LLMs as On-demand Customizable Service

Souvika Sarkar[1], Mohammad Fakhruddin Babar[2], Monowar Hasan[2], Shubhra Kanti Karmaker Santu[1]
[1] Big Data Intelligence (BDI) Lab, Auburn University
[2] School of Electrical Engineering & Computer Science, Washington State University
szs0239@auburn.edu, m.babar@wsu.edu, monowar.hasan@wsu.edu, sks0086@auburn.edu

Abstract

LLMs, despite their remarkable advantages, face significant challenges due to their immense size. These models require substantial computational resources, often lacking on local devices, hindering their accessibility and customization. To address these challenges, we propose a hierarchical, distributed architecture.

1 Introduction

Our solution aims to enhance LLM accessibility and utility through the following aspects: **Hierarchical Organization of Knowledge,** by structuring LLMs hierarchically, we distribute vast knowledge across layers based on language, application domains, and sub-domains. This organization minimizes redundancy and allows for more efficient storage of information. **Enhanced Customization,** users can select LLMs tailored to their specific needs, avoiding monolithic models. Configurability further allows for adjustments to suit specific applications, enhancing flexibility. **Efficient Resource Management,** resource allocation is optimized by matching LLMs to hardware capabilities, preventing over-commitment of resources. This ensures effective operation across devices with varying computational capacities. **Scalability,** the hierarchical structure supports scalability, enabling users to upgrade to models with enhanced capabilities and larger knowledge bases as application demands grow. This ensures applications can handle more complex tasks without a complete overhaul of the model architecture.

2 A Multi-Layer LLM Architecture

This proposed architecture (Figure 1) organizes multiple language models in a hierarchical order, considering languages, domains, sub-domains, variations in size, resource requirements, and computational cost (For details of the components refer to 4.1). Language models are arranged in a "top-down" manner, with larger models at the top and smaller models at the bottom, following a decreasing order of size, resource availability, and computation cost. This hierarchical arrangement enables users to select a language model that suits their needs and available resources.

Workflow: i. The interacts with a Virtual Assistant, specifying requirements for their application. ii. The virtual assistant consults a Language Model Recommender System to recommend the most suitable model, considering user specifications and resource constraints. iii. The user clones the recommended model and fine-tunes it on their goal task using local devices. iv. Continual learning allows users to update the model with new data, ensuring it remains relevant and accurate over time. v. Peer language models are notified of updates or fine-tuning, ensuring consistency across the system. vi. Knowledge transfer mechanisms takes place for sharing of new information between language models, both upstream and downstream, enhancing the overall knowledge base of the system.

3 Challenges and Conclusion

There exist several open deployment challenges, as below:

Challenge 1: Identifying Suitable Language Models A thorough examination of LLMs across diverse resource constraints and accuracy requirements is vital. This study enables informed decisions when selecting LLMs for applications, streamlining development. A comprehensive investigation, as conducted by (Karmaker et al., 2021), can elucidate the pipeline of machine learning tasks and pinpoint the stages most vulnerable to resource constraints.

Challenge 2: Coordinating Continuous Updates Effective collaboration among layers is essential for seamless continual learning. Knowledge transfer occurs dynamically in both *Upstream* and *Downstream*. To address the privacy concerns, a continual learning process with two com-

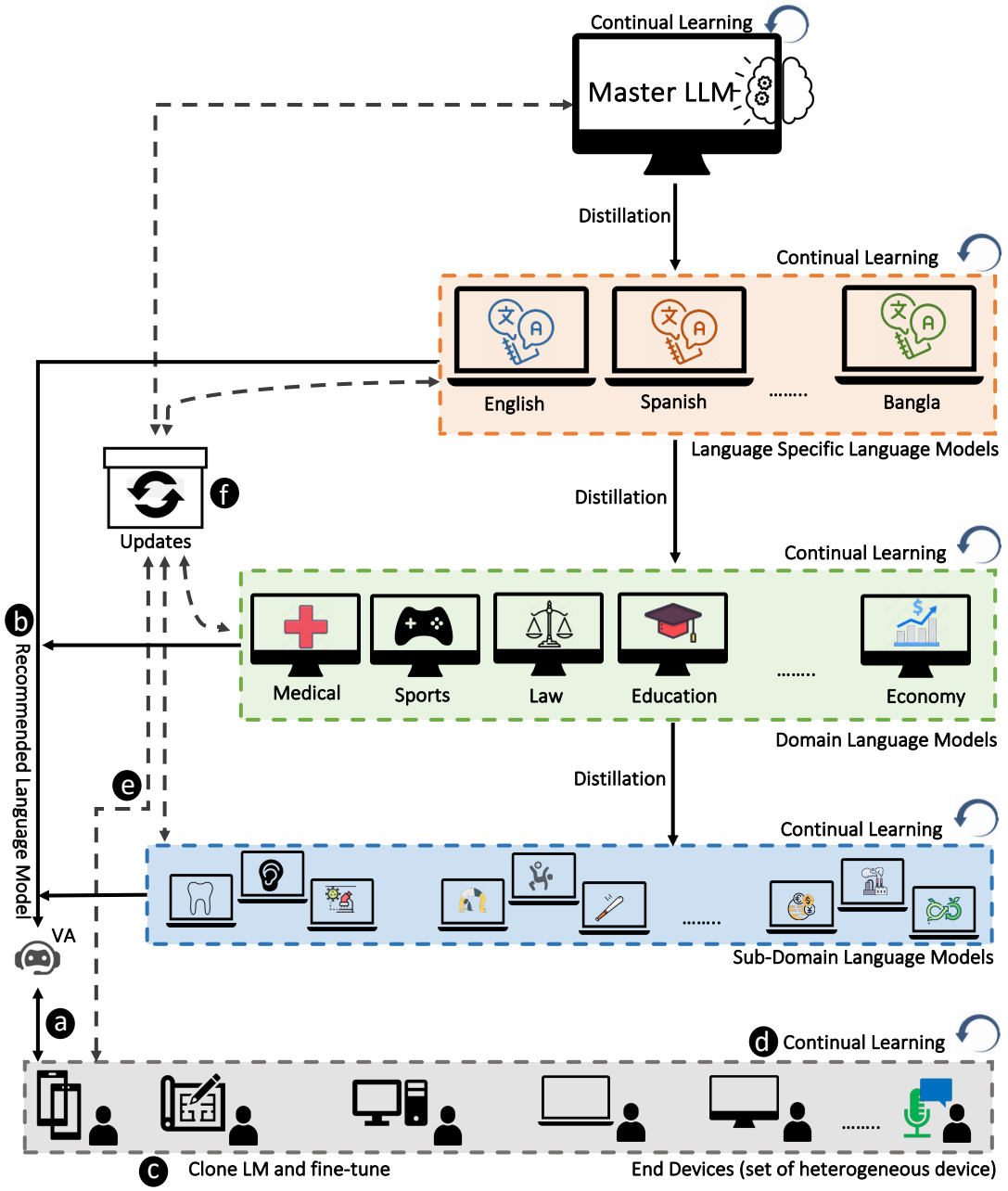

Figure 1: High-level schematic diagram of a multi-tier distributed LLM architecture.

ponents can be leveraged: the *generator* and the *learner* (Sun et al., 2019).

Challenge 3: Preventing Loss of Previously Learned Knowledge Catastrophic forgetting (Li et al., 2019) poses a challenge where previously acquired knowledge is at risk of being lost during continual learning. Further research is needed to address this challenge in hierarchical LLM architectures.

Challenge 4: Timing Updates for the Parent Language Model Determining when to update the parent language model is crucial. Managing constant updates from numerous end devices requires evaluating the significance of new data. One strategy could be evaluating the significance of new data in contributing to the broader knowledge base (Ke et al., 2023).

Challenge 5: Addressing Malicious Nodes Risks from malicious nodes, such as data or model poisoning attacks, threaten system stability (Tolpegin et al., 2020). Techniques to isolate malicious nodes, limit their information propagation are necessary for robustness.

This "layered" LLM architecture can tackle the challenges of deploying LLMs in practical, real-world applications. We believe this concept can serve as a stepping stone for implementing open-source, customizable LLM architecture.

References

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819.

Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. 2021. Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54(8):1–36.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pretraining of language models. In *The Eleventh International Conference on Learning Representations*.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. 2019. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 480–501. Springer.

4 Appendix

4.1 Components and Functions

We now present the components and describe the functionalities of each layer.

User. The user represents the end user who desires to obtain language model services based on their specific requirements and preferences.

Virtual Assistant (VA). The VA interfaces the user and the backend layered architecture. The user interacts with the VA and provides specifications, such as the desired platform and services they are looking for. The VA then recommends the most suitable language model instance.

Master LLM Layer (Root). At the root of our hierarchical architecture resides the *Master LLM*, the largest general-purpose language model available, and serves as the base (*"Teacher"*) model for transferring knowledge to successor language models.

Language Specific Language Model (LSLM) Layer. The following layer in the hierarchy is language models specific to a particular language (called LSLM, depicted by the Orange box in Fig. 1). LSLMs are smaller than the Master LLM. We can use distillation techniques (Gou et al., 2021) across the hierarchy to transfer knowledge from a larger model to a smaller model. As an example, the Master LLM acts as the *"Teacher"* and the language-specific (e.g. English, Spanish) models as the *"Student"* during the distillation process.

Domain Language Model (DLM) Layer. The subsequent layer contains domain-specific language models for each Language Specific Language Model (LSLM), such as Medical, Sports, Law, and Education, as shown by the Green box in Fig. 1). These domain-specific language models are i. compact in size, ii. possess fewer parameters, and iii. exhibit lower complexity. However, they possess the essential knowledge/information to excel in their respective domains in a specific language.

Sub-Domain Language Model (SDLM) Layer. The next layer of the architecture consists of SDLMs, essentially specialized language models tailored to specific sub-areas of a domain. For example, in the medical field, sub-domains include Virology or Heart Health. Likewise, in the sports industry, sub-domains may contain language models related to Gymnastics or Soccer. SDLMs can be customized to cater to the specific requirements of each domain precisely while ensuring optimal performance and usability. As we descend the hierarchy, these specialized models (illustrated by the Blue rectangle in Fig. 1) become increasingly focused, compact, and application-friendly.

End Devices Layer. End devices include heterogeneous computing systems such as laptops, tablets, smartwatches, and embedded devices (as shown inside the Gray rectangle in Fig. 1). Depending on the specific application scenarios and requirements/constraints, a user can a. acquire a preferred language model compatible with computing resources and b. fine-tune it on their system depending on the goal task.

Continual Learning. Continual learning is a machine learning paradigm where a model learns from a continuous stream of data over time. Unlike traditional machine learning, where models are typically trained on a static dataset and then tested on new data, continual learning models are designed to adapt and improve their performance as they encounter new data. In our setup, continual learning plays a pivotal role in updating the

knowledge of the entire architecture. This allows the model to a. adapt to new data and language patterns, b. learn from recent examples, and c. improve its performance over time. The open-source nature of the architecture plays a vital part in this process, as it encourages a collaborative effort among a diverse community. With a crowd-sourced community-driven approach, the model can adapt to the latest developments and benefit from a wealth of recent examples derived from various sources, including niche domains (based on user consent). This collective intelligence promotes constant improvements, leading to an architecture that continually refines its performance and consistently delivers more accurate and relevant results over time. As Fig. 1 depicts, we leverage continual learning techniques (Sun et al., 2019) throughout the layer hierarchy (indicated by the circular arrow at the right corner of each layer).

Upstream & Downstream Knowledge Transfer. The bidirectional arrow in Fig. 1 signifies the architecture's dynamic flow of updates and information exchange. When a language model engages in continual learning and updates itself, it triggers a two-way knowledge transfer process. This transfer occurs both Upstream (from bottom to top) and Downstream (from top to bottom) across all layers of language models. This collaborative exchange ensures that all models remain synchronized and can capitalize on the latest advancements and data insights. Although it is conceivable to perform knowledge transfer among language models within the same layer, this falls outside the current scope of our architecture.