# [Supplementary Material] Learning to Collaborate: An Orchestrated-Decentralized Framework for Peer-to-Peer LLM Federation

**Inderjeet Singh**[†]**, Eleonore Vissaul-Gaudin, Andikan Otung, Motoyoshi Sekiya**

Fujitsu Research of Europe
Slough, United Kingdom
{inderjeet.singh, eleonore.gaudin, andikan.otung, motoyoshi.sekiya}@fujitsu.com

**Code Availability:** The complete source code for the KNEXA-FL framework, all experiments, and analysis scripts is publicly available at: https://github.com/FujitsuResearch/knexa-fl

## Detailed Algorithms

This section provides the precise algorithmic specifications for the Central Profiler/Matchmaker (CPM) and the multi-phase asynchronous training loop of KNEXA-FL. The presentation follows the notation introduced in the main paper.

### CPM Matchmaking Logic

Algorithm 1 details the contextual combinatorial bandit procedure used by the CPM to select up to $K_p$ disjoint peer-to-peer (P2P) interactions per round. The CPM evaluates every admissible tuple $(a_i, a_j, K_{\text{rec}}, R_{ij})$, comprising a sender $a_i$, a receiver $a_j$, the recommended knowledge-exchange protocol $K_{\text{rec}}$ (e.g., Adaptive Knowledge Distillation, AKD), and a role assignment $R_{ij}$, with an upper-confidence-bound (UCB) score. Pairs are greedily selected until either (i) the pair budget $K_p$ is reached or (ii) no agents remain unmatched.

### End-to-End Asynchronous Training Round

Algorithm 2 expands the high-level protocol (Algorithm 1 in the main paper) into an explicit four-phase sequence executed at every communication round $t \in \{1, \ldots, T\}$. All network communication is authenticated and encrypted (e.g., via mTLS) to guarantee the confidentiality and integrity of the exchanged profiles and knowledge packages.

## Experimental Setup Details

All experiments were conducted under a fixed random seed (`42`) to ensure reproducibility of data partitioning, model initialization, and matchmaking. The software stack included `PyTorch 2.3`, `Hugging Face Transformers 4.43`, and `Accelerate 0.29`.

Algorithm 1: CPM: LinUCB-based Matchmaking

```
Require: Profiles $\{\mathbf{p}_i^{(t)}\}_{i=1}^N$, LinUCB state $(\mathbf{A}, \mathbf{b})$, exploration $\beta$, pair budget $K_p$
 1: $\hat{\boldsymbol{\theta}} \leftarrow \mathbf{A}^{-1}\mathbf{b}$   ▷ Current estimate of reward model weights
 2: $\mathcal{C} \leftarrow \emptyset, \mathcal{E}_t \leftarrow \emptyset, \mathcal{U} \leftarrow \{a_1, \ldots, a_N\}$
 3: for all unordered pairs $(a_i, a_j)$ with $a_i, a_j \in \mathcal{U}$ do
 4:     for all admissible $(K_{\text{rec}}, R_{ij})$ do
 5:         $\mathbf{x} \leftarrow \varphi(\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t)}, K_{\text{rec}}, R_{ij})$   ▷ Construct context vector
 6:         $u \leftarrow \hat{\boldsymbol{\theta}}^\top \mathbf{x} + \beta\sqrt{\mathbf{x}^\top \mathbf{A}^{-1}\mathbf{x}}$   ▷ Calculate UCB score
 7:         Append $(a_i, a_j, K_{\text{rec}}, R_{ij}, u)$ to $\mathcal{C}$
 8: Sort $\mathcal{C}$ in descending order by UCB score $u$
 9: for all $(a_i, a_j, K_{\text{rec}}, R_{ij}, u) \in \mathcal{C}$ in order do
10:     if $|\mathcal{E}_t| < K_p$ and $a_i, a_j \in \mathcal{U}$ then
11:         Add $(a_i, a_j, K_{\text{rec}}, R_{ij})$ to $\mathcal{E}_t$
12:         $\mathcal{U} \leftarrow \mathcal{U} \setminus \{a_i, a_j\}$   ▷ Remove agents from available pool
13: return $\mathcal{E}_t$
```

Training was performed using `FP16` precision with automatic mixed-precision scaling.

### Core Hyperparameters

Table 1 summarizes the key hyperparameters shared across all real-model experiments.

Table 1: Global hyperparameter configuration for all experiments.

| **Category** | **Value / Specification** |
|---|---|
| Optimizer | AdamW (`betas=(0.9, 0.98),eps=1e-6`) |
| Learning Rate | $3 \times 10^{-5}$ with linear warmup (2% of steps) |
| Local Batch Size | 8 (gradient accumulated to an effective size of 32) |
| Local Epochs / Round | 1 |
| Communication Rounds ($T$) | 20 (for main experiments), 100 (for synthetic) |
| PEFT Method | LoRA, rank tuned per model (2.2–3.0% trainable) |
| KD Temperature $\mathcal{T}$ | 1.5–2.5 (linearly annealed over training) |
| KD Weight $\alpha_{\text{KD}}$ | 0.2 (initial) $\rightarrow$ 0.5 (final), linear schedule |
| Compute Hardware | NVIDIA A100 80GB or H100 90GB (mixed fleet) |
| Parallelism | One client per GPU; CPM on CPU (`Intel Xeon 6338`) |

[†]Corresponding author.

Algorithm 2: KNEXA-FL: Asynchronous Training Round $t$

```
Inputs: Agent set $\mathcal{A} = \{a_1, \ldots, a_N\}$, CPM $\mathcal{P}$, public transfer set $\mathcal{X}_u$

Phase 1 (Agent-side local updates, parallel):
1:  for all $a_i \in \mathcal{A}$ in parallel do
2:      Perform $E$ local epochs on private data $D_i$: $\phi_i \leftarrow \phi_i - \eta \nabla_{\phi_i} \mathcal{L}_i$.
3:      Prepare knowledge package $\kappa_i$ (e.g., logits on $\mathcal{X}_u$) and profile $\mathbf{p}_i^{(t)}$.
4:      Securely transmit profile $\mathbf{p}_i^{(t)}$ to $\mathcal{P}$.

Phase 2 (CPM-side matchmaking, periodic):
5:  On quorum or timer: obtain latest profiles $\{\mathbf{p}_i^{(t)}\}$; compute $\mathcal{E}_t$ via Alg. 1.
6:  Dispatch pairing directives to the involved agents.

Phase 3 (P2P knowledge exchange, parallel):
7:  for all $(a_s, a_r, K_{\text{rec}}, R_{sr}) \in \mathcal{E}_t$ in parallel do
8:      Establish secure, ephemeral channel; $a_s$ transmits $\kappa_s$ to $a_r$.
9:      $a_r$ integrates $\kappa_s$ using protocol $K_{\text{rec}}$ (AKD by default).

Phase 4 (Reward reporting & CPM update, asynchronous):
10: for all receiving agents $a_r$ do
11:     Compute reward $r_{sr}^{(t)}$ (Eq. 3 in main text) and send $(\mathbf{x}_{sr}^{(t)}, r_{sr}^{(t)})$ to $\mathcal{P}$.
12:     On receipt, $\mathcal{P}$ updates bandit model: $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{x}\mathbf{x}^\top$, $\mathbf{b} \leftarrow \mathbf{b} + r\mathbf{x}$.
```

## Six-Client Heterogeneous Federation

The primary empirical study (Sections 4-5 in the main paper) employs a federation of six heterogeneous clients. Table 2 lists the backbone models, parameter counts, and data partition sizes, which were generated using a Dirichlet distribution ($\alpha = 0.1$) over the combined `HumanEval+MBPP` training set.

Table 2: Client fleet configuration for real-model experiments. *Trainable %* denotes the proportion of parameters updated via LoRA.

| Client | Backbone Model | Total Params | Trainable % | Train / Val |
|---|---|---|---|---|
| C0 | `Qwen1.5-0.5B` | 475M | 2.39% | 45 / 12 |
| C1 | `Cerebras-GPT-590M` | 604M | 2.34% | 43 / 11 |
| C2 | `BLOOM-560M` | 572M | 2.20% | 44 / 12 |
| C3 | `Pythia-410M` | 418M | 3.01% | 46 / 12 |
| C4 | `Qwen1.5-0.5B` | 475M | 2.39% | 54 / 14 |
| C5 | `Cerebras-GPT-590M` | 604M | 2.34% | 44 / 11 |

## LinUCB CPM Simulation Details

To rigorously stress-test the contextual-bandit CPM in isolation, we executed large-scale synthetic simulations (up to 64 clients) following the design protocol in Lattimore and Szepesvári (2020). These experiments allow for a clean analysis of the learning algorithm's behavior, free from the confounding variables of real-world model training.

Figure 1 provides a detailed visualization of the learning dynamics, complementing the summary results presented in Figure 3 of the main manuscript. The plots for learning convergence (a) and cumulative regret (b) offer strong visual evidence for the efficacy of the LinUCB algorithm. They confirm that the CPM learns a near-optimal matchmaking policy over time, leading to monotonic performance improvements and outperforming the random baseline by a significant margin. The sub-linear regret curve for KNEXA-FL is characteristic of an efficient learning algorithm, contrasting sharply with the linear regret of the non-adaptive random strategy.

## Detailed Experimental Results

This section provides a more granular analysis of the experimental results, including per-client performance trajectories and a principled discussion of the baseline methods' failure modes.

### Aggregate Performance Summary

Table 3 consolidates the final global test performance after 20 communication rounds for all collaborative methods and after 12 isolated rounds for the `LocalOnly` baseline.

Table 3: Final average performance on the 116-problem global test set. Best global-test performance is in **bold**. Footnotes match the main paper.

| Method | Pass@1 (%) | Pass@5 (%) | Pass@10 (%) | CodeBLEU |
|---|---|---|---|---|
| LocalOnly | 2.22 | 5.42 | 5.55 | 0.260 |
| FedID-CentralKD | 1.11 | 5.56 | 5.56 | 0.181 |
| Central-KD[†] | 2.00 | 7.80 | 10.00 | 0.268 |
| Heuristic-P2P[‡] | 6.67 | 16.67 | 27.78 | 0.392 |
| Random-P2P | 8.89 | 22.40 | 27.80 | 0.239 |
| **KNEXA-FL** | **13.33** | **31.25** | **44.44** | **0.344** |

[†]`Central-KD` was volatile; peaked at 18.33% (6-client) but collapsed to 2.00% (4-client instability analysis). [‡]`Heuristic-P2P` was evaluated for restricted rounds and on data due to time/compute constraints, to ablate learning vs. a static heuristic.

### Baseline Instability and Inefficiency Analysis

**Central-KD: Catastrophic Collapse.** The `Central-KD` baseline, which forces all clients to distill from a single, globally-averaged teacher distribution, exhibited severe training instability. This phenomenon arises from an inherent incompatibility between global logit averaging and a highly heterogeneous federation. Forcing diverse, specialized PEFT-based models to conform to a single ensemble teacher leads to catastrophic forgetting, as locally acquired knowledge is overwritten by destructive updates. Table 4 quantifies this collapse by showing the dramatic drop from peak to final performance, confirming that this approach is fundamentally unreliable in heterogeneous settings.

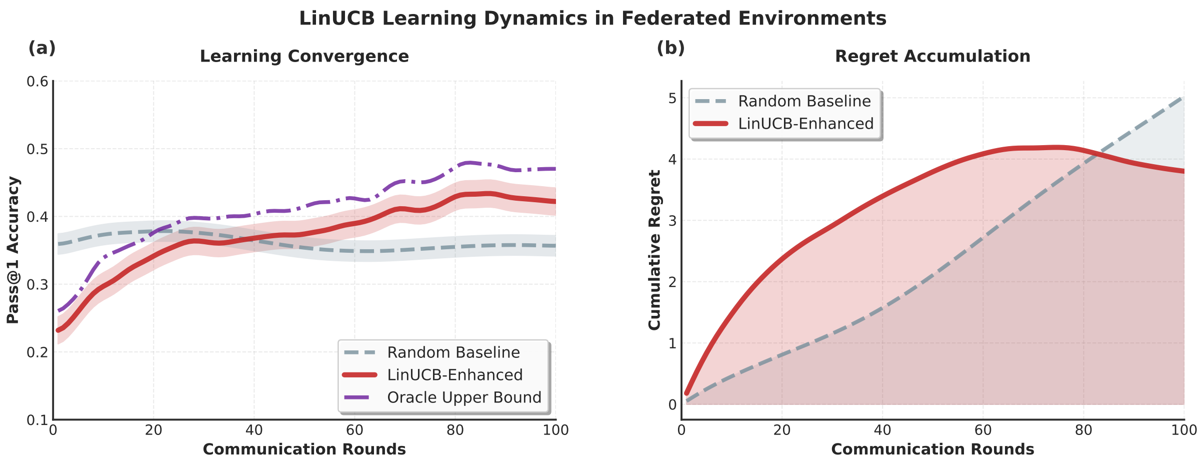

Figure 1: LinUCB-based CPM learning dynamics over 100 communication rounds in synthetic federated environments. (a) **Learning Convergence:** Pass@1 accuracy trajectories demonstrate the LinUCB algorithm's ability to learn optimal pairing strategies over time. The LinUCB-enhanced approach (red) shows steady improvement and convergence toward the oracle upper bound (purple dashed), while the random baseline (gray dashed) plateaus at suboptimal performance. The shaded regions represent 95% confidence intervals. (b) **Regret Accumulation:** Cumulative regret quantifies the performance gap between each method and the oracle policy. LinUCB's regret stabilizes after approximately 60 rounds, indicating convergence to a near-optimal policy, while the random baseline's regret grows linearly. The plateauing of LinUCB's regret curve confirms the algorithm's successful exploitation of learned pairing patterns. These synthetic experiments validate the theoretical foundations of our contextual bandit approach and demonstrate why intelligent matchmaking is crucial for maximizing collaborative gains in heterogeneous federated learning environments.

Table 4: Performance collapse of the `Central-KD` baseline, comparing peak Pass@1 (%) to final Pass@1 after 14 rounds on the global test set.

| Client (Model) | Peak Pass@1 | Final Pass@1 | Degradation |
|---|---|---|---|
| C0 (Qwen-0.5B) | 13.33% | 0.00% | -13.33 pp |
| C1 (Cerebras-590M) | 0.00% | 0.00% | 0.00 pp |
| C2 (BLOOM-560M) | 20.00% | 0.00% | -20.00 pp |
| C3 (Pythia-410M) | 40.00% | 8.00% | -32.00 pp |
| **Average** | **18.33%** | **2.00%** | **-16.33 pp** |

**Random-P2P: Inefficiency of Unguided Collaboration.** The `Random-P2P` baseline avoids catastrophic collapse but demonstrates the statistical inefficiency of unguided collaboration. Its final Pass@1 of 8.89% is substantially lower than the 13.33% from KNEXA-FL. This gap highlights the value of the CPM: random pairings are unlikely to consistently identify the most synergistic knowledge transfers, leading to slower convergence and a lower overall performance ceiling. The CPM's learned policy is the decisive factor that elevates KNEXA-FL beyond both the instability of centralization and the inefficiency of randomness.

## Per-Client Improvements under KNEXA-FL

Table 5 reports the individual Pass@1 improvements for each client, comparing their final KNEXA-FL performance to their `LocalOnly` starting point. The results show that all six clients benefit significantly, with the most dramatic gains seen by Client C2. This confirms that CPM-guided collaboration effectively raises the performance of the entire federation, rather than merely amplifying the strongest members.

Table 5: Per-client Pass@1 (%) on the global test set. Absolute improvement ($\Delta$) is relative to the `LocalOnly` baseline.

| Client | LocalOnly (%) | KNEXA-FL (%) | Improvement ($\Delta$) |
|---|---|---|---|
| C0 (Qwen) | 2.22 | 11.11 | **+8.89 pp** |
| C1 (Cerebras) | 0.00 | 6.67 | **+6.67 pp** |
| C2 (BLOOM) | 6.67 | 36.67 | **+30.00 pp** |
| C3 (Pythia) | 0.00 | 8.89 | **+8.89 pp** |
| C4 (Qwen) | 2.22 | 10.00 | **+7.78 pp** |
| C5 (Cerebras) | 0.00 | 7.78 | **+7.78 pp** |

## Expanded Reproducibility Checklist

To ensure full transparency and replicability, this supplementary material provides the following artifacts:

- **Detailed Algorithms:** Complete pseudocode for both the CPM matchmaking logic and the end-to-end asynchronous training round are provided in Algorithms 1 and 2.
- **Comprehensive Hyperparameters:** The full training configuration, including optimizer settings, learning rate schedules, and hardware specifications, is detailed in Table 1.
- **Federation Configuration:** The precise specification of the heterogeneous client fleet, including backbone models and data partitions, is provided in Table 2.

- **Granular Results:** Detailed tables showing per-client performance gains (Table 5) and the quantitative collapse of the centralized baseline (Table 4) are included.
- **Public Code Release:** All source code, experiment scripts, configuration files, and scripts to reproduce the synthetic benchmarks are publicly available at https://github.com/FujitsuResearch/knexa-fl.

## Rebuttal Additions and Protocol Details

To address reviewer feedback and ensure full reproducibility, this section provides explicit details on the CPM's inputs, the disjoint-pair matching algorithm, and the implementation of our new baselines.

### CPM Context Vector and Reward Normalization

As promised in our rebuttal, we specify the construction of the CPM's inputs and reward signal, as referenced in Section 3.3 of the main paper.

**Context Vector Construction.** The abstract agent profile $\mathbf{p}_i \in \mathbb{R}^{d_p}$ submitted by agent $a_i$ is a concatenation of features. For our experiments, $d_p = 32$. The profile includes:

- **Static Features (8 dims):** One-hot encodings of the agent's LLM backbone family (e.g., Qwen, Pythia, BLOOM).
- **Dynamic Performance Features (12 dims):** Recent local validation metrics (e.g., Pass@1, CodeBLEU, perplexity) and their deltas from the previous round.
- **Data Distribution Features (12 dims):** A normalized histogram representing the agent's local data distribution over $k = 12$ problem categories (e.g., string manipulation, algorithms, data structures).

When the CPM considers a pair $(a_i, a_j)$, the final context vector $\mathbf{x}_{ij} \in \mathbb{R}^{2d_p}$ is constructed by concatenating the individual profiles $\mathbf{x}_{ij} = [\mathbf{p}_i, \mathbf{p}_j]$. This simple concatenation allows the LinUCB model's linear weights $\hat{\boldsymbol{\theta}}$ to learn the cross-agent feature interactions.

**Reward Normalization.** The raw reward signal $r_{ij}^{(t)}$ from Eq. 3 (main paper) can be volatile. To stabilize the LinUCB algorithm, we apply running z-score normalization. The CPM maintains a running mean $\mu_r$ and standard deviation $\sigma_r$ of all rewards it has observed. The reward $r_{ij}^{(t)}$ reported by agent $a_i$ is normalized before being used to update the bandit model:

$$r_{\text{norm}}^{(t)} = \frac{r_{ij}^{(t)} - \mu_r}{\sigma_r + \epsilon}$$

where $\epsilon = 10^{-6}$ is a small constant for numerical stability. This ensures the reward signal has zero mean and unit variance, preventing large reward values from disproportionately skewing the bandit's parameter updates.

### Baseline Algorithm Specifications

We provide the algorithmic details for the new baselines added for the camera-ready version.

Algorithm 3: Heuristic-P2P (Hetero-Greedy) Pairing

```
Require: Client set $\mathcal{C} = \{a_1, \dots, a_N\}$, per-client data distributions $\{d_i\}$, per-client recent performance $\{u_i\}$, pair budget $K_p = \lfloor N/2 \rfloor$.
 1: Initialize candidate list $\mathcal{S} \leftarrow \emptyset$, matched pairs $\mathcal{E}_t \leftarrow \emptyset$, and set of available agents $\mathcal{U} \leftarrow \mathcal{C}$.
 2: for all unordered pairs $(a_i, a_j)$ with $a_i, a_j \in \mathcal{U}$ do
 3:     Compute score $s_{ij} = \text{JS}(d_i, d_j)$.
 4:     Add $(a_i, a_j, s_{ij})$ to $\mathcal{S}$.
 5: Sort $\mathcal{S}$ by score $s_{ij}$ in descending order.
 6: for all sorted tuple $(a_i, a_j, s_{ij})$ in $\mathcal{S}$ do
 7:     if $|\mathcal{E}_t| < K_p$ and $a_i \in \mathcal{U}$ and $a_j \in \mathcal{U}$ then
 8:         Assign roles: $a_t \leftarrow \arg\max(u_i, u_j)$, $a_s \leftarrow \arg\min(u_i, u_j)$.
 9:         Add directed pair $(a_s, a_t)$ to $\mathcal{E}_t$.
10:         $\mathcal{U} \leftarrow \mathcal{U} \setminus \{a_i, a_j\}$.
11: return $\mathcal{E}_t$.
```

**FedID-CentralKD.** This baseline, adapted from `FedID` (Ma et al. 2023), was implemented as described in the main paper's 'Baselines' paragraph (Section 4.1). A central `Qwen-0.5B + LoRA` model was trained for 20 rounds using text-level KD. The ensemble teacher signal was created by collecting decoded text from all 6 clients on public prompts, then performing confidence-weighted majority voting over normalized code sequences. The central model's validation loss on a private held-out set steadily decreased, but as shown in Table 4 in the main paper, this central knowledge failed to transfer effectively to the heterogeneous clients, resulting in 1.11% Pass@1.

**Heuristic-P2P (Hetero-Greedy).** This baseline replaces the CPM's learned policy (Algorithm 2) with a static, non-learning heuristic. The algorithm, promised in our rebuttal, is specified in Algorithm 3. It greedily selects disjoint pairs that maximize JS divergence of their data distributions. As reported in Table 4, this approach (6.67% Pass@1) was detrimental, performing worse than random pairings (8.89% Pass@1).

## References

Lattimore, T.; and Szepesvári, C. 2020. *Bandit Algorithms*. Cambridge University Press.

Ma, X.; Liu, J.; Wang, J.; and Zhang, X. 2023. FedID: Federated Interactive Distillation for Large-Scale Pretraining Language Models. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8566–8577. Singapore: Association for Computational Linguistics.

## Reproducibility Checklist

**This paper:**

- Includes a conceptual outline and/or pseudocode description of AI methods introduced: **Yes**. See the 'The KNEXA-FL Framework' section and the pseudocode for the KNEXA-FL Protocol.

- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results: **Yes**.
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper: **Yes**.

## Does this paper make theoretical contributions? Yes.

- All assumptions and restrictions are stated clearly and formally: **Yes**. See the 'Security, Privacy, and Theoretical Insight' subsection.
- All novel claims are stated formally (e.g., in theorem statements): **Partial**. We formally state theoretical claims but do not introduce new theorems.
- Proofs of all novel claims are included: **NA**. We synthesize existing, cited theoretical results.
- Proof sketches or intuitions are given for complex and/or novel results: **Yes**. See the 'Security, Privacy, and Theoretical Insight' subsection.
- Appropriate citations to theoretical tools used are given: **Yes**.
- All theoretical claims are demonstrated empirically to hold: **Yes**.
- All experimental code used to eliminate or disprove claims is included: **NA**.

## Does this paper rely on one or more datasets? Yes.

- A motivation is given for why the experiments are conducted on the selected datasets: **Yes**. See the 'Experimental Setup' subsection.
- All novel datasets introduced in this paper are included in a data appendix: **NA**. We use a novel combination of existing public datasets.
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes: **NA**.
- All datasets drawn from the existing literature are accompanied by appropriate citations: **Yes**.
- All datasets drawn from the existing literature are publicly available: **Yes**.
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing: **NA**.

## Does this paper include computational experiments? Yes.

- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting: **Partial**. Final hyperparameters are listed in the Appendix; the full search space will be in the supplementary material.
- Any code required for pre-processing data is included in the appendix: **Yes**. Code will be provided with the supplementary material.
- All source code required for conducting and analyzing the experiments is included in a code appendix: **Yes**.
- All source code required for conducting and analyzing the experiments is available at https://github.com/FujitsuResearch/knexa-fl with a license that allows free usage for research purposes: **Yes**.
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from: **Yes**.
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results: **Yes**. See the 'Experimental Setup' subsection.
- This paper specifies the computing infrastructure used for running experiments (hardware and software): **Yes**. See the 'Experimental Setup' subsection. Full software versions will be provided with the code.
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics: **Yes**. See the 'Experimental Setup' subsection.
- This paper states the number of algorithm runs used to compute each reported result: **Yes**. Details are in the 'Profiler Ablation' subsection.
- Analysis of experiments goes beyond single-dimensional summaries of performance to include measures of variation, confidence, or other distributional information: **Yes**. See the synthetic ablation study figure.
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests: **No**. We rely on the substantial magnitude of performance differences and confidence intervals.
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments: **Yes**. A comprehensive list will be provided in the Appendix.