MALM: Modular Adapters for Efficient and Scalable Multilingual Language Models

Hilal Limo
Independent Researcher

Abstract

Large multilingual LMs are costly to train and serve; adding or improving languages often requires full-model updates. I propose *MALM*, a modular design that separates a single strong *Core Language Model* (CLM; English-centric for reasoning and instruction following) from a bank of lightweight *Specialized Translation Adapters* (STAs). The CLM emits structured *delegation tokens* that an orchestration layer routes to STA experts, returning translated segments for final assembly. This keeps reasoning weights compact, allows drop-in language upgrades, and plays well with modern efficient fine-tuning, quantization, and serving. I discuss training, latency/quality trade-offs, and why small models benefit disproportionately. I position MALM versus adapters, MoE, and modern MT systems, and outline practical paths to deployment.

1 Introduction

The rapid advancement of Large Language Models (LLMs) has unlocked unprecedented capabilities in natural language understanding and generation. A key area of progress has been multilingualism, where models like GPT-5 and Gemini can converse and translate across dozens of languages. However, this capability is achieved by scaling model size and training data to immense proportions. This "bigger is better" approach presents significant challenges:

- **Computational Cost:** Training a single, massive multilingual model requires thousands of high-end GPUs running for weeks or months, an endeavor accessible only to a few large corporations. Inference costs are also substantial.
- **The Curse of Multilinguality:** A model's finite parameter space must be shared among all languages it supports. This can lead to knowledge dilution, where proficiency in any single language is compromised by the need to accommodate others.
- **Scalability and Maintenance:** Adding a new language or updating an existing one requires expensive retraining or fine-tuning of the entire monolithic model. This process is slow, inefficient, and not scalable.

To address these challenges, I propose the **Modular Adapter-based Language Model (MALM)**. MALM shifts from a single, all-knowing model to a collaborative system of specialized agents. The core is a highly proficient language model trained in a single high-resource language (e.g., English). This *Core Language Model* (CLM) is the "brain," responsible for reasoning and conversational flow. When another language is required, the CLM does not perform the translation itself; instead, it delegates to a small, efficient, specialized translation adapter.

2 Related Work

MALM builds on several research directions in machine learning.

2.1 Mixture-of-Experts (MoE)

MoE models such as Switch Transformer [1] employ gating to route inputs to specialized sub-networks, enabling trillion-parameter models with constant compute per input. MALM echoes this idea but at a higher semantic level: the CLM acts as a router that delegates tasks to external experts.

2.2 Tool-Augmented LMs

Tool-augmented models like Toolformer [2] show that LMs can learn to issue structured API calls to external tools. MALM applies this principle to translation: the CLM learns when to call a translation adapter, separating reasoning from execution.

2.3 Adapter-based Fine-Tuning

Adapters [3] provide parameter-efficient task adaptation by injecting small modules into frozen LMs. MALM takes inspiration but externalizes the adapters as standalone models, achieving modularity and easy swapping.

3 The MALM Architecture

MALM has three components: a Core Language Model (CLM), a set of Specialized Translation Adapters (STAs), and an Orchestration Layer.

CLM. Trained mainly on English, the CLM excels at reasoning and instruction following. It is tuned to produce structured delegation tokens instead of performing translation directly.

STAs. Lightweight NMT models specialized per direction (e.g., EN↔DE). They are small, efficient, and swappable.

Orchestration. The orchestrator parses tokens and routes payloads to the correct STA(s), reintegrating outputs into the final response.

3.1 Example Conversation Flows

To illustrate how MALM operates, I show a few simple exchanges where the CLM delegates translation to STAs via delegation tokens:

- **English → German User:** Translate "my name is Adam" into German.
  CLM Output: `<to:de> my name is Adam </to>`
  Final Response: "Mein Name ist Adam"
- **Spanish → English (input STA) + reasoning User:** (Spanish) "¿Cuánto es 12 + 7?"
  Input STA: "How much is 12 + 7?"
  CLM: "The answer is `<to:es> 19 </to>`"
  Final Response: "La respuesta es 19"
- **Multilingual Output User:** Say "good morning" in French and Japanese.
  CLM Output: `<to:fr> good morning </to>` and `<to:ja> good morning </to>`
  Final Response: "In French: bonjour. In Japanese: おはようございます"

These flows show how the CLM handles reasoning and orchestration, while STAs perform the actual translations.

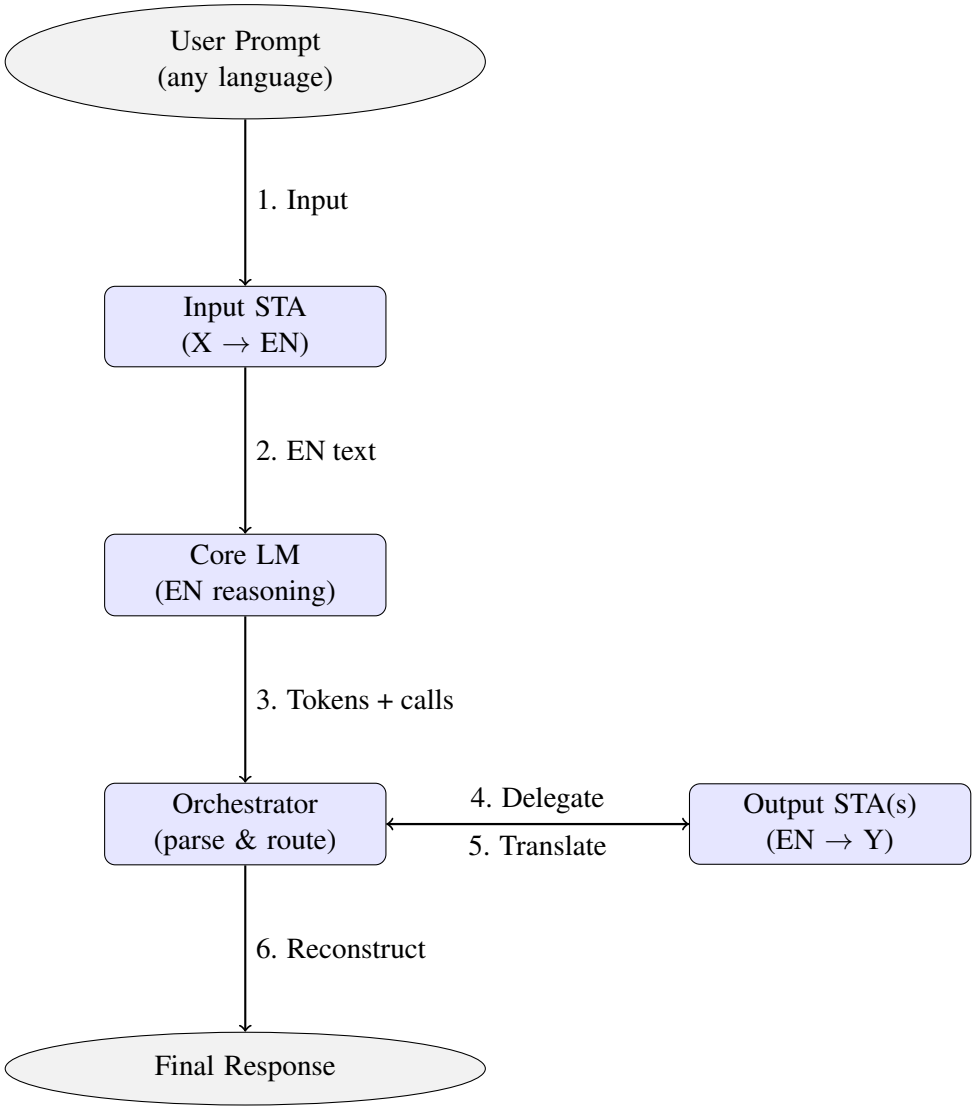

Figure 1: MALM delegation and assembly.

4 Advantages

- **Efficiency:** A smaller CLM plus lightweight adapters reduce both training and inference cost.
- **Scalability:** Adding a new language is as simple as plugging in a new STA.
- **Specialization:** Dedicated NMT models outperform general-purpose LMs on translation.
- **Maintainability:** Underperforming adapters can be swapped without retraining the core.

5 Limitations & Future Work

MALM introduces some latency from delegation. Direct adapters (X→Y) for common pairs and orchestrator optimization can help. Chained translations may risk nuance loss; exploring non-English-centric delegation is future work. Beyond translation, the same framework could support adapters for tasks like math, code, or multimodal reasoning.

6 Conclusion

MALM shows how separating reasoning from translation creates a system that is smaller, cheaper, and easier to maintain than monolithic multilingual LMs. By keeping one reasoning core and outsourcing skills to modular experts, I provide a path toward more flexible, composable AI.

References

[1] Fedus, W., Zoph, B., & Shazeer, N. (2021). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961.

[2] Schick, T. et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761.

[3] Houlsby, N. et al. (2019). Parameter-Efficient Transfer Learning for NLP. ICML 2019.