A Appendix

A.1 Hyperparameter Sensitivity Analysis

We introduce some hyperparameters in RecNet. τ is the hyperparameter that control the similarity of selected client agent, while Updatesize is to control the number of updated users or items between each time of preference propagation. To evaluate the sensitivity of our method to these hyperparameters, we vary their values while keeping all other settings fixed and optimal, and observe the resulting impact on model performance. In Figure 7, we present the effects of two key hyperparam- eters, on model performance across two datasets. These results indicate that RenNet achieves consistent improvements in a relatively stable range of hyperparameters, and achieves robustness of hyperparameter selection toward different datsets.
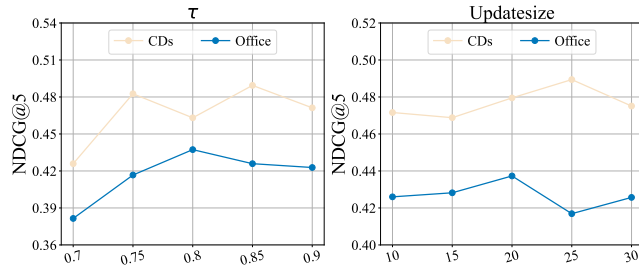

Figure 7: Hyperparameter Sensitivity Analysis w.r.t. different τ and Updatesize.

A.2 Prompts Used in RecNet

In this section, we present the concise versions of the prompts used in RecNet. Detailed prompt construction can be viewed in our code repository: https://anonymous.4open.science/r/RecNet-8A05.

Prompt$_{extract}$

Instruction:
Given a user or item profile, extract a structured set of fine-grained preference attributes or feature attributes from the profile. Each extracted attribute should reflect an explicit or implicit preference, interest, or feature mentioned in the text.

Input:
{User Profile / Item Profile}

The reasoning process should include:
1. Reading and understanding the profile to identify explicit and implicit preferences, interests, or features.
2. Categorizing the identified information into fine-grained attributes.
3. Inferring reasonable attributes when they are not explicitly stated but are strongly implied by the text.

Important Notes:
1. The extraction must be based strictly on the provided profile content.
2. Do not include explanations, Markdown syntax, or code block indicators in the output.
3. Each attribute category should be a key, and its value should be a list of attribute entities.
4. If no attributes are found for a category, output an empty list for that category.
5. All inferred attributes must be reasonable and clearly grounded in the profile text.

Output Format:
{"attribute_1": …, "attribute_2": …, …}

Prompt$_{summarize}$

Instruction:
Given an extracted list of fine-grained attributes and a router's preference profile, generate a concise and coherent textual summary that represents the overall preference pattern of a community of users and items. The summary should capture high-level concepts and implicit commonalities among the attributes, and be suitable for use as a routing embedding descriptor in a recommendation system.

Input:
{Attribute List}
{Router Profile}

The reasoning process should include:
1. Reviewing the fine-grained attribute list to identify recurring themes and dominant preference signals.
2. Analyzing the router profile to understand the current routing focus and alignment goals.
3. Abstracting high-level concepts from detailed attributes to form a coherent community-level preference pattern.

Important Notes:
1. The summary must be grounded strictly in the provided attribute list and router profile.
2. Do not add preambles, explanations, or Markdown formatting in the output.
3. The output should be concise, coherent, and focused on high-level preference patterns.
4. Avoid copying raw attribute lists; instead, abstract and generalize them.

Output Format:
{"summary": … }

Prompt$_{gradient}$

Instruction:
Given the router profile list, filter memory, original and merged user profiles, candidate items, the selected item,

and the ground-truth item, please perform a textual credit assignment and gradient analysis to evaluate how each module contributed to the final prediction. For textual credit assignment, please give a textual judgement for each module. For gradient analysis, please provide an actionable revision suggestion for improvement of each module.

Input:
{Router Profile List}
{Filter Memory}
{Original User Profile}
{Merged User Profile}
{Candidate Item Profiles}
{Selected Item}
{Ground-truth Item}

The reasoning process should include:
1. Comparing the {Selected Item} with the {Ground-truth Item} to assess prediction outcome consistency.
2. Evaluating each module ({Merged User Profile}, {Filter Memory}, and each router profile in {Router Profile List}) by describing its influence on the final prediction in textual terms. Indicate whether the impact is positive, neutral, or negative and explain briefly.
3. Identifying potential adjustments or refinements for each module based on its evaluation. Clearly describe modifications that could improve performance, or explicitly state "no adjustment" if none is needed.
4. Analyzing router profiles individually, even if multiple routers collectively contribute to the prediction, and provide consistent recommendations for each.

Important Notes:
1. Base all evaluations strictly on the provided inputs and the comparison between {Selected Item} and {Ground-truth Item}.
2. Do not fabricate information about modules; if impact is unclear, provide a cautious textual assessment.
3. Ensure every module has both a textual evaluation and a corresponding suggestion.
4. Suggestions should be detailed and actionable.
5. Maintain clarity and coherence so the output can be directly used in the next iteration of system optimization.

Output Format:
{"Merged User Profile Evaluation": ... , "Filter Memory Evaluation": ... , "Router Profiles Evaluation": [...]}

Prompt$_{merge}$

Instruction:
Given multiple router profiles with aggregated preference signals, an internal filter memory, and an original user or item profile, generate an updated and refined profile that integrates meaningful new preference information while preserving the core identity of the original profile.

Input:
{Router Profile List}
{Filter Memory}
{Original Profile}

The reasoning process should include:
1. Analyzing the router profiles to identify new preference signals relevant to the original profile.
2. Using the Filter memory to filter out outdated, contradictory, or unreliable information.
3. Selecting only those signals that semantically align with or meaningfully complement the original profile.
4. Integrating consistent new preferences in a natural and concise manner, avoiding direct copying.
5. Producing a refined profile that reflects an evolution of the original preferences rather than a replacement.

Important Notes:
1. The integration must preserve the structural organization and linguistic style of the original profile.
2. Do not introduce unrelated or fabricated preferences.
3. All merged information should be coherent and consistent with the existing preference identity.
4. Ensure the updated profile maintains individuality while incorporating new signals.
5. The result should be directly usable as the next iteration of the profile in the system.

Output Format:
{"updated_profile": ... }

Prompt$_{predict}$

Instruction:
Given a user profile describing the user's preferences and dislikes, predict which item the user would select from the provided candidate items.

Input:
{User Profile}
{Candidate Item Profiles}

The reasoning process should include:
1. Extracting the user's preferences and dislikes from the profile.
2. Evaluating each candidate item in relation to these preferences.
3. Selecting the item that best aligns with the user's inferred preferences, and providing a detailed explanation for the choice.

Important Notes:
1. The reasoning must be based strictly on the provided

user profile, without fabricating preferences.

2. If no specific preference is mentioned, general knowledge about the items may be used.

3. Exactly one item must be selected, and the explanation should be specific, explicitly linking the user's preferences to the features of the candidate items.

Output Format:

{"reasoning" : ... , "prediction" : ... }

Prompt*optimizer*

Instruction:

Given the following system components and feedback signals, optimize each module to generate an updated version for the next iteration of the system. The goal is to improve the overall prediction performance by refining preference profiles, filter memory, and router agents based on evaluation and textual gradient suggestions.

Input:

{User Profile} or {Filter Memory} or {Router Profile List}+{Router Profile}

{Textual Gradient}

The reasoning process should include:

1. Analyzing the suggestions(textual gradient) to identify strengths and weaknesses of each module.

2. For preference profile, refining the preference profile by enhancing, adjusting, or confirming features that reflect the inferred preferences based on suggestions. You should output the updated profile.

3. For filter memory, please update by modifying, reordering, or rewriting information fusion rules to improve filtering accuracy. You should output the whole updated filter memory.

4. For router profile, please evaluate each router agent and deciding whether to apply rewrite, split, or merge operations based on the suggested improvements. You should output the final operation and the updated router profile. If you choose merging, please also update the merged router profile within the router profile list.

Important Notes:

1. All optimizations must be based strictly on the provided rewards and textual gradient suggestions.

2. Do not introduce unrelated content; preserve the structural integrity of each module.

3. If no modification is required for a module, explicitly output "no adjustment."

4. Router agent operations must respect the current router profile structure and use the provided Router Profile List when modifications are applied.

5. Each updated module should clearly reflect the recommended improvements and be immediately usable in the

optimization loop.

Output Format:

{"Type": … , "Operation": … , "Updated Content": … }