A Multi-Agent Conversational Bandit Approach to Online Evaluation and Selection of User-Aligned LLM Responses

Xiangxiang Dai[1], Yuejin Xie[2], Maoli Liu[1], Xuchuang Wang[3], Zhuohua Li[4]*, Huanyu Wang[5], John C.S. Lui[1]

[1]The Chinese University of Hong Kong, [2]Huazhong University of Science and Technology, [3]UMass Amherst,
[4]Xidian University, [5]Huawei Technologies Co., Ltd.

1. Introduction & Motivation

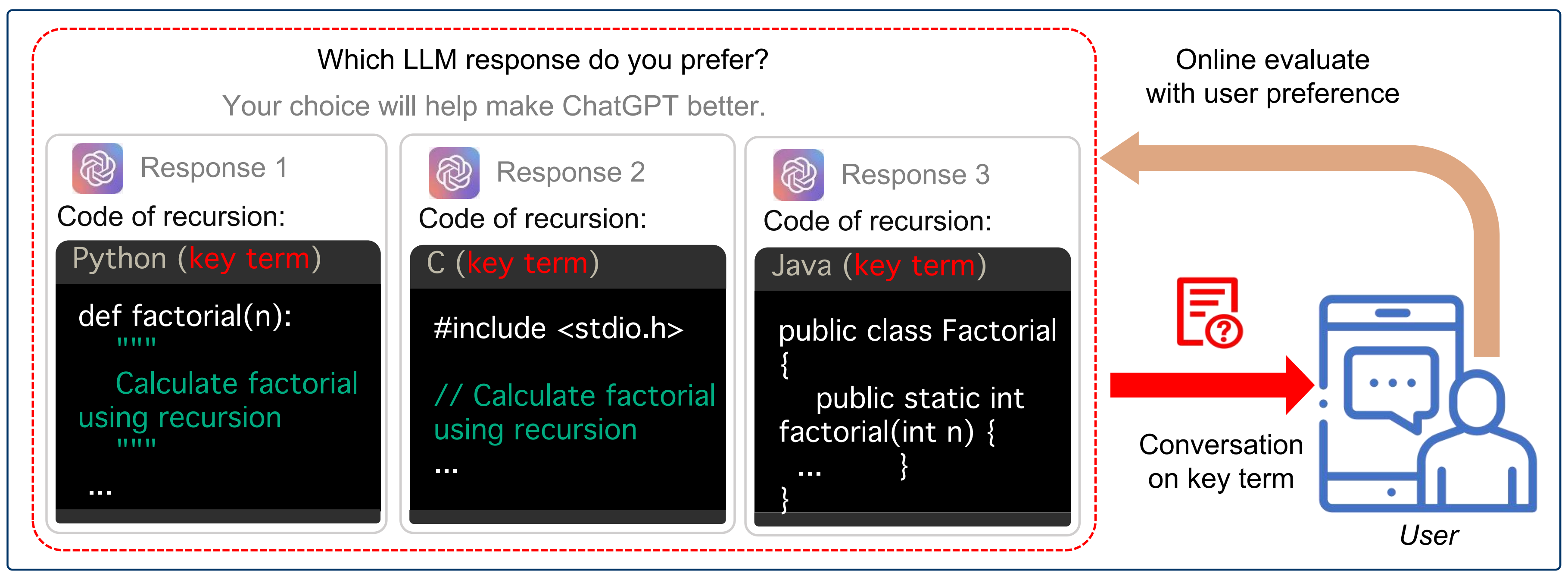
Problem: Optimizing LLM responses using offline prompt engineering is computationally intensive and often fails to accommodate diverse user response styles (e.g., humorous vs. formal).

Challenges:

- High-dimensional features of LLM responses.
- Large but finite sets of candidate responses.
- Need for adaptive alignment with personalized user preferences.
- Multi-device access (distributed agents).

Our Solution (MACO): Multi-Agent Conversational Online Learning.

- **Online Evaluation:** Selects optimal responses dynamically.
- **Conversational Bandit:** Agents query users on "Key Terms" (e.g., style preference) to speed up learning.
- **Collaboration:** The cloud server aggregates data to guide local agents.



```
def factorial(n):
    """
    Calculate factorial
using recursion
    """
 ...
```

```
#include <stdio.h>

// Calculate factorial
using recursion
...
```

```
public class Factorial
{
    public static int
factorial(int n) {
  ...         }
}
```

Figure: Example: online user feedback on coding style guides ChatGPT to refine recursive code outputs, improving user preference alignment.

2. System Model

We consider a distributed setting with M local agents and one Cloud Server.

- **Arms ($\mathcal{A}_m$):** Finite set of LLM responses generated offline.
- **Reward:** $r_{m,t} = \langle x_{a_{m,t}}, \theta_t^* \rangle + \eta_{m,t}$, where θ^* is the unknown user preference.
- **Key Terms ($\mathcal{K}$):** Attributes (e.g., "Python Code", "Casual Tone") for queries.

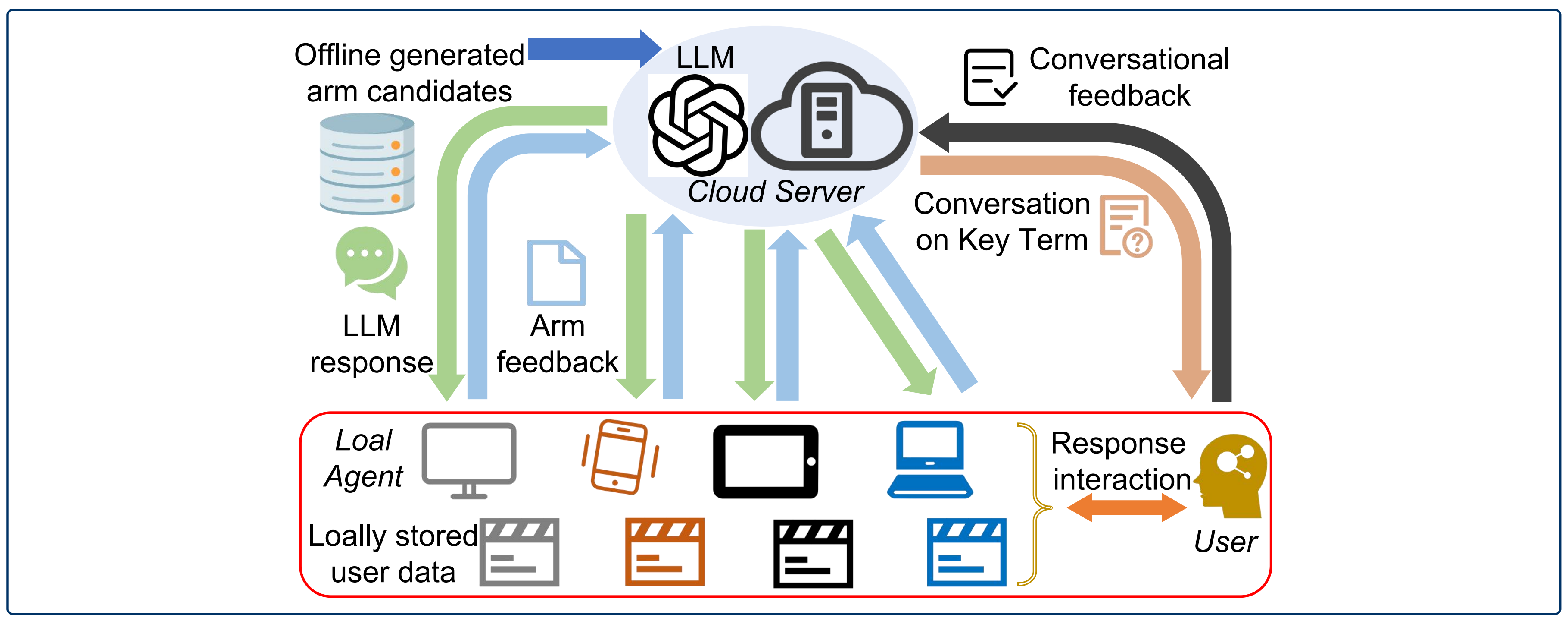


Figure: Local agents handle response selection; Server manages conversation flow via key terms.

3. MACO Algorithm Design

MACO consists of two synchronized components:

1. MACO-A (Local Agent):

- **Information Matrix:** Calculates M_m^p based on active arms.
- **Eigenvalue Check:** If variance in a direction is high (eigenvalue $\lambda < h_p$), upload the eigenvector to the cloud.
- **Action:** Pulls arms and queries specific Key Terms sent by the server.
- **Elimination:** Removes sub-optimal arms based on the updated global preference estimate $\hat{\theta}$.

2. MACO-S (Cloud Server):

- **Aggregation:** Receives uploaded eigenvectors from agents.
- **Optimization:** Selects Key Terms ($k \in \mathcal{K}$) that maximize information gain in under-explored directions.
- **Estimation:** Aggregates feedback matrices G and W to compute $\hat{\theta} = G^{-1}W$.

4. Theoretical Guarantees

MACO achieves near-optimal performance with adaptive communication.

Theorem 1: Regret Bounds

Upper Bound: Cumulative regret is: $R_M(T) \le \mathcal{O}\left(\sqrt{dMT \log \frac{AM \log T}{\delta}}\right)$

Lower Bound: We prove a matching lower bound of $\Omega(\sqrt{dMT})$.

Result: MACO is minimax optimal up to logarithmic factors.

Theorem 2: Communication Cost

The communication overhead scales as: $\mathcal{O}(d^2 M \log T)$ *Scales logarithmically with T and is independent of arm pool size A.*

Theorem 3: Conversation Frequency

Conversations are triggered adaptively based on information gain:

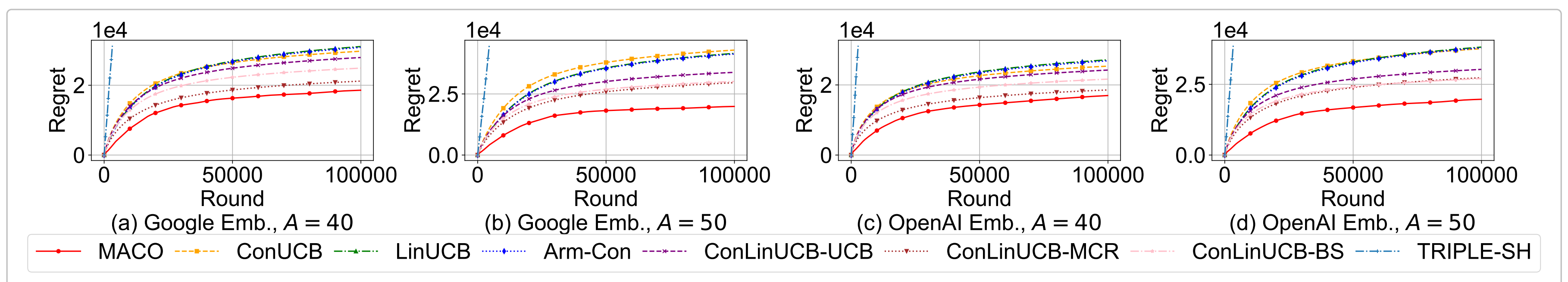
- If information is sufficient ($\gamma \ge h_p$), zero conversations are initiated.
- Otherwise, frequency is bounded, avoiding unnecessary user interruptions.

5. Evaluation

Basic Setup:

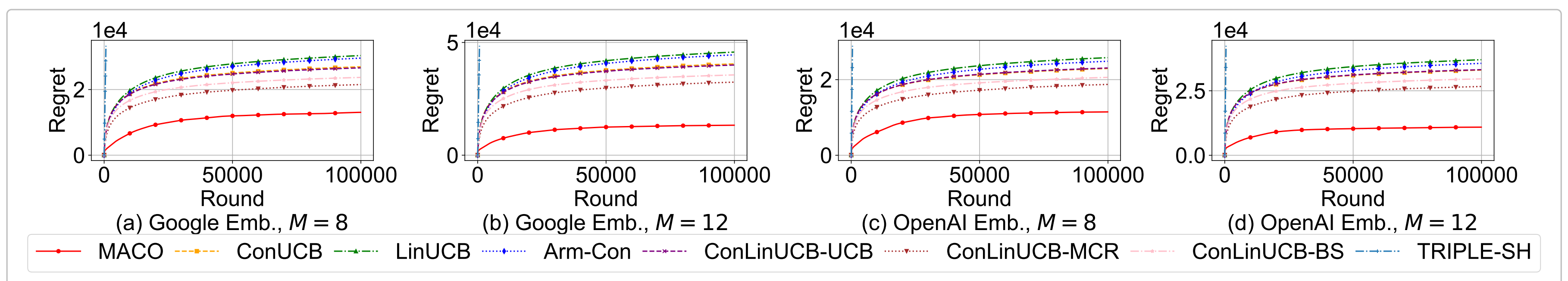
- **Models:** Google Text Embeddings & OpenAI Embeddings.
- **Dataset:** StyleEval, Multilingual datasets + Llama-3 generated responses.
- **Baselines:** LinUCB, Arm-Con, ConUCB, TRIPLE-SH, ConLinUCB-UCB, ConLinUCB-BS, ConLinUCB-MCR.

Evaluation 1. Regret Analysis: Robustness & Scalability



(a) Google Emb., $A = 40$ (b) Google Emb., $A = 50$ (c) OpenAI Emb., $A = 40$ (d) OpenAI Emb., $A = 50$

Regret on embedding models from Google and OpenAI across different arm pool sizes A.



(a) Google Emb., $M = 8$ (b) Google Emb., $M = 12$ (c) OpenAI Emb., $M = 8$ (d) OpenAI Emb., $M = 12$

Regret on embedding models from Google and OpenAI across different agent counts M.

Evaluation 2. Efficiency & Real-world Performance

Table: Execution time and average reward on different settings (± standard deviation).

Algorithm / Setting	**MACO (w/o G)**		MACO (w/G)		ConLinUCB-BS	
	Time (s)	Reward	Time (s)	Reward	Time (s)	Reward
Setting (a)	2.576 ± 0.047	61.849 ± 0.558	9.766 ± 2.709	61.847 ± 0.565	18.124 ± 0.111	59.811 ± 0.610
Setting (b)	2.546 ± 0.039	61.605 ± 0.642	14.272 ± 7.107	61.591 ± 0.649	18.056 ± 0.065	59.663 ± 0.671
Setting (c)	2.576 ± 0.085	47.405 ± 0.977	6.369 ± 2.832	47.381 ± 1.002	17.926 ± 0.095	46.104 ± 0.962
Setting (d)	2.661 ± 0.056	41.770 ± 0.349	6.270 ± 2.013	41.858 ± 0.412	17.919 ± 0.072	40.720 ± 0.349

Key Findings: achieves the lowest regret on real-world datasets. By removing the expensive G-optimal design, `MACO (w/o G)` reduces execution time by ~**7x** compared to baselines while maintaining equivalent reward performance.

6. Conclusion

- **Proposes** MACO, a multi-agent framework for online LLM response alignment.
- **Utilizes** an adaptive conversational mechanism to query preferences efficiently.
- **Proves** near-optimal regret bounds and reduced communication costs.
- **Demonstrates** superior performance on real-world datasets with Llama.

References & Resources

Full Paper (arXiv)

Source Code

Association for the Advancement of Artificial Intelligence