Design Pattern for Personal Language Model (PLM) Development

Architectural Overview

Building a Personal Language Model (PLM) involves combining knowledge from "the masses" (many users' data) with on-device personalization. The goal is to jump-start model development using large-scale, multi-user data, then adapt the model to each individual while respecting limited user time/effort and device constraints. This requires a research-backed framework that integrates federated edge training (to leverage multi-user data without centralizing it) and efficient fine-tuning (to personalize with minimal user input). The architecture can be divided into stages: a shared base model learned from many users (ensuring multi-user generalization), and a personalized component tuned on each user's device. This design balances global generalization and individual customization, informed by recent AI research on federated learning, parameter-efficient tuning, and on-device adaptation. Below, we outline the data collection approach and model training pipeline in detail.

Data Collection from the Masses (Pretraining Stage)

To *"jump start"* the PLM, we first gather a large, diverse dataset from many users. This could include public text corpora or opt-in user data (conversations, writing samples, etc.), aggregated in a privacy-conscious way. Rather than requiring extensive manual labeling by third parties (whose *"memory and motivation... is limited"*), this stage leans on **self-supervised learning** (e.g. next-word prediction) to make use of raw text. A broad foundation model is **pre-trained** on this multi-user data to capture general language patterns, facts, and common behaviors. Leveraging a large pre-trained foundation model is beneficial because it provides strong **feature representations** that can be fine-tuned with only small personal datasets[1]. In other words, a model trained on *"the masses"* can supply general linguistic knowledge and vocabulary, forming a **shared starter PLM** for everyone.

Privacy-Preserving Collection: Importantly, data from users should be collected with privacy in mind. One approach is **Federated Learning (FL)** for the initial training. In FL, users' devices collaborate to train a global model **without uploading raw data** to a server[2]. Each device uses local data to train the model and only shares **model updates (parameters)** which the server aggregates into a global model[3]. This way, the *"output"* of the data collection phase is not the raw personal texts, but **learned model parameters** or gradients that reflect patterns in many users' data. For example, Google's Gboard keyboard uses FL to improve its suggestion model using typing data from millions of users, all while keeping each user's keystrokes on their device[4]. This federated pretraining ensures the base PLM **generalizes across users** (multi-user knowledge) but **preserves privacy** by design.

Federated Edge Training for a Shared Base Model

Figure: Federated learning architecture – multiple edge devices (clients) train a shared model collaboratively without centralizing their private data. Each round, devices train locally on their own data and send only model parameter updates to a central server, which aggregates them into a global model[3].

To harness multi-user data on the **edge (on-device)** as specified, the PLM training follows a **federated learning loop** on user devices. The server initializes a base model (potentially the pre-trained foundation model) and sends it to participating devices. Each device (client) then **trains the model on its local dataset** (e.g. the user's messages, documents, etc.), improving on patterns relevant to that user. Rather than training to convergence on one device, the system performs **many small local updates** and periodically sends the updated model (or just the **diff**/gradients) back to the server. The server **aggregates** these updates (e.g. by averaging, known as FedAvg) to yield an improved *global model* that now incorporates knowledge from all users[3]. This global model is sent back out to devices for further rounds, iteratively.

Using this **collaborative edge training**, the *"masses"* of users all contribute to a **shared general model**. The outcome is a PLM that has learned **general language patterns and facts from a wide community**, which new users can download as a strong starting point. Research emphasizes that federated training must handle **heterogeneous data** — each user's data distribution may differ (non-IID)[4]. Indeed, a single global model might perform poorly if one user's data is very different from another's. This motivates *Personalized Federated Learning (PFL)* techniques[5], discussed next, to balance global generalization with local personalization.

Balancing Generalization and Personalization

A key architectural pattern from literature is to **decouple the model into global and personal parameters**. Rather than every device training a completely separate model, we maintain a **shared part** of the model (that benefits from multi-user data) and a **personal part** (that captures user-specific nuances). For example, Arivazhagan et al. (2019) propose splitting the neural network into a **base layer (global)** trained federatively on all users, and a **personalized layer** trained only on each user's device[6]. The global layers learn representations that are broadly useful (common syntax, general knowledge), while the last layer(s) are unique to the user, learning that user's writing style, favorite phrases, or domain-specific terms. This **parameter decoupling** allows the system to integrate *multi-user generalization* (through the shared base) with *per-user specialization*[6]. Crucially, when the global model is updated with new data from many users, it won't override each user's personal parameters – avoiding the "one size fits none" problem and the issue of wiping out personal quirks[7]. In practice, the model effectively becomes *"a distribution of versions"* across clients: all share a core model, but each has some custom components[8].

Recent research extends this idea using **mixture-of-experts and modular layers**. One approach (CoMiGS, 2024) trains some expert modules as **global generalists** (shared across all users) while keeping other experts as **local specialists** unique to each user[9][10]. During inference, the model can route each input between general or specialist experts as needed. This achieved strong performance by balancing collaboration and personalization in federated settings[11]. Similarly, clustering techniques can train **several global sub-models** for groups of users with similar data, rather than a single global model for all[6]. These strategies all share the same pattern: **multi-user knowledge is retained in shared parts of the model, while user-specific knowledge is isolated in personal parts**. This architectural principle is foundational for PLMs, ensuring that knowledge from the masses *jump-starts* the model without diluting each user's individuality.

Efficient On-Device Personalization (Fine-Tuning Stage)

Once a strong global model is in place, the next stage is **personalizing the model for each user**. Given that *"the memory and motivation of third party individuals is limited,"* the design must **minimize the effort and resources required** from each user. Instead of lengthy training sessions or large labeled datasets per user, we leverage **one-shot or few-shot tuning techniques** that are *lightweight* but effective. Two research-backed techniques stand out:

- **Low-Rank Adaptation (LoRA) and Adapters:** These are **parameter-efficient fine-tuning (PEFT)** methods that add small trainable weight matrices to the model and **freeze the rest of the parameters**. LoRA, for instance, inserts low-rank update matrices into each layer of the network; during training, only those matrices are updated, vastly reducing compute and data requirements[12][13]. This means a user can fine-tune the PLM with just a few examples or limited data, since the model doesn't need to update all billions of weights – just a tiny fraction. Research has shown LoRA can achieve near full fine-tuning performance while using far fewer resources[12]. In a federated setting, a recent framework called **FDLoRA** (2023) uses a dual-adapter approach: it places **two LoRA modules** on each device – one captures personal nuances, and one captures global knowledge to share[14]. Only the *global* adapter is sent to the server for aggregation, while the *personal* adapter stays on the device[14]. An adaptive fusion then combines them, so the user's model benefits from both personal tuning and community data[14]. This is an elegant design pattern: it **preserves privacy (personal adapter never leaves**

the device) and keeps communication low, yet still lets the *"masses"* collectively improve the global parts. Overall, adapter-based tuning allows quick personalization *"without incurring high communication and computing costs"* on the edge[14].

- **Prompt-Based Personalization:** Another minimal-effort approach is to personalize **at inference-time using prompts** rather than changing model weights. In this method, the system learns a short **"soft prompt"** (an embedding or prefix) that, when prepended to the user's inputs, biases the model towards that user's style or preferences[15]. Essentially, the model is given a *hint* or context about the user each time it generates text. This can be learned by a brief training phase on device: the PLM finds an optimal prompt that yields outputs matching the user's past writing. This **one-shot tuning** (since it's often a small number of prompt tokens) is very fast and requires little data[15]. It's akin to giving the model a quick memory of the user without altering its core weights[16]. For example, if a user tends to write in a casual tone with certain slang, the learned prompt steers the model to that tone. The advantage is that this doesn't require ongoing retraining – the prompt can be learned once (or updated occasionally) and used for all future interactions. Research notes we must secure these prompts (they are effectively a distilled version of a user's data) so that they don't leak sensitive info[16]. Nonetheless, prompt-based personalization is a powerful way to utilize limited user data and effort for tailoring model output.

Continuous Learning vs. One-Shot: Given limited user motivation, the PLM should not rely on constant feedback or extensive retraining. Instead, it can **update occasionally in small increments**. For instance, the device might fine-tune the personal adapter overnight when the phone is charging (using any new messages from that day as additional training data), which is a form of *continual learning* that is unobtrusive. However, as research points out, fully continuous on-device training can be risky – models might overfit to new user data and *"lose generality,"* or even forget earlier knowledge (catastrophic forgetting)[17]. The combination of techniques above mitigates this: using a strong global model as a foundation makes the model less prone to overfitting on tiny datasets[1], and tuning only small modules (adapters or prompts) limits how much the model can stray from its base behavior[18]. In practice, a **one-shot or occasional fine-tune** is preferred over continuous heavy updates, to best *"take advantage of limited energy"* and attention from users. Users might only need to provide a few example corrections or a small set of documents once, and the PLM adapts from that.

Multi-User Generalization and Model Update Cycle

Crucially, the PLM design enables **multi-user generalization** even after personalization. Because each user's device contributes to the global model (through FL updates or shared adapters), the improvements gained from one user's data can benefit others in a general sense. For example, if many users start using new slang or a new technical term, the global model will pick up on it and make that available to everyone's PLM (without leaking who used those terms). The shared model acts as a **common knowledge base** that all personal models draw from[8]. At the same time, personal customizations (like one user's unique contact names or quirks) remain local. The system periodically repeats the federated update cycle: devices train on new local data and send updates for aggregation. Thanks to the *decoupling of base and personal parameters*, these global updates can be applied **without overwriting personal adaptations**[7]. Each user thus runs a model that is *up-to-date* with the community's knowledge but still **retains their individual learned preferences**. In effect, we achieve a design where the PLM is both *"user-trained, user-owned"* (personalized on-device) and *collectively improved* via the masses[19].

From an architectural standpoint, this pattern marries the benefits of large-scale training and individualization: a **federated backbone model** provides general linguistic competence, and a **personal memory or model fork** provides customization. The framework can be extended to multi-modal or multi-persona scenarios as well (e.g. separate personas or domains for a user, each with its own data and adapter, all built on the same base model)[20], but the core principle remains the separation of shared vs. local knowledge.

Conclusion

In summary, a research-backed design pattern for personal language models involves: (1) **Crowdsourced data collection** via federated or collaborative training to learn a robust *base model* from the masses' knowledge while preserving privacy[4]; (2) **Edge computing for training**, so that model improvement happens on-device (no central raw data, low latency, user control)[21][22]; (3) **Parameter-efficient personalization** through adapters or prompts, enabling one-shot or lightweight fine-tuning that respects users' limited time and device constraints[15]; and (4) **A hybrid architecture** that maintains a global-generalized component and a personal-specialized component, striking a balance between common skills and individual preferences[6]. This approach ensures that *the output of a web application or user interaction* – e.g. chat logs, writing samples – can be fed back (in processed form) to continuously refine the personal model. By following this design, the **masses collectively jump-start the model**, and each user's PLM becomes increasingly accurate and tailored with minimal effort, all grounded in the latest AI research on federated personalization and on-device learning.

Sources: The solution integrates insights from federated learning surveys and personalization research, including how Google's Gboard uses FL for suggestions[4], methods like personal layers in federated models[6], mixture-of-expert schemes for collaboration vs. specialization[23][11], and efficient fine-tuning techniques such as LoRA adapters[14] and prompt tuning[15] that make on-device personalization feasible under limited data and energy constraints. These approaches form the foundation of the proposed PLM design pattern, offering a blueprint to collect training data and build personal language models that are both **general and personal** by design.

[1] [2] [6] Dual prompt personalized federated learning in foundation models | Scientific Reports

https://www.nature.com/articles/s41598-025-11864-4?error=cookies_not_supported&code=45eef247-4f66-46ee-9df7-535a8ee3f894

[3] Federated learning - Wikipedia

https://en.wikipedia.org/wiki/Federated_learning

[4] [7] [8] [13] [15] [16] [17] [18] [21] [22] Generative AI at the Edge: Challenges and Opportunities - ACM Queue

https://queue.acm.org/detail.cfm?id=3733702

[5] [12] [14] FDLoRA: Personalized Federated Learning of Large Language Model via Dual LoRA Tuning

https://arxiv.org/html/2406.07925v1

[9] [10] [11] [23] On-Device Collaborative Language Modeling via a Mixture of Generalists and Specialists | OpenReview

https://openreview.net/forum?id=rf0ZoDASnS

[19] [20] CES 2024: Announcing Personal AI MODEL-2: Multi-Persona, Multi-Modal, and Multi-Channel | by Suman Kanuganti | Personal AI

https://blog.personal.ai/ces-2024-announcing-personal-ai-model-2-cde0515384b8?gi=579cbd2b8a24