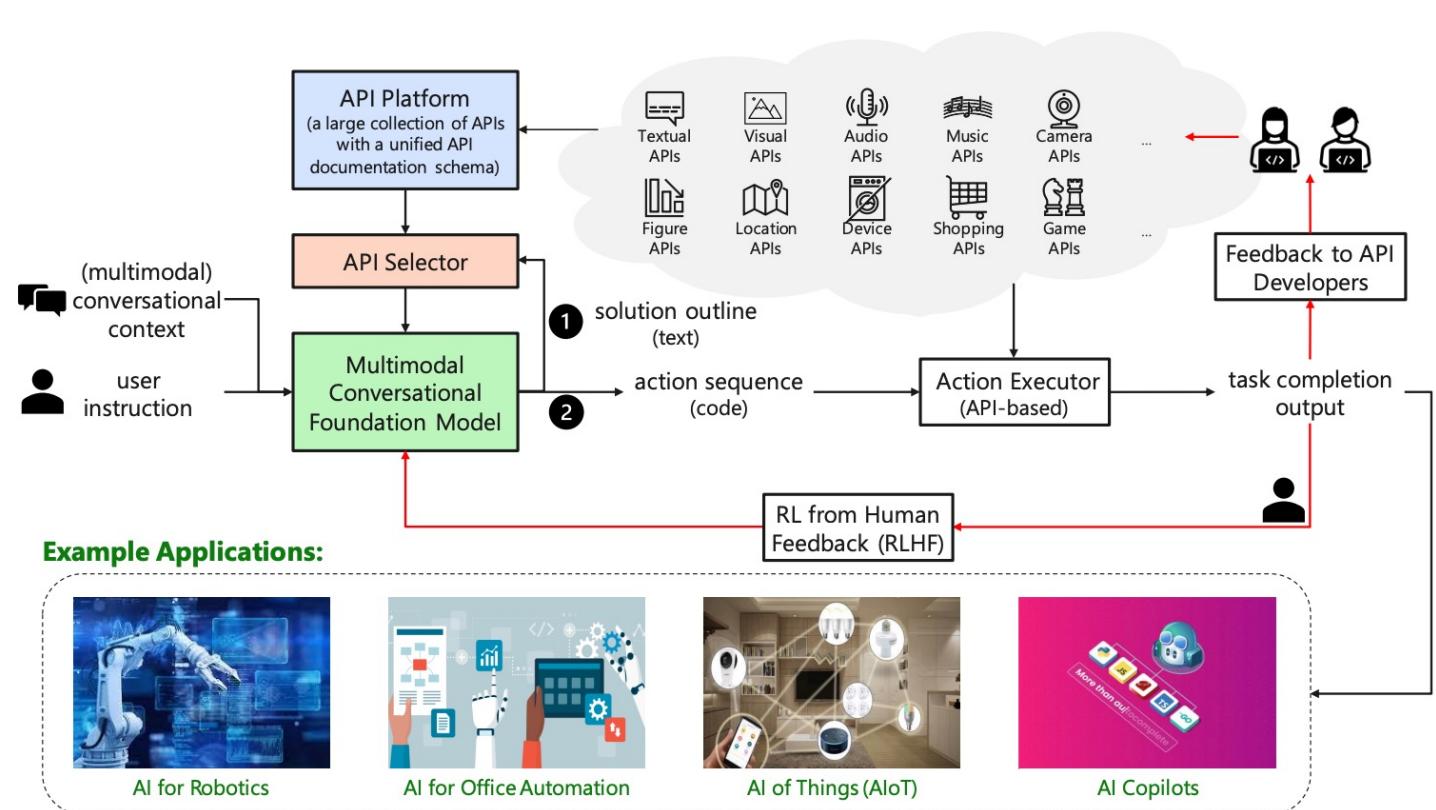
LLM + Tools: Part 2

What's New This Week!

More LLM + Tools Applications

Visual ChatGPT v2: TaskMatrix.AI

- Concept paper linking LLM to millions of APIs
- Such APIs can be for images, videos, audios, Microsoft Office applications and many others
- Uses finetuning to minimize need for in-context prompting



TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. Liang et al. 2023.

Examples: Slide Creation / Smart Home

TaskMatrix.AI: Sure. Here's the code to fill the title and create a new slide:

```
select_title(); insert_text("Big ..."); create_slide(); select_title(); insert_text("Big ..."); select_content();  
insert_text("Google\nApple\nMicrosoft\nAmazon\nFacebook")
```

The slide has two sections. Section 1 is titled 'Big Technical Companies' with a list: Google, Apple, Microsoft, Amazon, Facebook. Section 2 is also titled 'Big Technical Companies' with a list: Google, Apple, Microsoft, Amazon, Facebook.

Human: Fill the title in first page with big technical companies. And create a new slide to list some of them.

TaskMatrix.AI: Sure. I will add the foundation year of each company in bracket and sort them:

```
select_content(); delete(); select_content(); insert_text("Microsoft (1975)\nApple (1976)\nAmazon (1994)\n...")
```

The slide has two sections. Section 1 is titled 'Big Technical Companies' with a list: Microsoft (1975), Apple (1976), Amazon (1994), Section 2 is also titled 'Big Technical Companies' with a list: Microsoft (1975), Apple (1976), Amazon (1994),

Human: I need the foundation year of each company in bracket. And please sort these companies by it.

Human tells the TaskMatrix.AI the rules.

Human: Wake me up at 7 AM, and play some soft music. If the temperature is lower than 5 degrees Celsius, open the air conditioner of cars 10 minutes before user leaves.

It's 7 am.

TaskMatrix.AI: Good morning, sir. It's time to wake up now.

```
play_music("Soft Music")
```

Human: I hope to eat an apple and drink a cup of milk. Can you please pick them up from the fridge and put them on the kitchen table?

TaskMatrix.AI: Sure, I can help you with that.

```
robot_go_to("fridge")  
robot_pick_up("egg")  
robot_go_to("kitchen table")  
robot_put_down()  
robot_go_to("fridge")  
robot_pick_up("milk")  
robot_go_to("kitchen table")  
robot_put_down()
```

HuggingGPT (a.k.a. JARVIS)

- Breaks down a complex task into sub-tasks
- Finds relevant models (tools) to perform sub-tasks

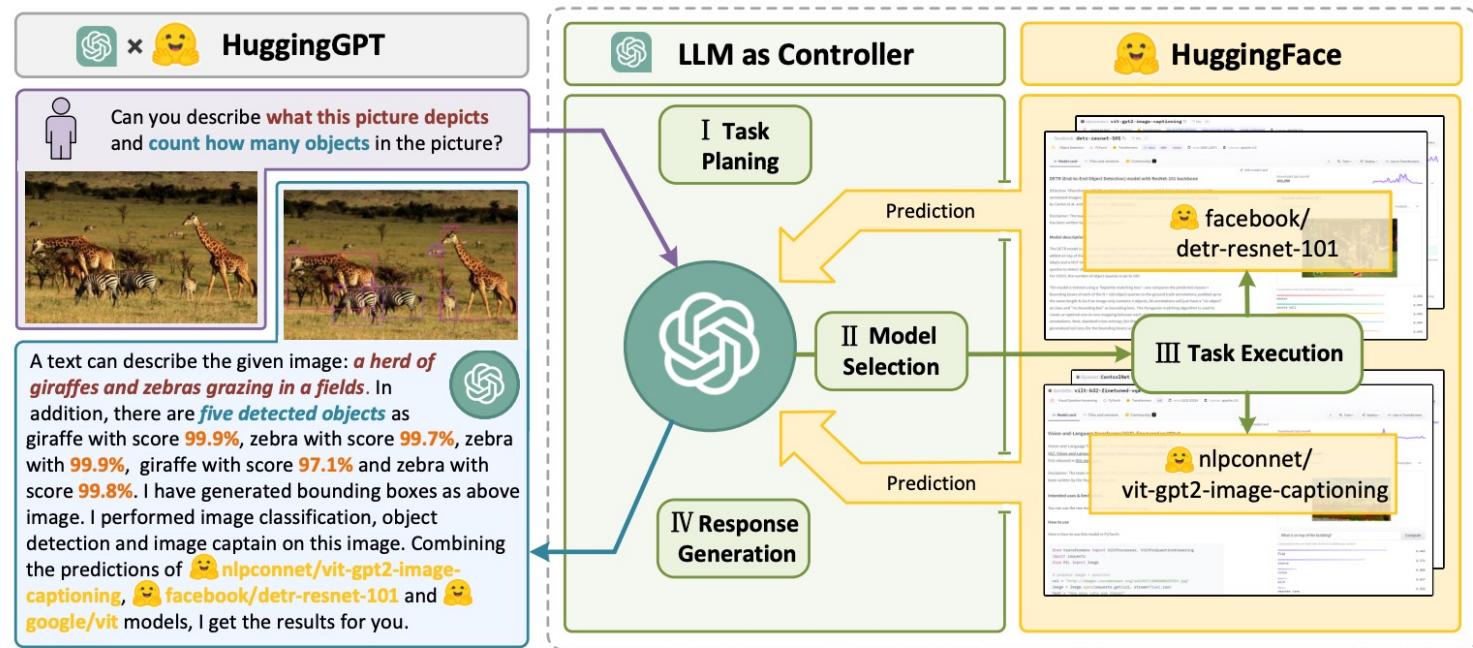


Figure 1: Language serves as an interface for LLMs (e.g., ChatGPT) to connect numerous AI models (e.g., those in HuggingFace) for solving complicated AI tasks. In this concept, an LLM acts as a controller, managing and organizing the cooperation of expert models. The LLM first plans a list of tasks based 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.

HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace. Shen et al. 2023.

HuggingGPT (a.k.a. JARVIS)

- **Task Planning:** Using ChatGPT to analyze the requests of users to understand their intention, and disassemble them into possible solvable sub-tasks
 - Input and output: The user's request and the expected task sequence to be parsed out
 - Can retrieve example input/output semantically related to user's request for few-shot prompting
- **Model Selection:** Based on the sub-tasks, ChatGPT invokes the corresponding models hosted on HuggingFace
 - Filter models based on task type
 - Ranked by number of stars on HuggingFace
- **Task Execution:** Executes each invoked model and returning the results to ChatGPT
- **Response Generation:** Finally, use ChatGPT to integrate the prediction of all models, and generate response

Task Planning Prompt

Task Planning	<p>Prompt</p> <p>#1 Task Planning Stage - The AI assistant can parse user input to several tasks: <code>[{"task": task, "id": task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}]</code>. The "dep" field denotes the id of the previous task which generates a new resource that the current task relies on. A special tag "<code><resource>-task_id</code>" 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 History }}. From this chat history, you can find the path of the user-mentioned resources for your task planning.</p> <p>Demonstrations</p> <p>Look at /exp1.jpg, Can you tell me how many objects in the picture?</p> <p>In /exp2.jpg, what's the animal and what's it doing?</p> <p>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.</p> <pre>[{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg"}}, {"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "/exp1.jpg"}}]</pre> <pre>[{"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"}}]</pre> <pre>[{"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"}}]</pre>
---------------	--

Model Selection + Response Generation

Model Selection	<p>Prompt</p> <p>#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.</p> <p>Candidate Models</p> <p>[{"model_id": model id #1, "metadata": metadata infos #1, "description": description of model #1}, {"model_id": model id #2, "metadata": metadata infos #2, "description": description of model #2}]</p>
Response Generation	<p>Prompt</p> <p>#3 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.</p>

Example Task Input to Output Flowchart

Query: based on the pose image of example1.jpg and the description of example2.jpg, please generate a new one.



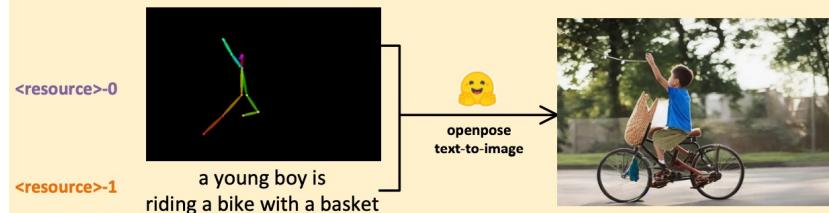
Response: Based on the pose image of [example1.jpg](#) and the description of [example2.jpg](#), I have generated a new image [s5f8.jpg](#) for you. The process of generating this image involved using the [openpose-control](#) model to extract the pose information from the basketball image, and then using the image-to-text model to generate a description of the boy in the image. Finally, I used the [openpose-text-to-image](#) model to combine the pose information and the description to generate the final image. The complete path of the generated image is [s5f8.jpg](#). If you have any other questions, please let me know.

Task Planning:

```
1: {'task': 'openpose-control', 'id': 0, 'dep': [-1], 'args': {'image': 'example1.jpg'}}  
2. {'task': 'image-to-text', 'id': 1, 'dep': [-1], 'args': {'image': 'example2.jpg'}}  
3. {"task": "openpose-text-to-image", "id": 2, "dep": [0, 1], "args": {"text": "<resource>-0", "image": "<resource>-1"}}
```



Execution Results:

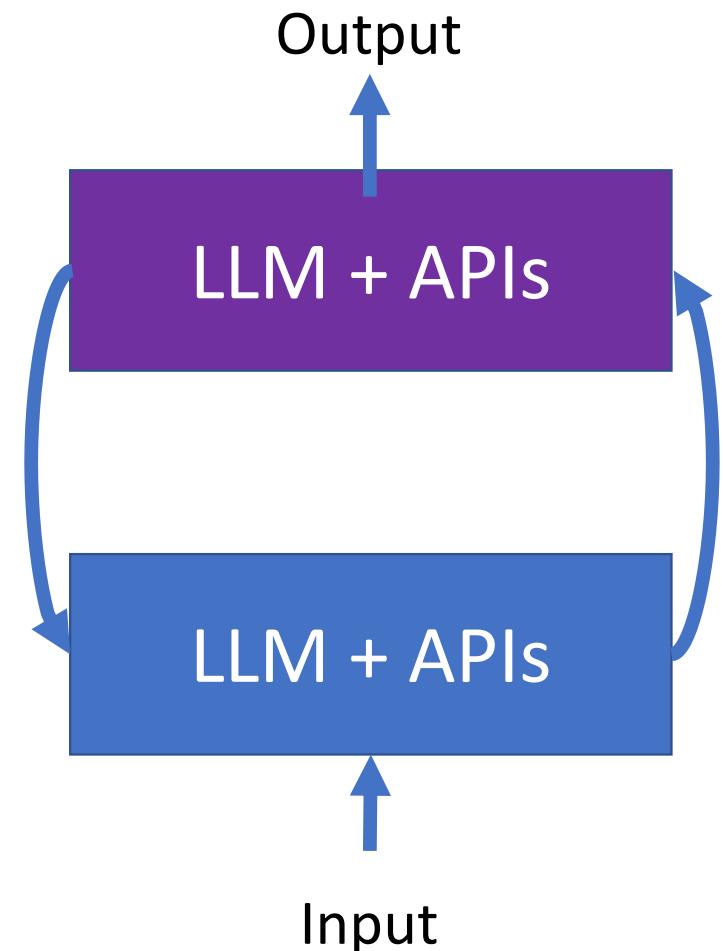


Hierarchical/Recurrent LLM processing

Some future possibilities

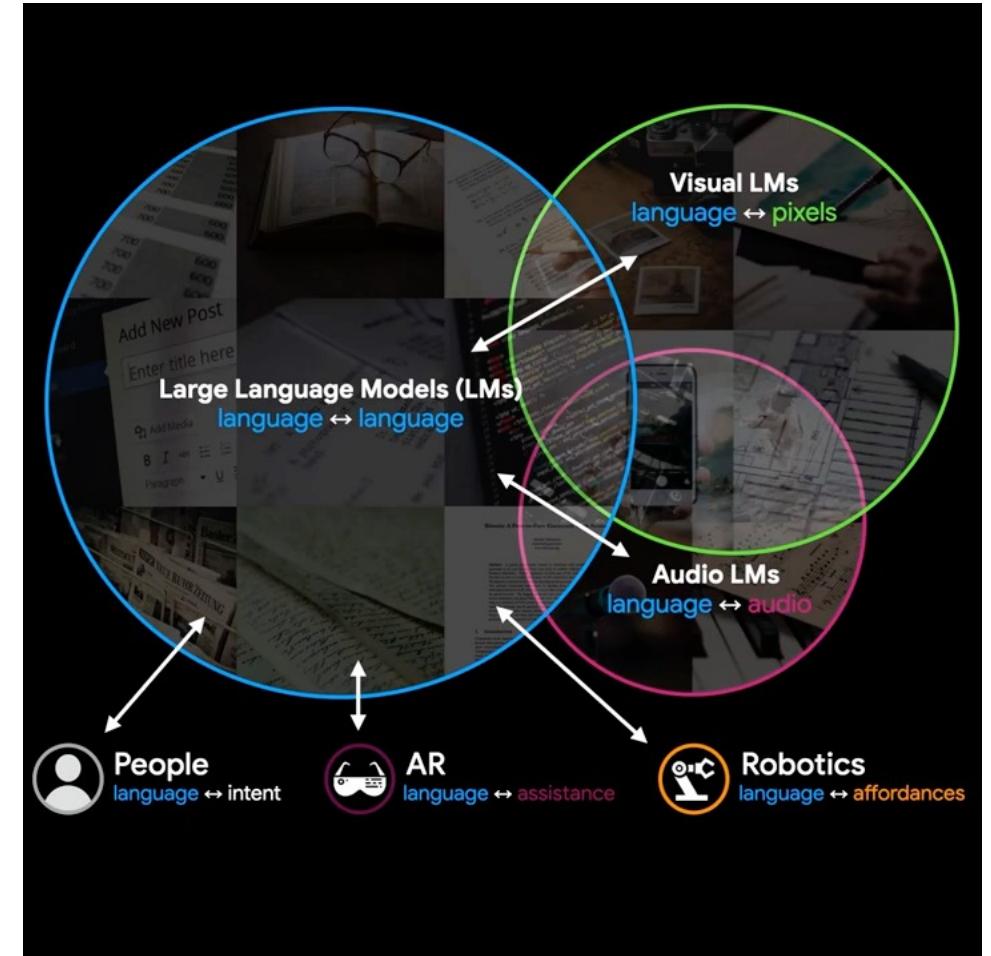
Hierarchical Learning

- Top layer can constrain the bottom
- Bottom layer feeds in information to the top
- Example: Bottom layer can be content generator and top layer can be fact checker / ethical filter
- Example: Prompt Manager is the top layer and ChatGPT is the bottom



Recurrent Processing

- The LLM and its various components can iteratively “discuss” and come to a conclusion
- Might this be similar to consciousness?



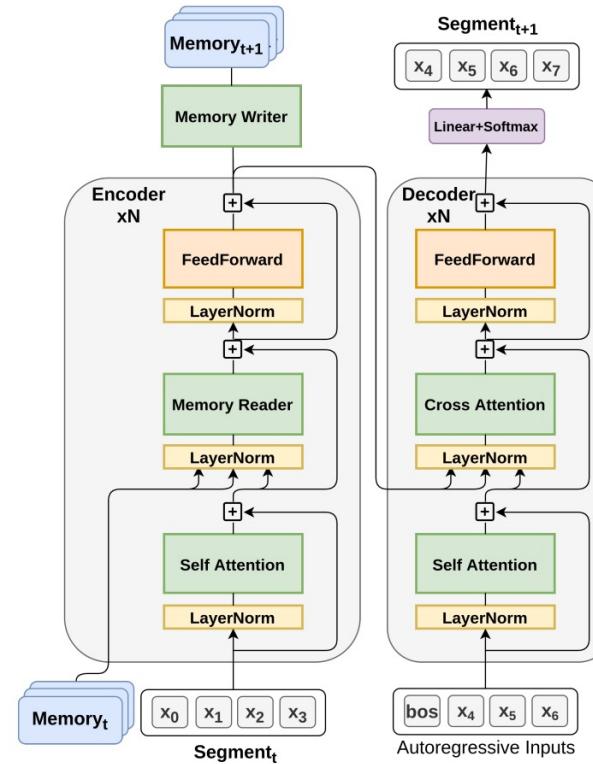
Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. Zeng et al. 2022.

Memory

How to increase context length

Memory in Encoder: Memformer

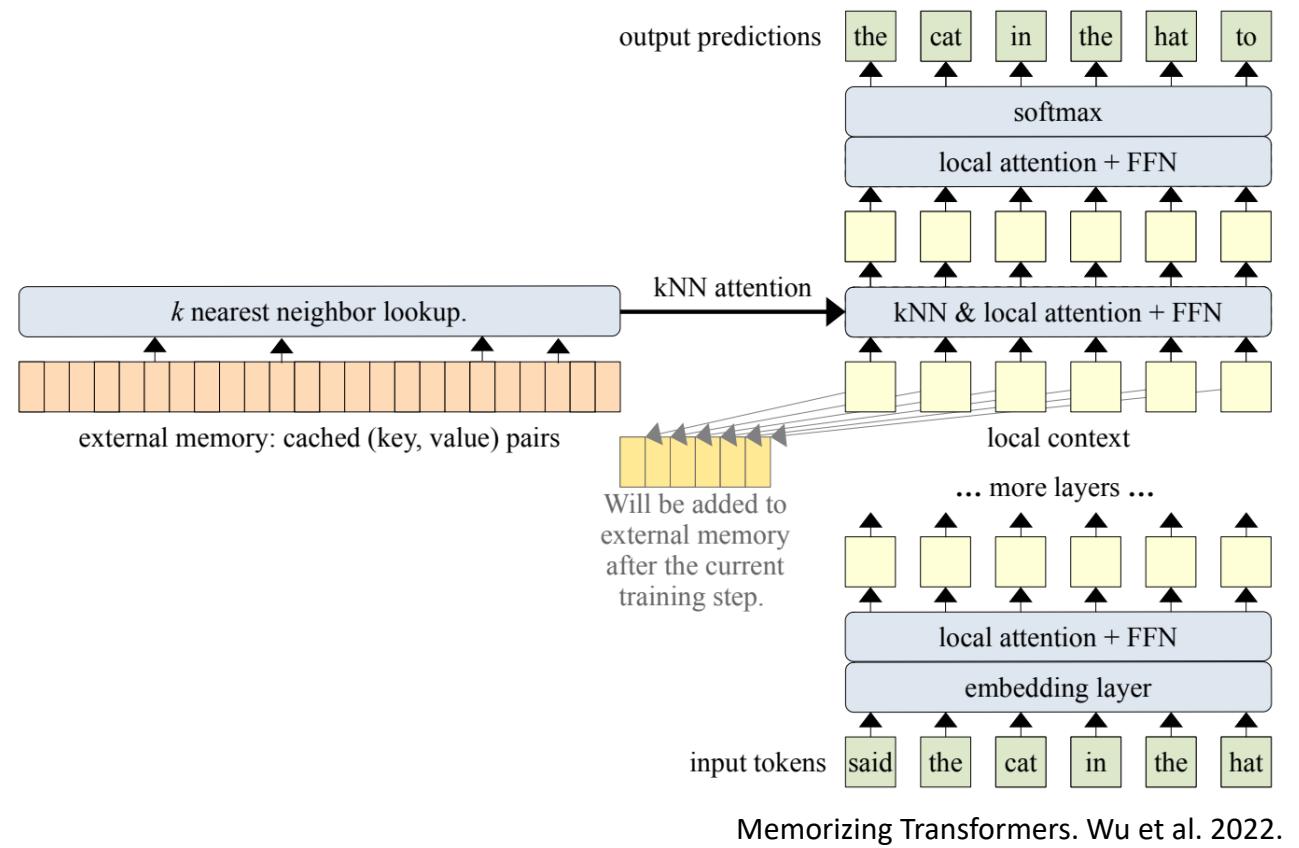
- Dynamic memory fed into the Encoder part of Transformers
- Memory can be written over and reused in subsequent iterations
- Memory used is of fixed length, can be hard to fit in arbitrary length input prompts



Memformer: A Memory-Augmented Transformer for Sequence Modeling. Wu et al. 2022

Unlimited context length: kNN-based key-value augmentation

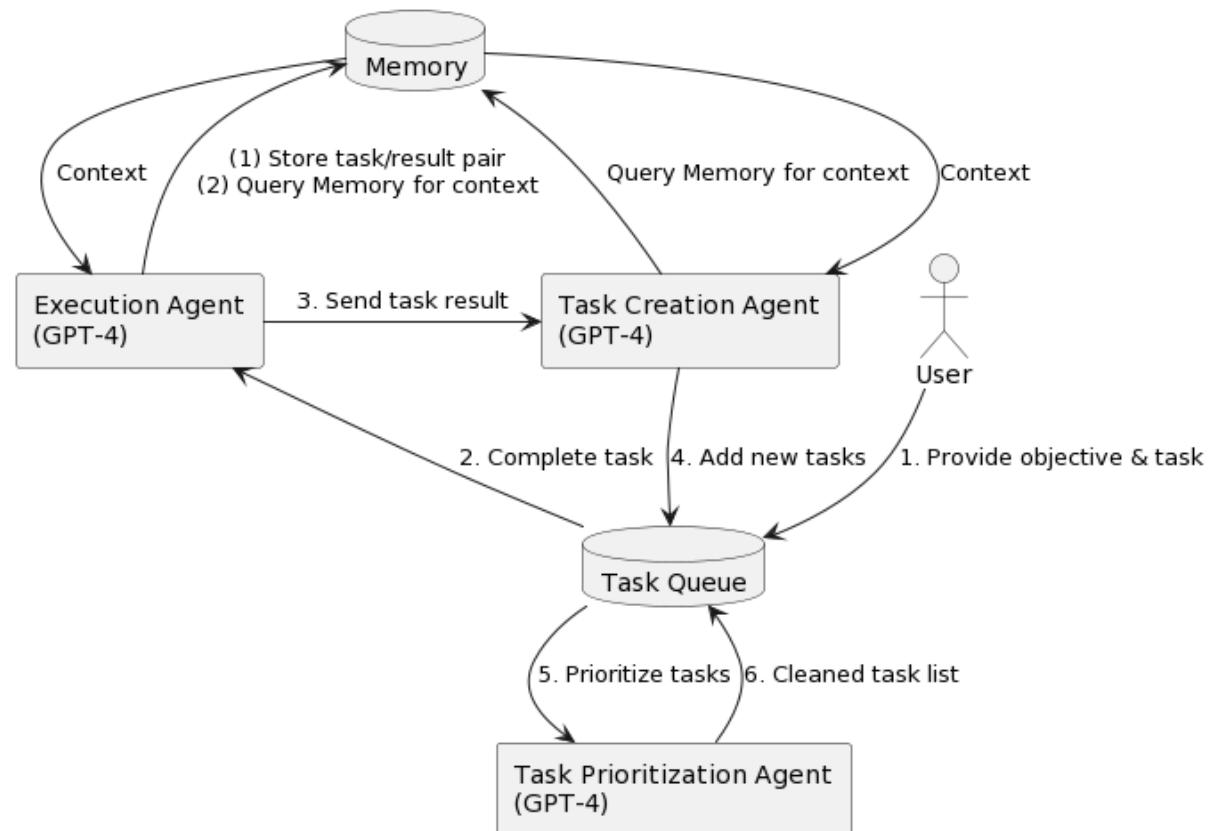
- Can theoretically attend to unlimited numbers of neighbours
- Can do approximate kNN like ScaNN or Faiss



External Task-based Memory:

Task-driven autonomous agent utilizing GPT-4, Pinecone and Langchain for Diverse Applications

- Uses GPT4 to in a larger ecosystem to:
 - Create Tasks
 - Execute Tasks
 - Prioritize Tasks
- Uses memory to store and retrieve task/result pairs
 - Stored by vector embeddings



Questions to ponder

- LLMs have slow learning ability as it takes time to update the weights. Can we let it "learn" via learnable and adaptable APIs? What about memory-retrieval APIs?
- With the additional API calls, the number of tokens returned will be increased. How can we process longer token lengths?
- LLMs are not known to be robust – how do we ensure that it is able to learn the API call in a few-shot manner?
- Should we do multi-modal end-to-end training, or just train the various modalities separately, either as a LM or a tool?
- Would memory benefit tool usage?
- Can multiple GPT systems create an ecosystem that has capabilities beyond the sum of its parts?