POLLEN: Prompt-based Orchestration of LLM-decomposed services over Edge Nodes

1st Francesco Cosimo Mazzitelli
dept. of Engineering
University of Sannio
Benevento, Italy
f.mazzitelli@studenti.unisannio.it

2nd Eugenio Zimeo
dept. of Engineering
University of Sannio / CINI
Benevento, Italy
zimeo@unisannio.it

***Abstract*—Smart cities increasingly rely on heterogeneous REST services for transportation, tourism, parking, environmental monitoring, and municipal assistance. However, these services remain fragmented across multiple platforms, limiting accessibility and usability for citizens and tourists.**

POLLEN is a distributed LLM-based orchestration framework that converts user requests into executable multi-step API workflows over existing REST infrastructures. Furthermore, the system leverages distributed edge inference via the *distributedLlama* engine, enabling fully on-premises deployment on low-cost devices without reliance on centralized cloud infrastructure or specialized hardware.

***Index Terms*—API Orchestration, Large Language Models, Semantic Retrieval, Edge Computing, Distributed Inference, Service Composition**

I. Introduction

Smart cities expose a growing number of REST APIs for transportation, tourism, parking, environmental monitoring, and municipal assistance. However, their fragmentation across independent platforms limits accessibility and usability. At the same time, smart-city infrastructures contain many underutilized edge devices whose computing resources remain largely unused and could be integrated into a distributed grid to support decentralized LLM inference.

These devices are often highly heterogeneous, ranging from low-cost single-board computers such as Raspberry Pi to more powerful edge accelerators such as NVIDIA Jetson platforms. POLLEN aims to exploit these underutilized resources by enabling their participation in a collaborative inference network that transforms existing smart-city infrastructure into a decentralized computing substrate for AI-powered services.

Although LLMs have recently shown promising capabilities in tool usage and structured generation, existing approaches often rely on large-scale models with billions of parameters, making their deployment challenging, particularly in resource-constrained edge environments.

POLLEN addresses these challenges through a distributed orchestration framework that transforms natural-language requests into executable API workflows within existing REST ecosystems. The system combines: (i) multi-stage semantic retrieval with reranking; (ii) guided JSON decoding for schema-constrained generation; (iii) failure-aware orchestration analysis; and (iv) distributed on-premise inference based on the *distributedLlama* engine, leveraging underutilized low-cost edge devices as a decentralized inference grid.

II. Related Works

Research on tool-augmented large language models has investigated how LLMs interact with external APIs and software systems. Early methods such as Toolformer [1] and Gorilla [2] enable direct tool invocation and API call generation from documentation, but mainly target single-function execution and provide limited support for multi-step orchestration, dependency handling, and strict schema validation.

Subsequent planning-oriented approaches, including LLM-Compiler [3] and ReAct-style agents [4], improve multi-step reasoning and tool chaining, but still rely on loosely structured outputs that require post-processing and remain prone to hallucinated parameters or invalid field generation.

Advances in semantic retrieval improved grounding over large API and document collections. Sentence-BERT [5] enables scalable semantic search via bi-encoder embeddings, while BGE reranking [6] improves precision using cross-encoder re-ranking. POLLEN builds on these methods through a multi-stage retrieval pipeline that combines semantic search, reranking, and query decomposition to improve recall and precision across heterogeneous API catalogs.

Finally, distributed inference frameworks such as distributedLlama demonstrate collaborative execution of large transformer models across low-resource devices. POLLEN adopts this paradigm to enable fully on-premise orchestration without reliance on centralized cloud inference providers.

III. POLLEN

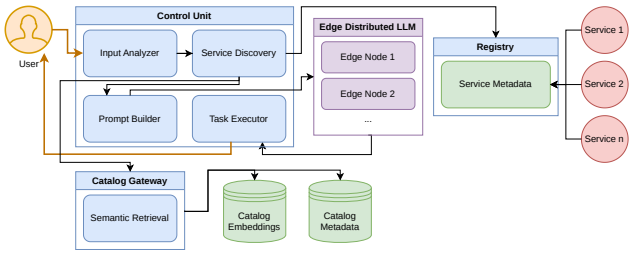

Fig. 1. System architecture

POLLEN is characterized by a modular orchestration framework that translates natural language queries into executable multi-step workflows. As shown in Fig. 1 the system grounds user requests against a structured service catalog using a multi-stage semantic retrieval pipeline that combines query decomposition, semantic search, and reranking to improve both recall and endpoint precision across heterogeneous domains. Retrieved services guide a constrained planning stage in which workflows are generated under a closed-world assumption: only known services and valid parameters can be used. Structured decoding and schema-aware validation enforce consistency and prevent the introduction of hallucinated fields, unsupported operations, or invalid parameter values.

Execution is performed, enabling outputs from earlier tasks to dynamically parameterize subsequent operations. This supports data-dependent orchestration, multi-service chaining, and analytical transformations such as joins, aggregations, and filtering across heterogeneous API responses. To improve robustness, POLLEN includes a fallback analysis stage for cases where no valid workflow can be produced. Instead of failing silently, the system classifies the reason for failure, including ambiguous queries, invalid requests, or missing capabilities, and can optionally generate a conceptual API specification describing absent functionality.

The distributed inference layer allows heterogeneous edge nodes to collaboratively contribute to workflow generation without requiring homogeneous infrastructure. This enables gradual integration of existing municipal hardware resources, from low-power Raspberry Pi boards to GPU-enabled NVIDIA Jetson devices, without requiring homogeneous infrastructure.

IV. Evaluation and Results

POLLEN was evaluated on the RestBench benchmark, enabling direct comparison with RestGPT [7] on multi-step API workflows over TMDB and Spotify services. Evaluation measures plan correctness, execution success, and schema compliance through Oracle-based metrics and runtime validation. Scalability was assessed on a cluster of four Raspberry Pi devices (4 CPU cores, 8 GB RAM each) running pretrained models with distributedLlama. Raspberry Pi boards were chosen as a representative and widely adopted edge-computing platform for smart-city and IoT deployments.

TABLE I
Comparison between POLLEN and RestGPT.

Model	TMDB				Spotify			
	S%	CP%	ΔSL	sec	S%	CP%	ΔSL	sec
ReAct [4]	44.0	57.0	+0.76	-	54.5	49.1	+0.31	-
davinci	**75.0**	**79.0**	**+0.55**	-	**72.7**	**74.5**	+0.25	-
chatGPT	68.0	65.0	+0.72	-	69.1	72.3	+0.28	-
Vicuna	9.0	15.0	+1.21	-	12.7	20.6	+1.52	-
Qwen0.6b	40.0	42.0	+1.38	675.32	21.05	22.81	+1.39	624.83
Qwen30b	67.0	72.0	+0.60	1214.36	40.35	45.61	+1.12	1125.51

As shown in Table I, POLLEN achieves competitive orchestration performance using q4-quantized models deployed entirely on edge devices. We evaluate *Qwen3-0.6B*, a lightweight baseline, and *Qwen3-30B-A3B*, a Mixture-of-Experts model with 30B total and approximately 3B active parameters per token.

Following RestGPT, we report *Success Rate* (S%), *Correct Path Rate* (CP%), Δ *Solution Length* (ΔSL), and *Latency*. S% measures task completion, CP% path correctness, ΔSL excess API calls over the gold solution, and Latency plan generation time. *Qwen3-30B* approaches RestGPT performance on TMDB, achieving 67% S% and 72% CP% versus 75% and 79% for *text-davinci-003*. Although Spotify's results are lower, effective multi-step orchestration remains achievable without cloud services or GPU infrastructure.

Latency is influenced by two factors: the reasoning-oriented nature of Qwen3 models, which generate longer intermediate traces, and distributedLlama's token-level streaming, which introduces communication and I/O overhead. Reported values, therefore, reflect both inference and distributed coordination costs. These results indicate that orchestration quality depends not only on model scale but also on semantic retrieval, schema-aware validation, and guided decoding.

V. Conclusion

POLLEN is a distributed framework for natural-language orchestration of heterogeneous REST services in smart-city environments. By combining semantic retrieval, constrained workflow generation, guided JSON decoding, and failure-aware analysis, it generates executable workflows while reducing hallucinations and invalid service invocations.

A key contribution is the integration of distributed inference through distributedLlama, enabling deployment on clusters of low-cost devices connected through dedicated municipal networks. This removes dependence on external cloud providers while improving privacy, resilience, and operational control, making POLLEN suitable for interactive smart-city infrastructures such as urban information kiosks.

Future work will focus on overcoming current distributedLlama limitations and evaluating larger models on larger edge clusters to further improve orchestration capabilities and scalability.

References

[1] T. Schick *et al.*, "Toolformer: language models can teach themselves to use tools," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[2] S. G. Patil *et al.*, "Gorilla: Large language model connected with massive apis," in *Advances in Neural Information Processing Systems*, A. Globerson *et al.*, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 126 544–126 565.

[3] S. Kim *et al.*, "An llm compiler for parallel function calling," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[4] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[5] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[6] S. Xiao *et al.*, "C-pack: Packaged resources to advance general chinese embedding," 2023.

[7] Y. Song *et al.*, "Restgpt: Connecting large language models with real-world restful apis," 2023.