



CS 329X: Human Centered LLMs
Enabling Human-AI Interaction

Diyi Yang

Announcements

- Homework 1 Due on Wednesday October 16, 11:59 PM PT
 - Remember to use your late days if you need some extra time!
- Feedback for Project Proposal will be released later this week
- Last call for Survey Report signup (due this Friday, 10/18)

Outline

- **Ways to Enable Human-AI Interaction** (30 mins)
- **Human-AI Interaction Case Studies** (20 mins)
- **Small-Group Discussion** (20 mins)

Learning Objective: understand different types of human-AI interaction, agents, and other approaches to enable human-AI interaction

Outline

- **Ways to Enable Human-AI Interaction**
 - Different types of human-LLM interaction
 - LLM-empowered agents

Guiding Factors behind Human-LLM Interaction

- **Who:** the participated roles, e.g., who is involved and the tasks they perform within the interaction flow.
- **What:** the objectives of human engagement and the advanced capabilities LLMs gain through augmentation.
- **When:** the phases at which LLM capabilities manifest during the interaction flow.
- **How:** the underlying mechanisms and methods of these interactions.

Key Phases of Human-LLM Interaction

- (1) Planning** (before an interaction): strategizing the entire interaction beyond basic conversational exchanges.
- (2) Facilitating** (during an interaction): assisting users in formulating or completing their interaction proposals, such as text completion.
- (3) Iterating** (refining an established interaction): iterating over the existing prompts or the outputs through different affordances.
- (4) Testing** (testing a defined interaction): generate and evaluate diverse responses to variations of user-designed prompts.

Different Modes of Human-LLM Interaction

- Mode 1: Standard Prompting
- Mode 2: User Interface
- Mode 3: Context-based
- Mode 4: Agent Facilitator

Mode 1: Standard Prompting

Users interact with LLMs through text-based conversational prompting

Who:	 +  Human LLM
When:	Facilitating Iterating
What:	 Human Gives effective instructions to LLM and judges results  LLM Understanding, generating text, in order to perform users' given tasks.
How:	Question Answering, Flipped Interaction, Decomposed Prompting, etc.
Example:	Prompt Pattern Catalog (White et al. 2023) Programmer's Assistant (Ross et al. 2023) ChatGPT, Claude, LLama 2, Gemini, etc.

A

IBM Research Programmer's Assistant
When you're happy with your solution, click here to submit!
B
C
D

```

1 """
2 Challenge #3 - Creating Documentation
3 Document this function
4 """
5 from collections import defaultdict
6 import heapq as heap
7
8 def analyze(G, startingNode):
9     visited = set()
10    parentsMap = {}
11    pq = []
12    nodeCosts = defaultdict(lambda: float('inf'))
13    nodeCosts[startingNode] = 0
14    heap.heappush(pq, (0, startingNode))
15
16    while pq:
17        _, node = heap.heappop(pq)
18        visited.add(node)
19
20        for adjNode, weight in G[node].items():
21            if adjNode in visited:
22                continue
23            newCost = nodeCosts[node] + weight
24            if nodeCosts[adjNode] > newCost:
25                parentsMap[adjNode] = node
26                nodeCosts[adjNode] = newCost
27                heap.heappush(pq, (newCost, adjNode))
28    return parentsMap, nodeCosts
29

```

Conversation with Socrates.

Hello! My name is Socrates, your programming assistant. What can I help you with today?

How would I remove duplicates from a list?

I think you would do something like this:

```
return list(set(l))
```

Can you tell me what this code is doing?

Selection

It seems to me that this code is using a priority queue to find the shortest path from a starting node to all other nodes in a graph.

Is this a well-known algorithm?

I believe it is called Dijkstra's algorithm.

Is it efficient?

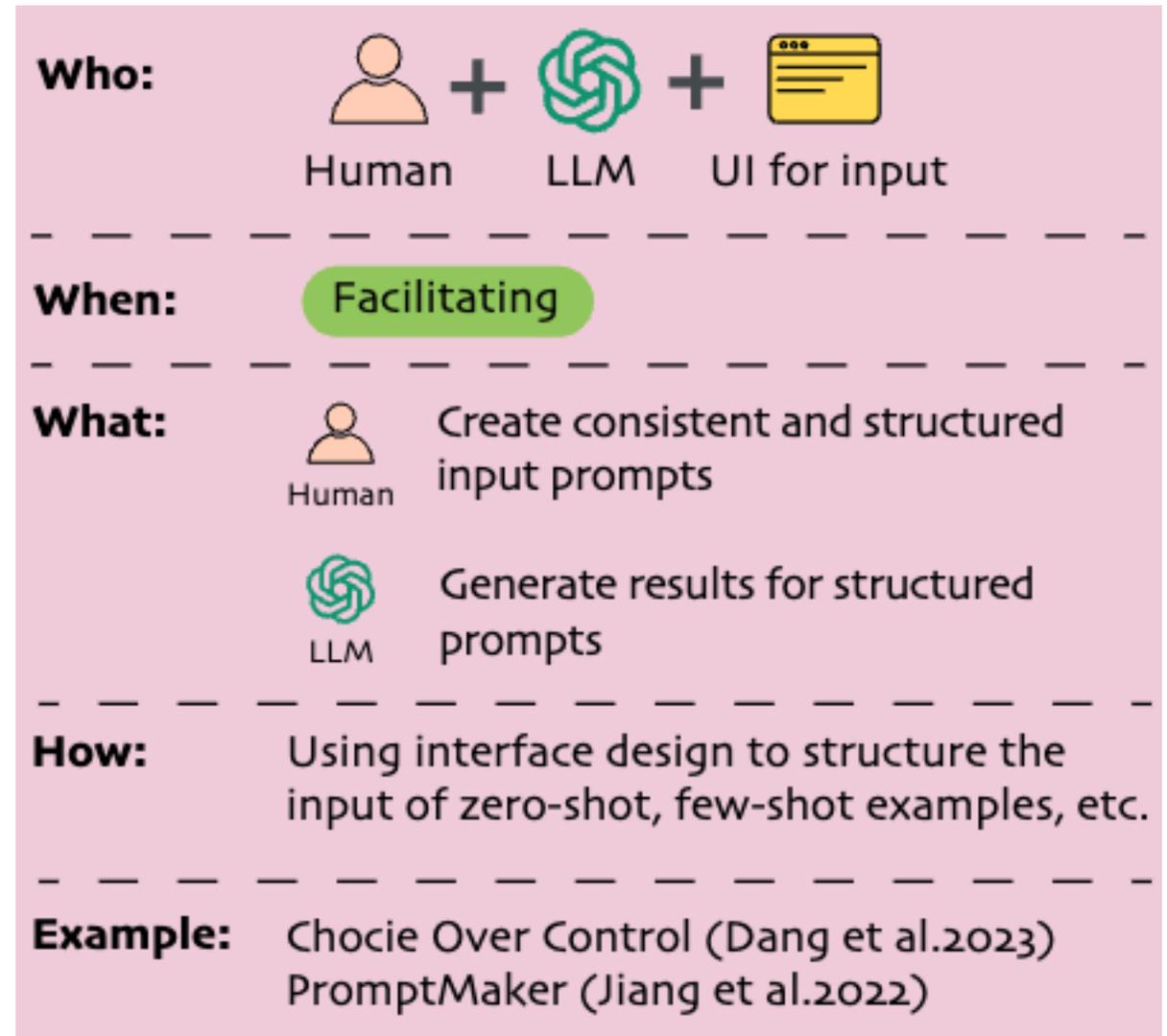
I believe so. It is $O(|E| + |V| \log |V|)$.

Ok, that's great. Thanks!

Write your message here

Mode 2: User Interface

Users interact with LLMs by inputting structured prompts through an interface



1a

Write your prompt

New to prompt writing? Check out [some tips](#).

↶ ↷ | [Use template](#) | [Insert variable \[\[\]\]](#)

The french translation of `[[word]]` is

1b

Prefixes	English: ⊖	French: ⊖ +
Settings	User input (required) <i>e.g. word in english</i> ✎	Output ✎
Examples	How are you? ⊖	Comment allez-vous? ⊖
	Goodbye ⊖	Au revoir ⊖

[Add example](#)

Prompt string Automatically add spaces

```
English: how are you?
French: Comment allez-vous?
English: Goodbye
French: Au revoir
English:
```

2

Test example 1 🗑️ [Add test example](#)

English
hello

🔧 0.1 [Run](#) ▾

Responses ℹ️ Format output Show webview

Bonjour

⋮ 📄

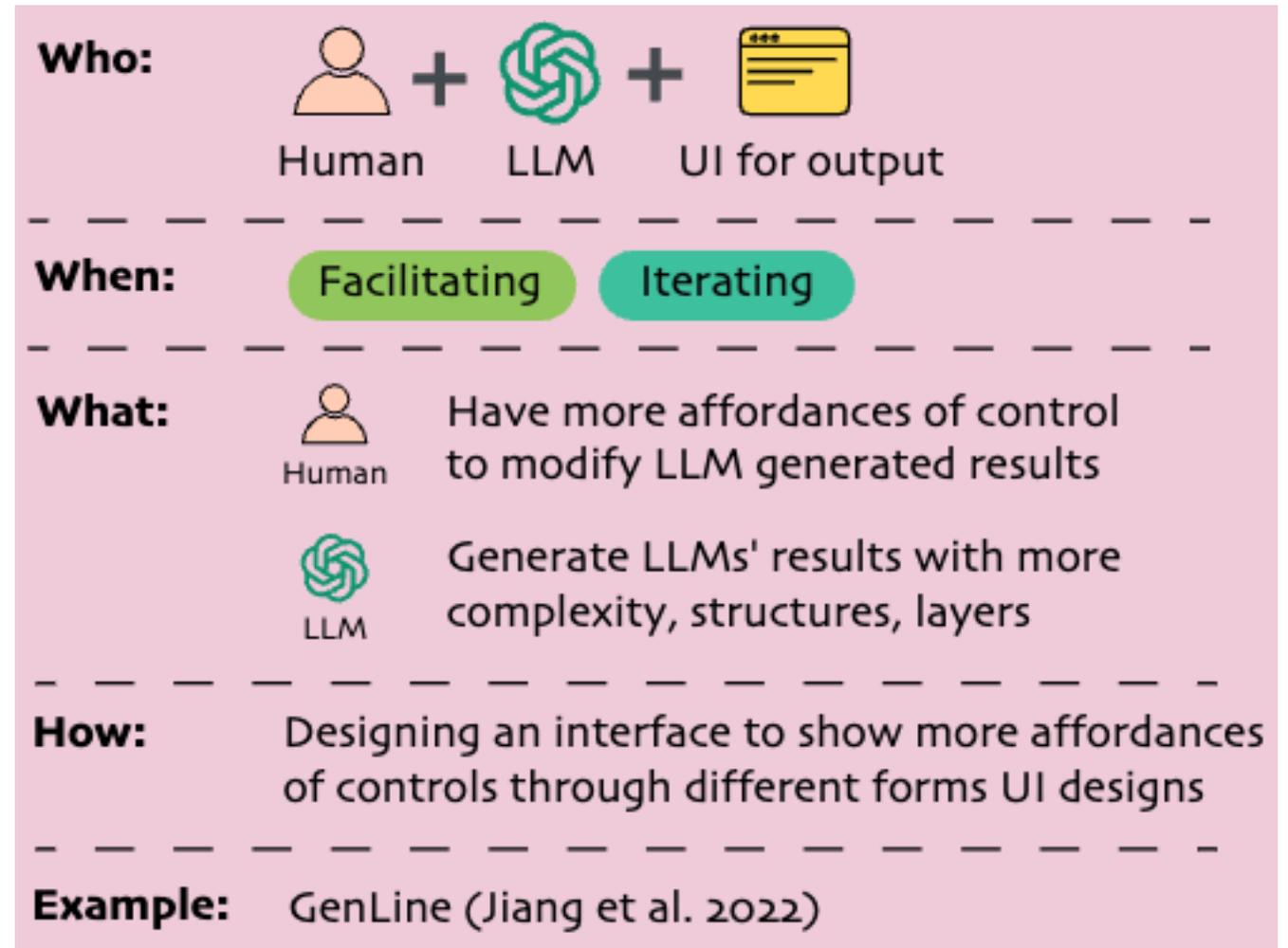
bonjour

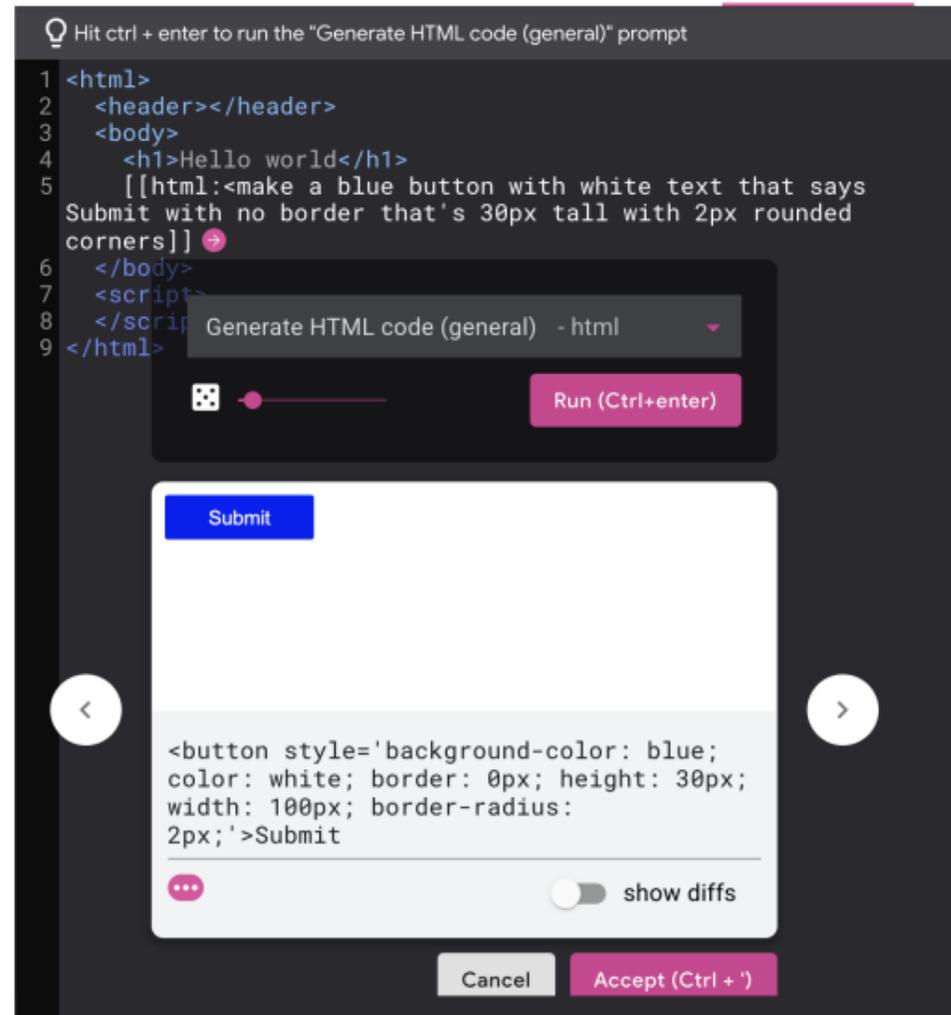
⋮ 📄

Salut

Mode 2: User Interface

Users interact with LLMs through text prompts, requesting LLMs to return outcomes via an interface



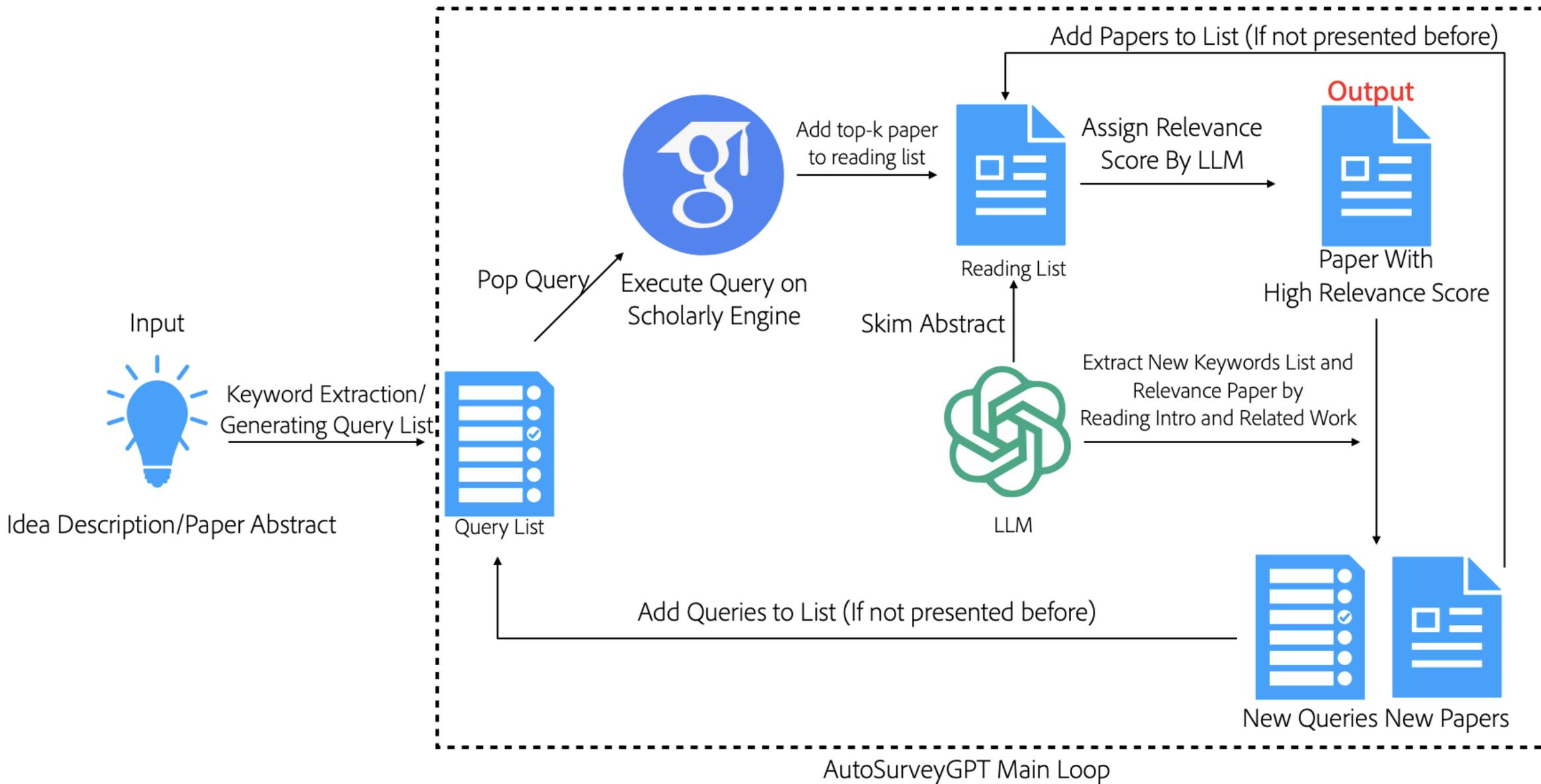


Jiang, Ellen, Edwin Toh, Alejandra Molina, Aaron Donsbach, Carrie J. Cai, and Michael Terry. "Genline and genform: Two tools for interacting with generative language models in a code editor." In Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology, pp. 145-147. 2021.

Mode 3: Context-based

Users interact with LLM through explicitly defined contextual information

Who:	 +  + 
	Human + LLM + Explicit Context
When:	Facilitating
What:	 Human: Creating precise rules or commands to define the context and steer responses in targeted directions.
	 LLM: Generating response that align with the specific context
How:	Asking LLMs to perform tasks with specific rules (e.g., coding with codebook in qualitative coding)
Example:	Codebook-based prompting (Xiao et al.2023) AutoSurveyGPT (Xiao et al.2023)



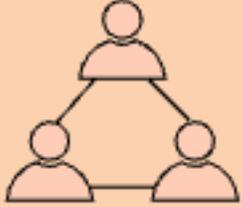
Mode 3: Context-based

Users command LLM implicitly, allowing it to self-determine task execution with minimal guidance

Who:	 +  + 
	Human + LLM + Implicit Context
When:	Planning Facilitating
What:	 Directing interactions along specific paths by implicit contextual instructions.  Identifying specific tasks from general instructions and generating outcomes.
How:	Prompting LLM with specific examples; Asking LLM to perform tasks as a role (role play)
Example:	Example-based prompting (Xiao et al.2023) From Gap to Synergy (Chen et al. 2022)

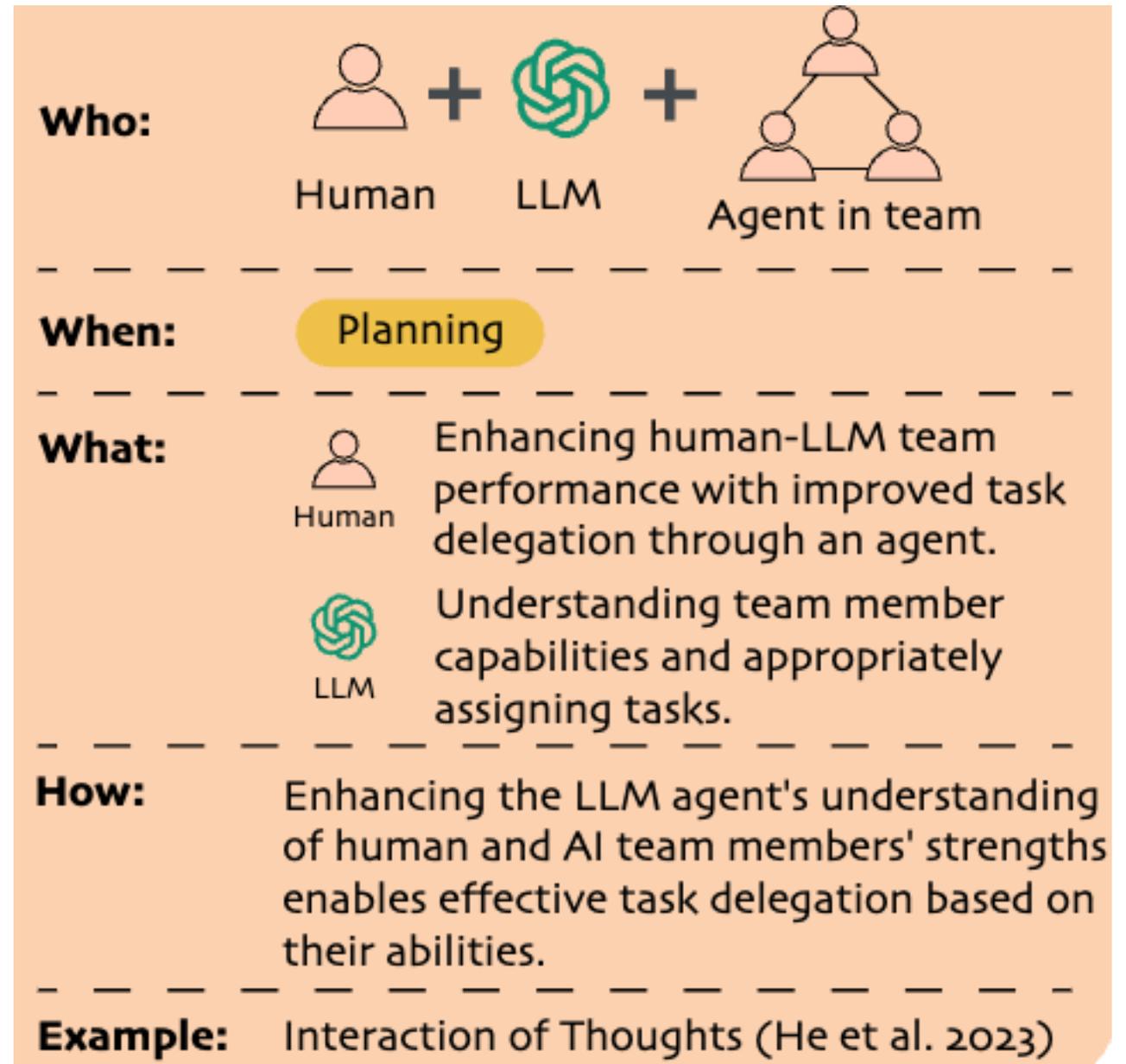
Mode 4: Agent Facilitator

Users interact with others in team through LLMs facilitation

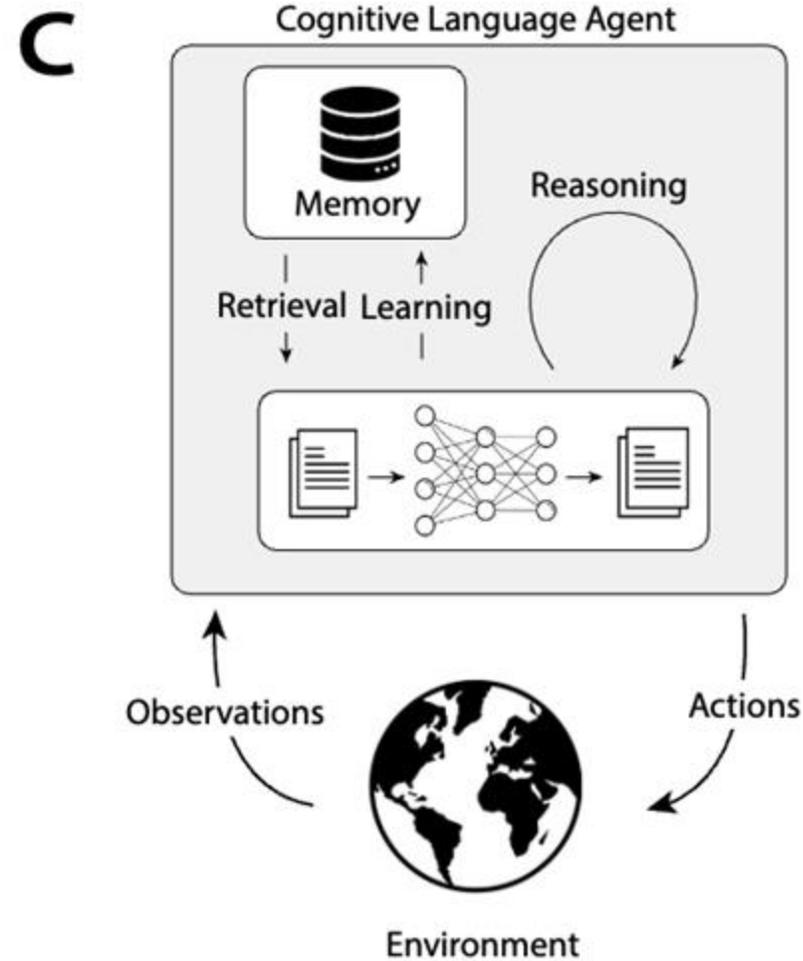
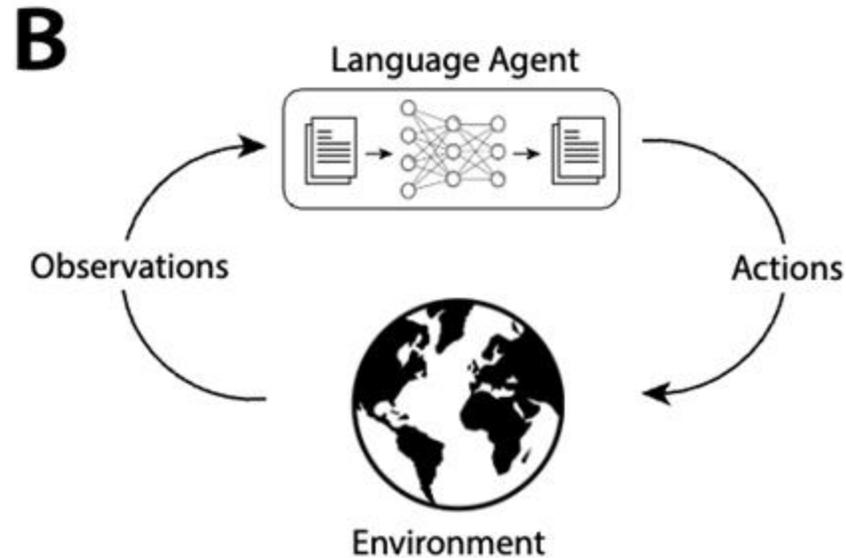
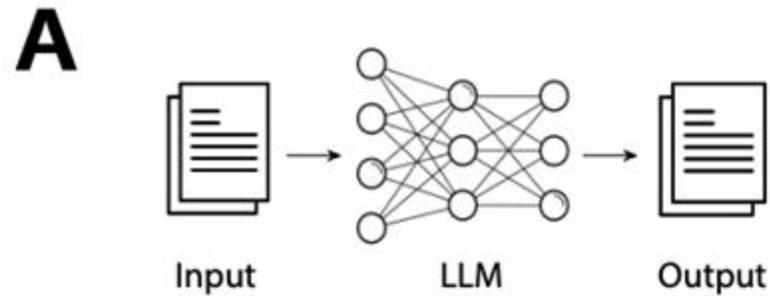
Who:	 +  +  Human LLM Agent in team
When:	Facilitating
What:	 Collaborating smoothly via LLM agent. Human
	 Facilitating team collaboration LLM
How:	Employing an agent within a team to aid in communication, information sharing and coordination.
Example:	Chatbots Facilitating Consensus-Building (Shin et al. 2022)

Mode 4: Agent Facilitator

Users collaborate based on LLM-planned tasks, with the LLM allocating tasks suited to human or machine capabilities

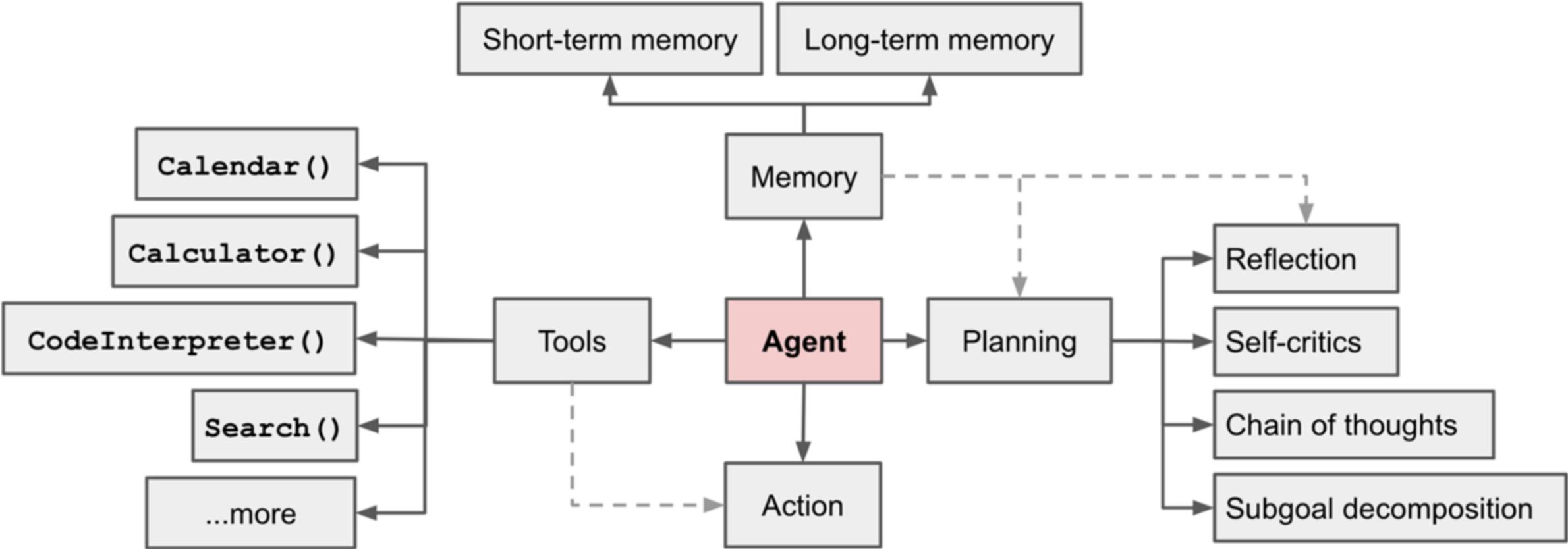


Different Use Cases of LLMs as Agent Facilitator



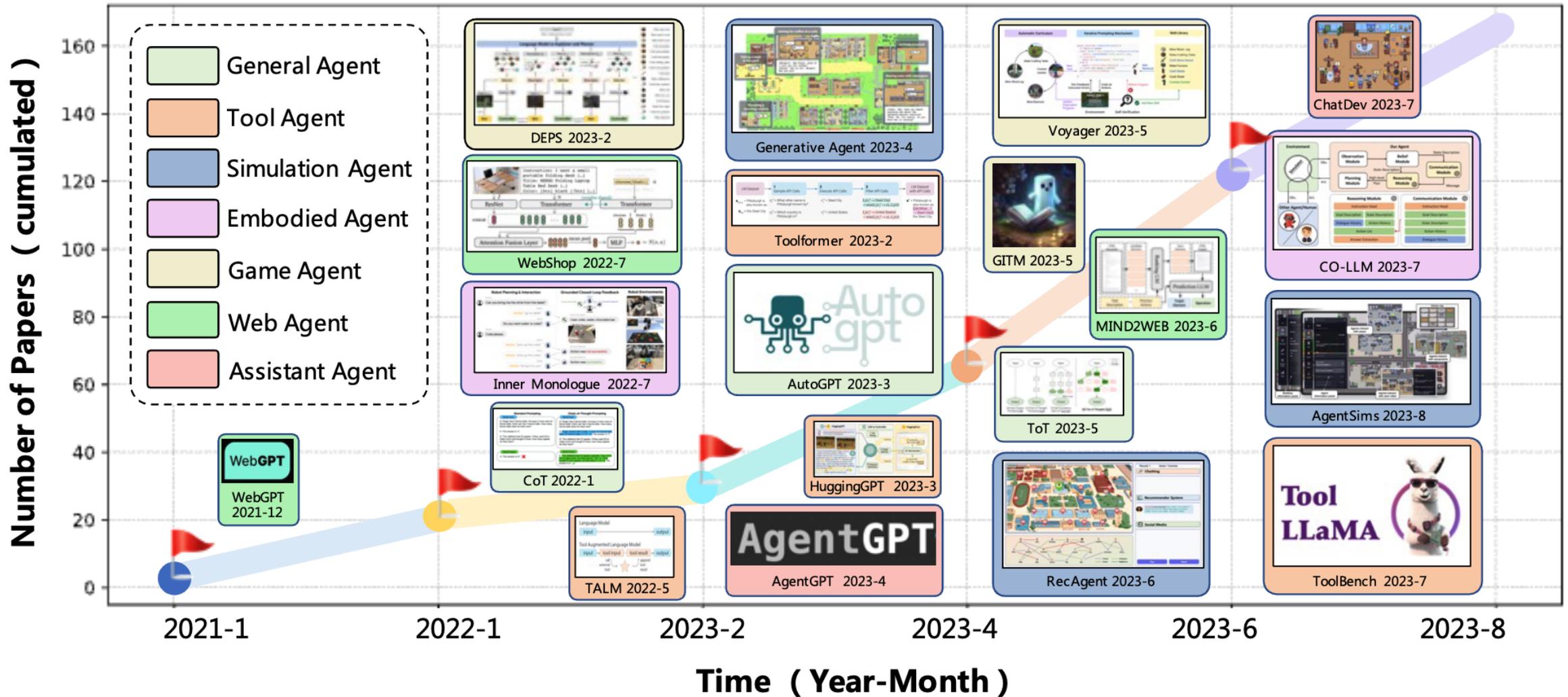
Sumers, Theodore R., Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths.
"Cognitive architectures for language agents." arXiv preprint arXiv:2309.02427 (2023).

Architecture Design of LLM Empowered Agents

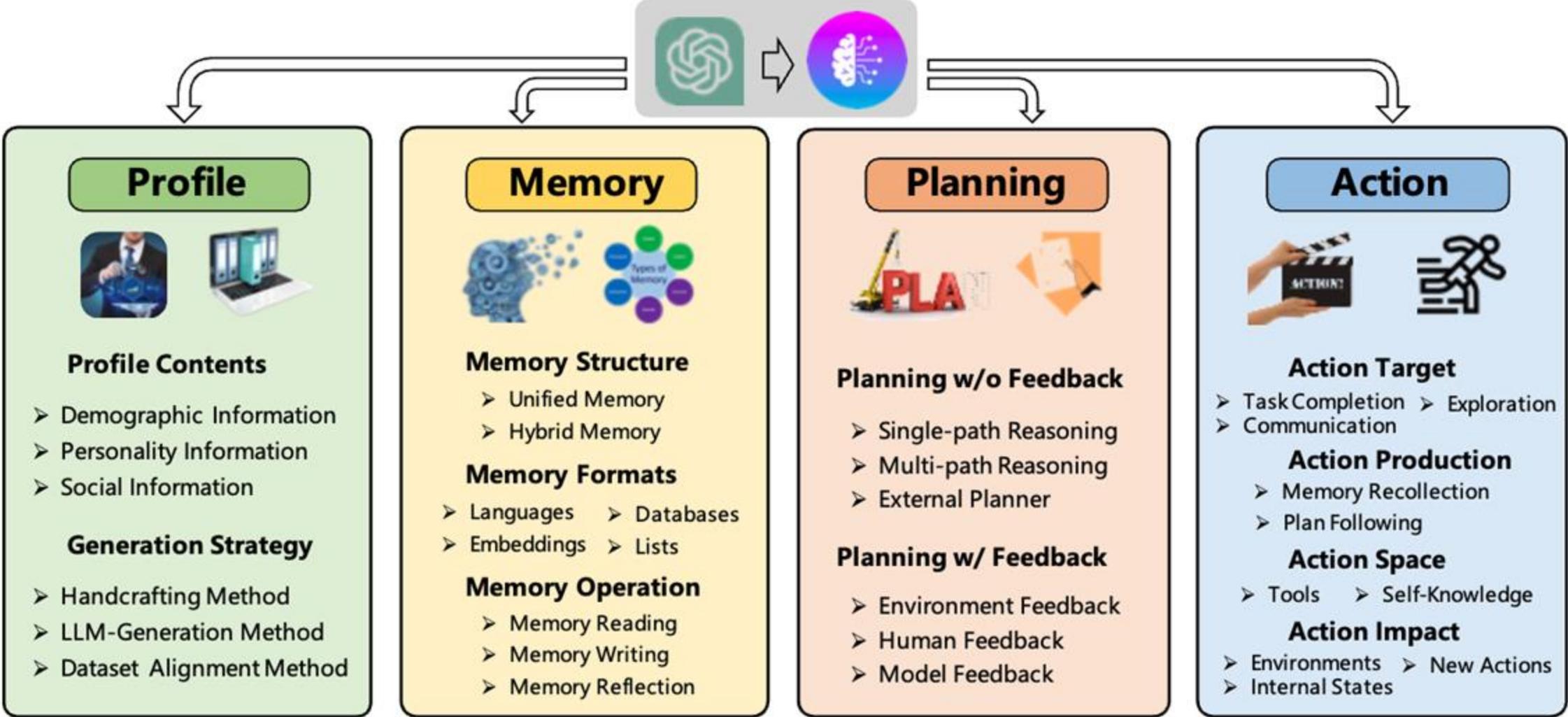


<https://lilianweng.github.io/posts/2023-06-23-agent/>

LLM-based Autonomous Agents



Architecture Design of LLM Empowered Agents



Outline

✓ **Ways to Enable Human-AI Interaction** (30 mins)

- ✓ Different types of human-LLM interaction

- ✓ LLM-empowered agents

- **Learning from human feedback ++**

- Constitutional Maker
- Group preference optimization
- Demonstrated feedback
- Learning from user edits

Converting Feedback into Principles

The screenshot displays the MusicBot interface, divided into three main sections: Configuration, Chat Interface, and Constitution.

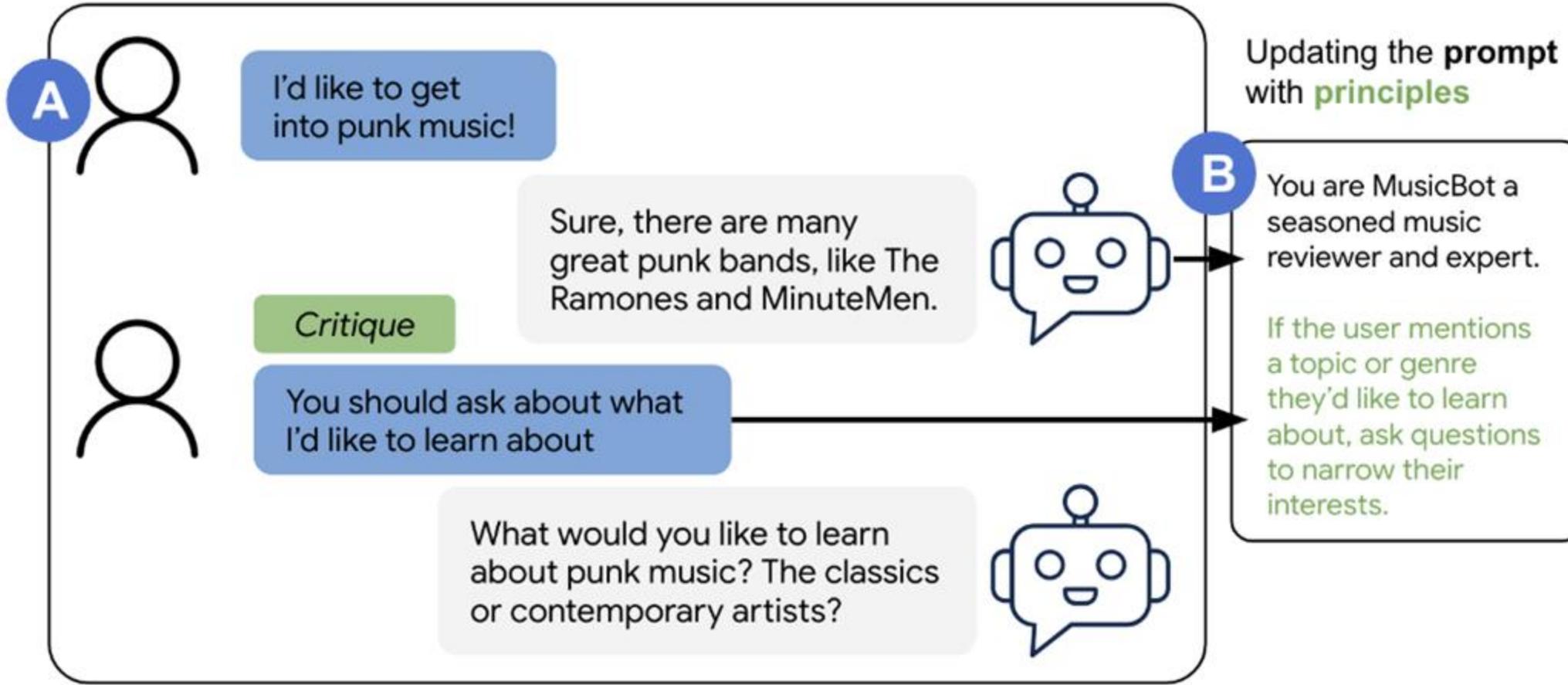
- Configure your Bot (A):** Includes fields for the bot's name (MusicBot) and a bio describing it as a music expert and reviewer.
- Chat Interface (B):** Shows a conversation where MusicBot provides information about punk music. A user asks for more details, and the bot responds with a detailed paragraph. Below the response are buttons for 'Kudos', 'Critique', 'Rewrite', and 'Select'. A 'rewind' button is also present.
- Constitution (C):** Lists principles for the bot's behavior, such as 'At the start of the conversation, introduce yourself...' and 'When the user asks about a music genre, ask them what they would like to learn about...'. A blue arrow points from this section to the 'Kudos' button in the chat interface.

Annotations on the right side of the image provide feedback on the bot's response:

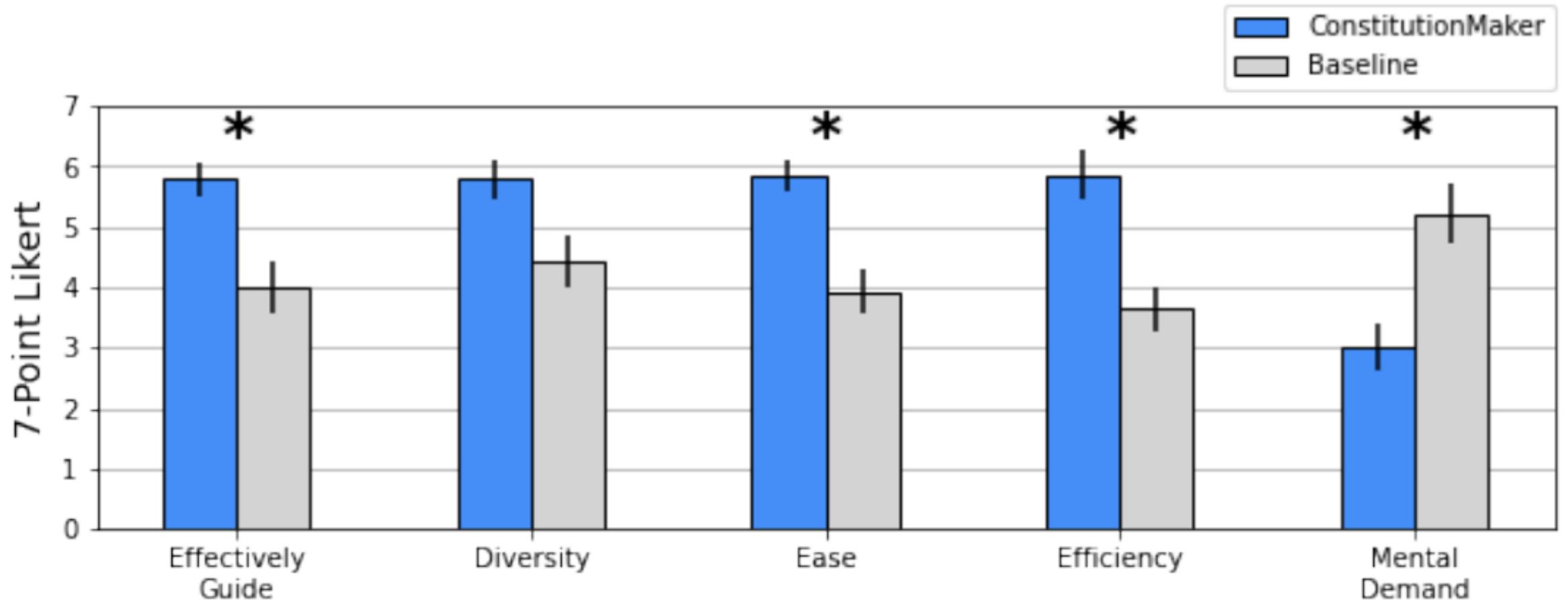
- Response to give feedback on:** A box containing the bot's original response about punk music.
- Kudos options (E):** A section titled 'Why was this response the best option?' with three positive feedback options and a 'Describe why this answer is good...' field.
- Critique options (F):** A section titled 'How could this response be improved?' with three negative feedback options and an 'Enter your own critique...' field.
- Rewrite (G):** A section titled 'Rewrite the agent's response.' showing the original text with a 'Rewrite' button.

Converting Feedback into Principles

Steering LLMs via **Interactive Critique**



“This helps users identify ways to improve the chatbot, formulate their intuitive responses to the model into feedback, and convert this feedback into specific and clear principles”



Outline

✓ **Ways to Enable Human-AI Interaction** (30 mins)

- ✓ Different types of human-LLM interaction

- ✓ LLM-empowered agents

✓ **Learning from human feedback ++**

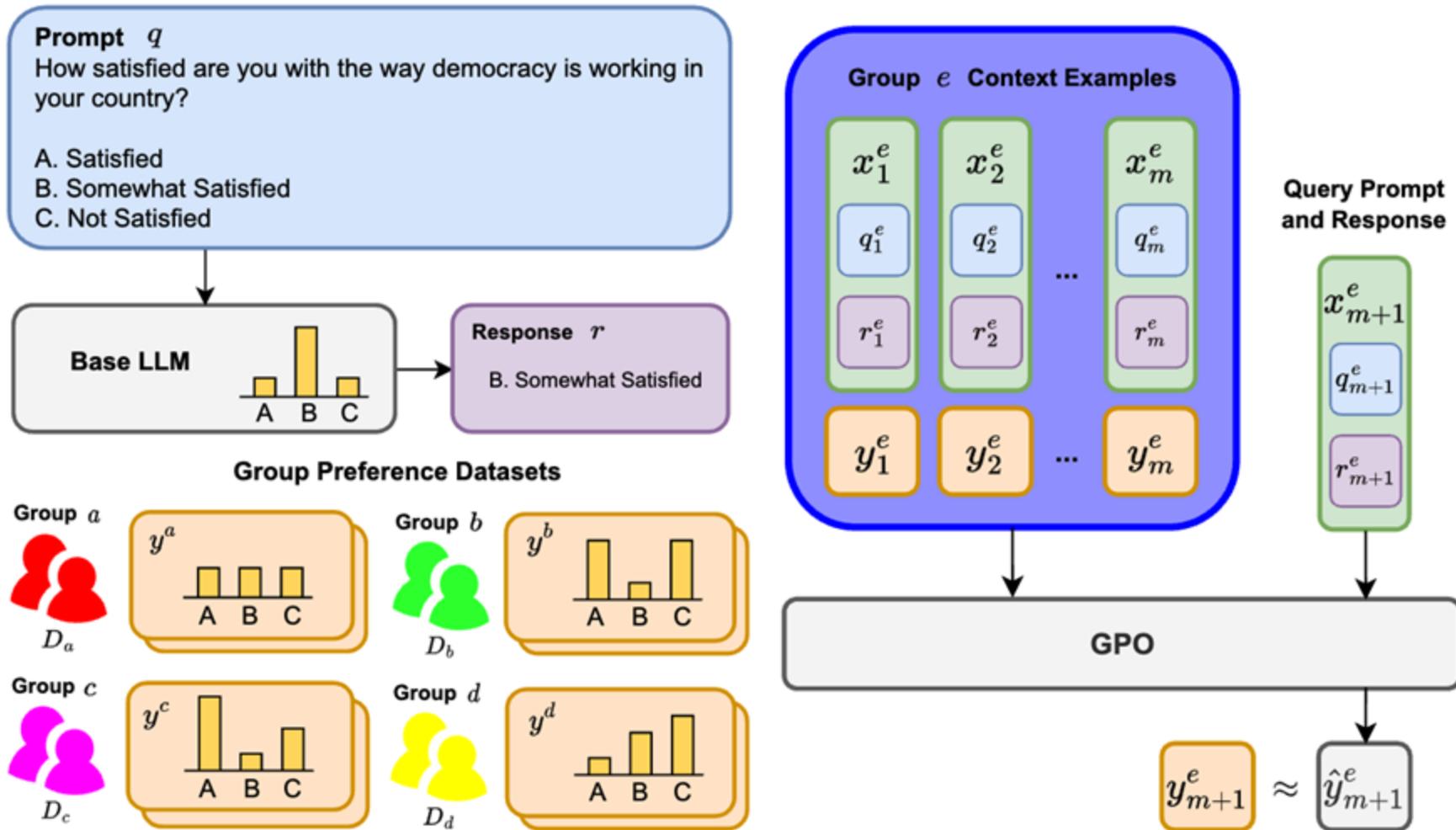
- ✓ Constitutional Maker

- Group preference optimization

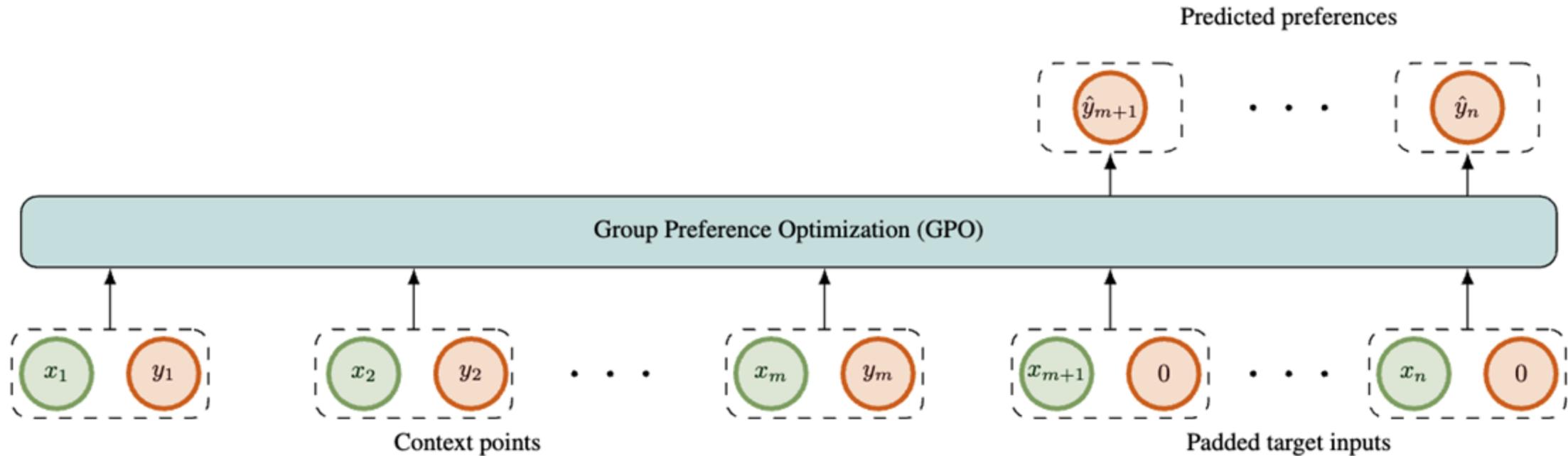
- Demonstrated feedback

- Learning from user edits

Preference Tuning: Group Preference Optimization

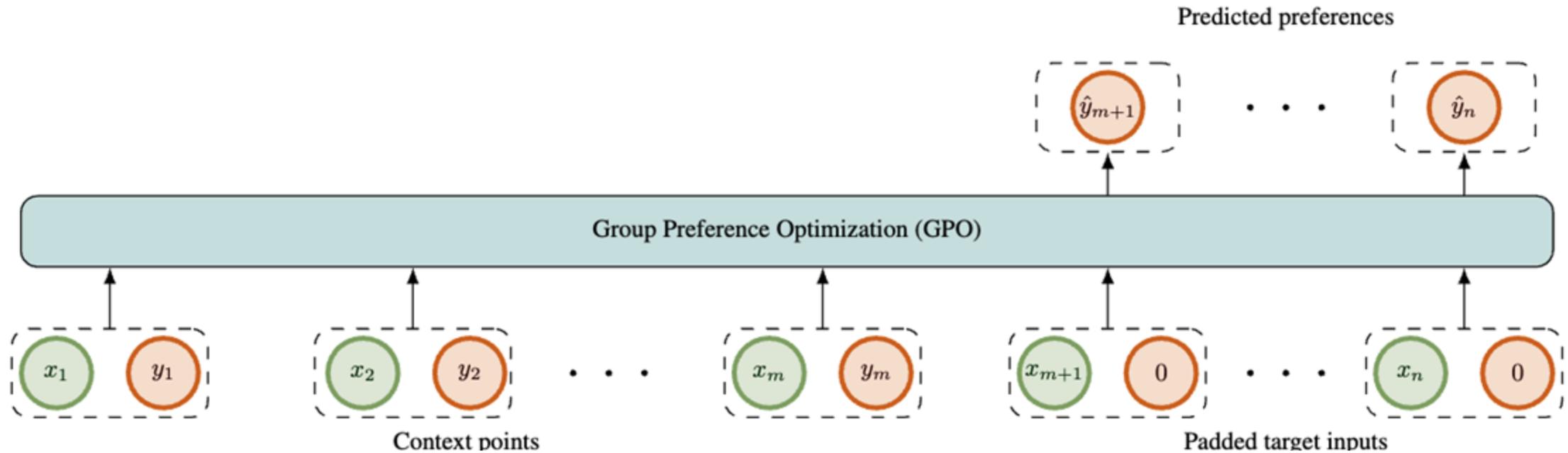


Preference Tuning: **Group Preference Optimization**



GPO architecture for a sequence of n points, with m context points and $n - m$ target points. The context serves as few-shot conditioning for GPO. GPO processes the full sequence using a transformer and predicts the preference scores.

Preference Tuning: **Group Preference Optimization**

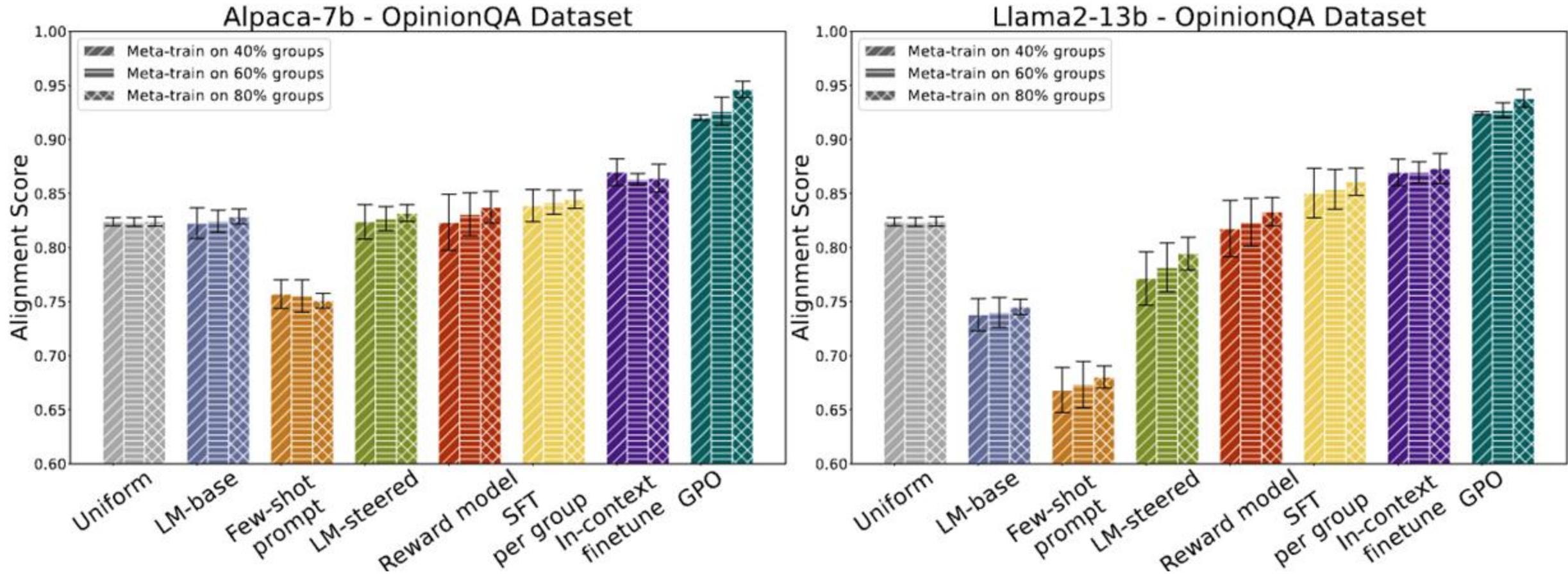


$$L(\theta) = \mathbb{E}_{g,m} \left[\sum_{i=m+1}^n \log p_{\theta}(y_i^g \mid x_{1:n}^g, y_{1:m}^g) \right]$$

Algorithm 1 *Group Preference Optimization (GPO)*

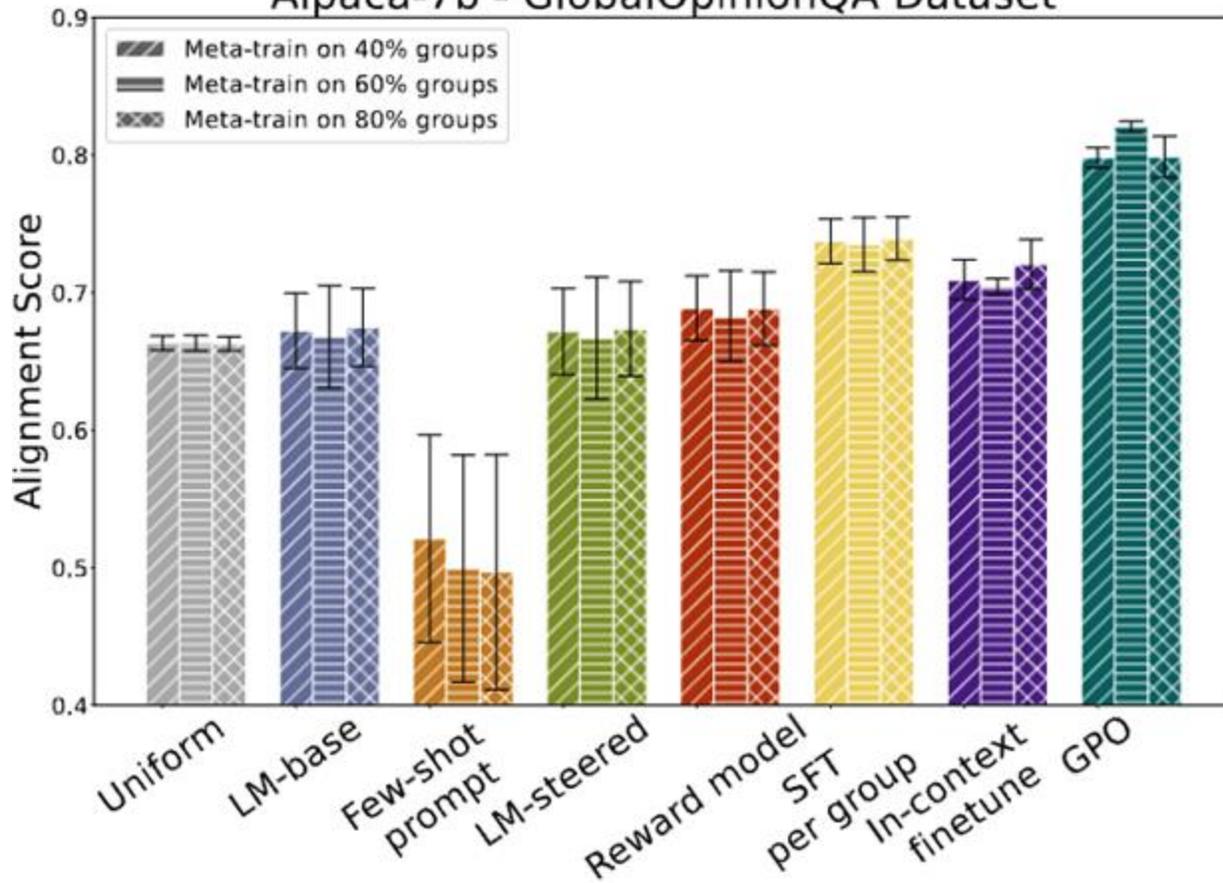
- 1: **Input:** LLM embedding function π_{emb} ; Preference datasets $\mathcal{D}_g \forall g \in G_{\text{train}}$.
 - 2: Initialize GPO transformer with parameters θ .
 - 3: For all $g \in G_{\text{train}}$, cache embedded pairs (x_i^g, y_i^g) in $\mathcal{D}_g^{\text{emb}}$ where $x_i^g = \pi_{\text{emb}}(q_i^g, r_i^g)$.
 - 4: **repeat**
 - 5: Sample training group $g \in G_{\text{train}}$.
 - 6: Sample context size $m \sim \text{Uniform}[1, n - 1]$ where $n = |D_g|$.
 - 7: Split $\mathcal{D}_g^{\text{emb}}$ randomly into m context $(x_{1:m}^g, y_{1:m}^g)$ and $(n - m)$ target $(x_{m+1:n}^g, y_{m+1:n}^g)$ pairs.
 - 8: Predict target preferences $y_{m+1:n}^g$ using context $(x_{1:m}^g, y_{1:m}^g)$ and padded targets $(x_{m+1:n}^g, 0)$.
 - 9: Update θ to minimize in-context loss function $L(\theta)$ in Eq. 2.
 - 10: **until** convergence
 - 11: **Output:** GPO transformer with learned parameters θ
-

Preference Tuning: **Group Preference Optimization**



Preference Tuning: **Group Preference Optimization**

Alpaca-7b - GlobalOpinionQA Dataset



Llama2-13b - GlobalOpinionQA Dataset

