arXiv:2608.10447v1 [cs.IR] 11 Aug 2026

Towards Efficient Reasoning in LLM-Based Recommender Systems via Model Merging

Linh Dieu Le
The University of Queensland
Brisbane, Australia

Tong Chen
The University of Queensland
Brisbane, Australia

Shazia Sadiq
The University of Queensland
Brisbane, Australia

Hongzhi Yin
The University of Queensland
Brisbane, Australia

Ming Jin
Griffith University
Brisbane, Australia

Junliang Yu*
Griffith University
Brisbane, Australia

Abstract

Large language model-based recommender systems are increasingly adopting slow-thinking models that generate step-by-step reasoning before making predictions, often achieving higher accuracy than fast-thinking models that predict directly. However, their reasoning traces are often unnecessarily verbose, increasing inference costs without commensurate accuracy gains. Existing training-based approaches to reasoning compression often incur substantial adaptation costs, while inference-time methods are brittle and difficult to scale. These limitations motivate model merging as a promising training-free direction for transferring specialised behaviours between models in a shared parameter space. In particular, merging a slow-thinking model with a fast-thinking counterpart provides a natural mechanism for balancing recommendation accuracy and reasoning conciseness. To this end, we propose, to our knowledge, the first model merging framework for reasoning compression in recommender systems. Unlike conventional merging methods that apply uniform merge coefficients across model components, our method performs fine-grained merging at the level of individual attention heads, capturing heterogeneous patterns in recommendation reasoning. Each attention head is assigned a distinct merge coefficient according to its contribution to critical reasoning evidence and its sensitivity to parameter change, enabling selective injection of the concise behaviour of the fast-thinking model into the slow-thinking model and reducing reasoning verbosity without compromising recommendation quality. Experiments on three benchmark datasets show that our method reduces reasoning length by up to 24.3% while outperforming competitive model merging baselines in maintaining recommendation accuracy. The code is available at https://github.com/linhledieu/REAM.

1 Introduction

Large language models (LLMs) have become powerful foundations for recommender systems, enabling richer preference modelling from textual signals such as user histories, item descriptions, and reviews [26, 43]. Early LLM-based recommenders typically formulate recommendation as a direct prediction task, estimating user preferences from user–item context without explicit deliberate thinking, supporting fast inference [3, 4, 25]. Building on these approaches, slow-thinking recommenders, echoing dual-process cognition [19], incorporate Chain-of-Thought (CoT) reasoning to accumulate evidence before producing a rating [7, 21, 22, 38]. Figure 1 illustrates this distinction: fast-thinking recommendation maps the input directly to a rating, whereas slow-thinking recommendation first constructs a structured reasoning trace. This additional deliberation can improve accuracy, particularly when preference signals are sparse or implicit [24], but often produces unnecessarily long traces regardless of input complexity, increasing inference latency and computational cost without commensurate benefits [34].

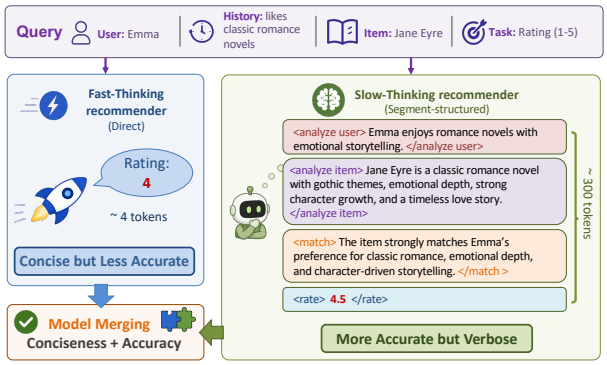


Figure 1: Motivation for merging fast- and slow-thinking recommenders toward concise and reliable prediction.

Consequently, reducing reasoning verbosity in slow-thinking recommender systems remains a critical open challenge. Existing approaches to reasoning compression fall into two categories, but both face limitations in efficiency or robustness. Training-based methods, such as distillation and length-penalised optimisation, can shorten reasoning traces but require additional data construction and model adaptation, limiting their practical scalability [13, 28]. By contrast, inference-time methods avoid retraining by imposing token budgets or instructing the model to reason in shorter steps [12, 46], yet their effectiveness can vary across inputs. These limitations motivate model merging as a training-free alternative that transfers behaviours by combining models in parameter space, commonly through task vectors—the arithmetic differences between fine-tuned and pretrained weights [17]. Recommender systems provide a natural testbed for this approach: fast- and slow-thinking recommenders share the same preference-prediction objective but differ in the amount of reasoning generated before reaching a prediction [4, 22]. Merging them may therefore transfer the concise generation behaviour of the fast-thinking model while preserving the reasoning required for accurate recommendation.

However, whether model merging can balance accuracy and efficiency in recommender systems remains unexplored, as prior work

*Corresponding author. Email: junliang.yu@griffith.edu.au

on reasoning compression has focused mainly on mathematical and coding tasks [42, 50]. Extending these methods to recommendation requires a more selective form of compression, since not all parts of a reasoning trace are equally safe to shorten: redundant elaboration may be reduced, whereas disrupting essential user–item evidence can directly alter the final prediction. Yet existing merging methods provide little control over how the reasoning process is altered. Wu et al.'s empirical analysis [42] shows that task-vector merging performance varies substantially with the choice of a single global coefficient. Seeking finer control, ACM [50] assigns a separate weight to each layer based on how similarly the two models behave, but its criterion does not explicitly account for how these updates reshape the reasoning trace. Taken together, existing work offers no reasoning-grounded basis for determining whether a shorter trace reflects useful compression or the loss of decision-critical evidence. Accordingly, to preserve recommendation accuracy, model merging must achieve selectivity at the component level, raising a central question: which components support the reasoning underlying the final prediction, and how should the fast-thinking update be allocated across them?

To determine where the fast-thinking update should be constrained, we first need a mechanistic signal of component-level reasoning importance. Prior work shows that reasoning-relevant behaviour is not distributed uniformly across the model but can be localised to a sparse subset of attention heads [36, 44]. This pattern is especially pronounced in GRPO-trained slow-thinking models such as DeepSeek-R1 [11], where reasoning-critical behaviour may emerge within only a small set of heads [32]. Together, these findings identify the attention head as the appropriate unit of analysis: heads serve distinct functional roles, and a small subset can contribute disproportionately to structured reasoning. At this granularity, a head's importance can be inferred from how it governs information flow during generation, determining which parts of the input and preceding trace shape the model's next output. Accordingly, this importance is especially consequential in slow-thinking recommender systems, where heads determine which user–item evidence enters the developing reasoning trace and which parts of that trace ultimately shape the final rating.

Building on this principle, we propose **REAM** (**R**easoning-H**E**ad-**A**ware **M**erging), a model merging method for reasoning compression in recommender systems. REAM formulates merging as selective transfer: it imports conciseness from the fast-thinking model while preserving the reasoning behaviour that supports the slow-thinking model's predictions. Rather than applying the fast-thinking update indiscriminately, REAM assigns each attention head a separate coefficient based jointly on its reasoning importance and sensitivity to parameter change. This constrains updates to heads critical for reasoning and prediction quality, while allowing less reasoning-critical and less sensitive heads to absorb more of the fast-thinking update. The resulting model requires no additional training or changes to decoding.

Our main contributions are as follows:

- We formulate reasoning compression in slow-thinking recommender systems as a selective merging problem, where the fast-thinking update is allocated according to the contribution of each model component to recommendation performance.
- We introduce REAM, to our knowledge, the first model merging framework for reasoning compression in recommender systems. By operating at the level of individual attention heads, REAM selectively transfers fast-thinking behaviour while limiting disruption to the heads most important for recommendation quality.
- Experiments on Yelp, Amazon Book, and Amazon Music datasets show that REAM can substantially reduce reasoning length by up to 24.3% while maintaining recommendation accuracy.

2 Related Work

2.1 LLM-based Recommender Systems

LLM-based recommender systems have progressed from direct prediction and knowledge augmentation toward explicit reasoning. Early work such as P5 [9] unified recommendation tasks within a text-to-text paradigm, while InstructRec [53] formulated recommendation as instruction following, enabling models to adapt their predictions to task-specific instructions. Subsequent systems such as TALLRec [4], LLaRA [25], and BIGRec [3] further adapted general-purpose language models to recommendation-specific settings, improving preference prediction across diverse tasks. Alongside these LLM-centric approaches, a complementary line augmented conventional recommender systems with LLM-derived knowledge or collaborative signals [35]. Despite these advances, these methods struggle in cold-start settings, where limited interaction histories provide insufficient grounding for accurate recommendations [24]. To address this limitation, recent work introduces explicit reasoning before prediction, enabling models to infer latent relationships between user preferences and item characteristics. For instance, RDRec [37] distils CoT rationales from review text; EXP3RT [21] and Reason4Rec [7] perform multi-step reasoning over user preferences and item attributes; and RecZero [22] demonstrates that structured recommendation reasoning can emerge through reinforcement learning (RL) alone. Collectively, these methods suggest that explicit reasoning is particularly valuable for inferring preferences from sparse or implicit user–item signals, but often produces unnecessarily long traces and exacerbates overthinking [34, 54].

2.2 Efficient Reasoning for LLMs

Efficiently shortening reasoning traces remains an open challenge. Training-based methods shorten reasoning via rationale distillation [37], length-penalised fine-tuning [28], RL with explicit length rewards [1, 13], or token-level pruning [45]. Their efficiency gains, however, incur adaptation costs: distillation depends on labelled or teacher-generated rationales, while fine-tuning, RL, and pruning require further target-model optimisation. By contrast, inference-time methods—including token-budget-aware generation [12], chain drafting [46], and cognitive-inspired sketching [2]—avoid retraining by shortening or restructuring reasoning during generation. However, they often depend on predefined budgets, prompting strategies, or reasoning formats, making their effectiveness sensitive to input and task difficulty. Within recommender systems, efficiency methods largely follow the training-based route but tend to replace explicit reasoning rather than compress it. SCoTER transfers reasoning patterns offline into a structure-preserving architecture that removes LLM inference at deployment [18], while LatentR^3 replaces CoT with RL-learned latent tokens [55]. Although effective in

reducing inference cost, these approaches may degrade prediction quality, interpretability, and adaptation to user context [8, 21].

2.3 Model Merging

Model merging transfers capabilities by combining model weights, often without additional training [48]. Early work focused on checkpoint averaging: Model Soups [41] averaged checkpoints fine-tuned from a shared base, while Task Arithmetic [17] represented fine-tuning updates as composable task vectors. To reduce interference between task vectors, TIES-Merging [47] trims small updates and resolves sign conflicts, whereas DARE [52] drops and rescales parameter deltas. Another line derives non-uniform coefficients to control each component's contribution: AIM [31] uses activation magnitudes to preserve important base-model weights, while SensMerging [27] adjusts coefficients according to parameter sensitivity within and across tasks. For reasoning compression, L2S-Merge [42] combines slow- and fast-thinking models to shorten responses while preserving accuracy, while ACM [50] pursues the same objective through layer-wise coefficients derived from activation statistics. Other work targets different objectives: RCP-Merging [49] preserves long-CoT capability when integrating domain-specific knowledge, whereas RAIN-Merging [16] uses variation in attention to instruction-relevant tokens to guide coefficient selection for instruction following. Model merging has also been explored in recommender systems, but for different objectives. An early work used Fisher-weighted merging to combine contrastive sequential recommenders [33], while more recent work merges models for multi-domain, cross-domain, federated, and generative recommendation [6, 14, 20, 39, 51]. These methods consolidate recommendation knowledge rather than compressing explicit reasoning.

3 Preliminaries

3.1 Task Setting

In this work, we study model merging for rating prediction in recommender systems. Let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items. Given a user $u \in \mathcal{U}$, their interaction history, and a target item $i \in \mathcal{I}$, the goal is to predict the rating $\hat{r} \in \mathbb{R}$ that u would assign to i. Unlike direct-prediction LLM recommenders [3, 4, 25], slow-thinking recommenders first generate an explicit reasoning trace before producing $\hat{r}$ [7, 21, 22]. This intermediate trace introduces overhead, requiring a long sequence before the final rating.

Following RecZero [22], we organise the generated output into four sequentially conditioned segments: `<analyze_user>` for user-preference analysis, `<analyze_item>` for item-characteristic analysis, `<match>` for user–item compatibility reasoning, and `<rate>` for the final numerical prediction. These form the USER, ITEM, MATCH, and RATE segments; the first three constitute the reasoning trace, while $\hat{r}$ is generated in RATE conditioned on it. This structure is adopted for its discrete regions, reflecting staged reasoning in recent slow-thinking recommenders [7, 21]; REAM adapts to other formats by redefining these regions accordingly (Appendix A.7).

3.2 Task Vectors and Model Merging

A *task vector* [17] is the weight difference $\Delta = \theta - \theta_B$ between a fine-tuned model θ and a base model θ_B of the same architecture, so parameters occupy identical positions. Task vectors relative to the same reference can therefore be combined as

$$\theta_{\text{merged}} = \theta_B + \sum_{m=1}^{M} \lambda_m \Delta_m,$$

where Δ_m is the update for the m-th model and $\lambda_m \in [0, 1]$ controls its contribution. We instantiate this framework with two recommendation models represented relative to the same reference θ_B:

- **Slow-thinking model** θ_S (RecZero [22]): generates three structured reasoning segments followed by a RATE segment. Its task vector is $\Delta_S = \theta_S - \theta_B$.
- **Fast-thinking model** θ_F (TALLRec [4]): predicts ratings without intermediate reasoning. We adapt TALLRec from binary like/dislike prediction to direct rating prediction and fully fine-tune it so that its parameter differences are expressed over full weight matrices, matching our merging granularity (Appendix A.1). Its task vector is $\Delta_F = \theta_F - \theta_B$.

The two models share the same rating-prediction objective but differ in how they reach a prediction (see Appendix A.2 for examples). Section 4 specialises this formulation by using θ_S as the anchor and partitioning the fast-thinking update into components indexed by b, with α_b controlling how much of each component is merged.

4 Proposed Method

To transfer concise generation behaviour from a fast-thinking recommender to a slow-thinking one while preserving the reasoning that supports its predictions, we develop our method by investigating three questions (Figure 2 summarises the resulting pipeline): (1) which attention heads are most important to the reasoning underlying the final prediction (§4.1)? (2) How sensitive is each head to the fast-thinking update (§4.2)? (3) How should the update be allocated to improve conciseness while limiting degradation in reasoning and recommendation quality (§4.3)?

4.1 Assessing Head-Level Reasoning Importance

Before allocating the fast-thinking update, we first assess each head's importance to the reasoning underlying the final prediction, since reasoning-critical heads may require more conservative updates to preserve recommendation quality. Since attention heads govern information flow during generation, their importance may differ across reasoning stages. Inspired by Wu et al. [44], we posit that a head's importance may be reflected in its ability to retrieve relevant user–item evidence from the input context and previously generated trace to support the developing reasoning; we define this importance as *retrieval criticality*. Later, at the RATE step, we posit that its importance may instead arise from connecting the final prediction to the preceding reasoning, attending to key conclusions established in the trace; we refer to the strength of this connection as *decision faithfulness*. These two complementary signals can serve as proxies for each head's reasoning importance.

4.1.1 Retrieval Criticality. We operationalise retrieval criticality by measuring how frequently each head retrieves segment-relevant evidence throughout the reasoning trace. Frequent retrieval indicates that the head repeatedly restores information needed by subsequent reasoning stages, whereas disrupted retrieval can cause

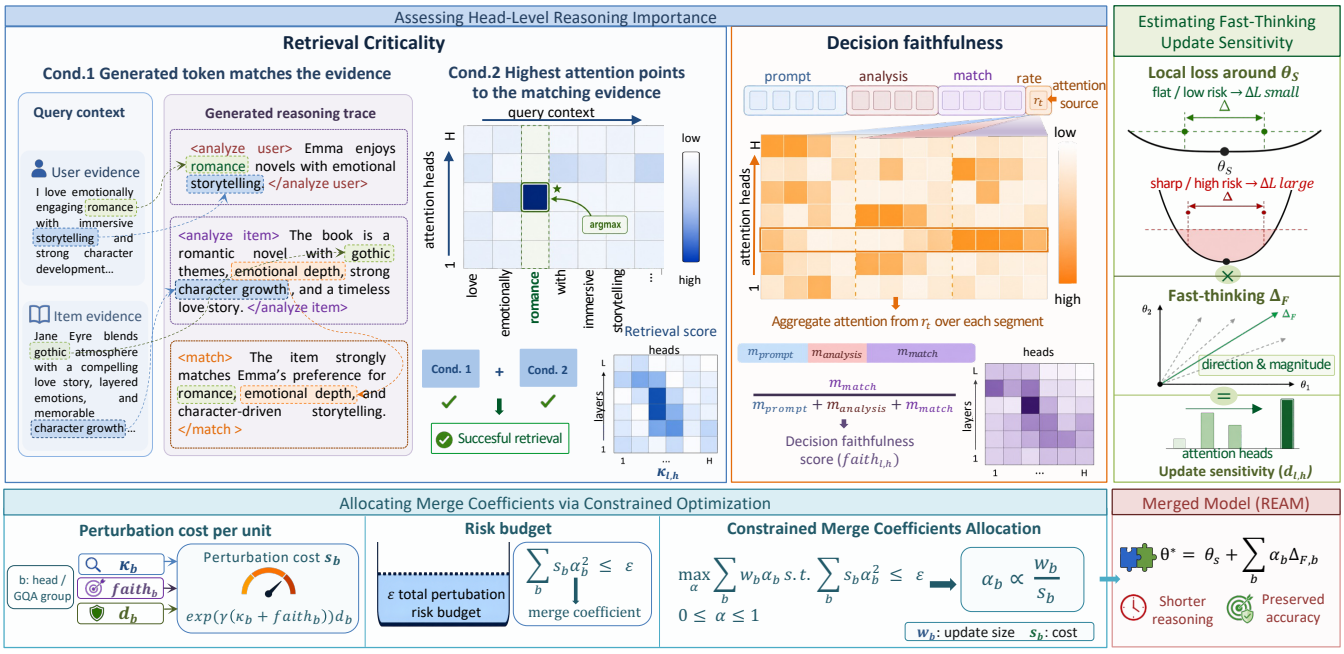


Figure 2: Overview of the proposed framework.

later generation to drift from the user–item context and propagate errors to the final prediction.

Concretely, we follow the retrieval-head criterion of Wu et al. [44], which identifies retrieval by aligning the generated token with the source position receiving a head's strongest attention. Since slow-thinking recommendation traces comprise distinct segments, we evaluate this criterion separately within each one. For an example x, consider head h in layer ℓ, denoted (ℓ, h), at decoding step t of reasoning segment s. We count retrieval as successful only if both conditions hold: (Cond.1) the generated token w_t matches a token in the segment-relevant evidence region $K^{(s)}(x)$; and (Cond.2) head (ℓ, h) assigns its maximum attention to a position containing a matching token within that region. Here, $K^{(s)}(x)$ is defined by the role of segment s. For USER and ITEM, it contains the corresponding user or item context. For MATCH, it also includes the preceding USER and ITEM analyses, since compatibility reasoning should integrate the original user–item evidence with conclusions formed earlier.

We aggregate successful retrieval events over the slow-thinking calibration set $\mathcal{D}_{\text{slow}}$, comprising 500 correctly predicted, well-formed prompt–trace pairs from the validation split (stability analysis in Appendix A.8). To reduce spurious matches, stopword and punctuation steps are excluded before evaluating Cond.1–2, and successful retrievals are TF–IDF-weighted to emphasise rare tokens.

$$R_{\ell,h} = \frac{\sum_{x\in\mathcal{D}_{\text{slow}}} \sum_{s\in\mathcal{S}} \sum_t \tau_x(w_t)\cdot g^{(s)}_{\ell,h,t}(x)}{\sum_{x\in\mathcal{D}_{\text{slow}}} \sum_{s\in\mathcal{S}} |K^{(s)}(x)| + \delta}, \tag{1}$$

where $\mathcal{S} = \{\text{USER}, \text{ITEM}, \text{MATCH}\}$, and t ranges over content-token steps in segment s; $g^{(s)}_{\ell,h,t}(x) \in \{0,1\}$ indicates whether head (ℓ, h) satisfies Cond.1–2 at step t; $\tau_x(w_t)$ is the token's TF–IDF weight (Appendix B.1); $|K^{(s)}(x)|$ is the evidence region's content-token count for segment s; and δ prevents division by zero. Thus, $R_{\ell,h}$ measures informativeness-weighted retrieval success per evidence-region content token, aggregated over $\mathcal{D}_{\text{slow}}$.

However, $R_{\ell,h}$ is unbounded and strongly right-skewed, with a small subset of heads accounting for most retrievals. We map it to a bounded, comparable weight via a log–z-score–sigmoid transform, $\kappa_{\ell,h} = \sigma\big((\log(R_{\ell,h}+\delta) - \mu_R)/s_R\big)$, where $\sigma(\cdot)$ is the logistic sigmoid and μ_R, s_R are the mean and standard deviation of $\log(R_{\ell,h}+\delta)$ across heads. Intuitively, larger $\kappa_{\ell,h}$ indicates greater retrieval criticality.

4.1.2 Decision Faithfulness. The performance of slow-thinking recommenders depends not only on preserving relevant information throughout the trace, but also on whether the final prediction faithfully draws on the preceding reasoning. This raises a complementary question: which heads make the key conclusions visible to the RATE step, guiding the model toward the correct rating rather than attending broadly across the context? We quantify this role by measuring how strongly each head connects the final rating to the part of the preceding reasoning that most directly supports it.

Since the reasoning trace comprises segments, this measure requires identifying which segment the rating relies on most. We evaluate this on $\mathcal{D}_{\text{slow}}$ by replacing USER, ITEM, and MATCH with length-matched filler text and measuring the Jensen–Shannon divergence between original and perturbed rate-token distributions. Across three datasets, replacing MATCH yields a divergence 3.3–5.1× larger than replacing USER or ITEM, indicating that the rating is most sensitive to perturbations of the MATCH segment (Appendix B.2). Accordingly, we measure decision faithfulness as the share of RATE's attention directed to MATCH, relative to the other regions.

To compute this signal, we partition the sequence preceding RATE into three regions: PROMPT, containing the input context; ANALYZE, combining USER and ITEM, both of which affect the rating distribution substantially less than MATCH (Appendix B.2); and MATCH. For each example $x \in \mathcal{D}_{\text{slow}}$, let $T_{\text{RATE}}(x)$ denote positions

of rating-content tokens in the RATE segment.

$$m^r_{\ell,h} = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_{x \in \mathcal{D}_{\text{slow}}} \frac{1}{|T_{\text{RATE}}(x)|} \sum_{t \in T_{\text{RATE}}(x)} \sum_{\tau \in \Omega_r(x,t)} \text{Att}^{(x)}_{\ell,h}[t,\tau], \tag{2}$$

where $\Omega_r(x,t)$ denotes positions in region r before t, and $\text{Att}^{(x)}_{\ell,h}[t,\tau]$ is the attention weight from t to τ. We use total attention mass, not per-token, to capture each region's contribution to the decision-time representation; Appendix B.3.1 tests length robustness.

We define decision faithfulness as the share of attention directed to MATCH among the three tracked regions:

$$\text{faith}_{\ell,h} = \frac{m^{\text{MATCH}}_{\ell,h}}{m^{\text{MATCH}}_{\ell,h} + m^{\text{ANALYZE}}_{\ell,h} + m^{\text{PROMPT}}_{\ell,h} + \delta}. \tag{3}$$

The score satisfies $\text{faith}_{\ell,h} \in [0,1)$, where larger values indicate head (ℓ,h) directs a greater share of decision-time attention to the compatibility judgement in MATCH than to the prompt or analyses.

It should be noted that both $\kappa_{\ell,h}$ and $\text{faith}_{\ell,h}$ are defined at the query-head level. Under grouped-query attention (GQA), several query heads share key and value projections. Let $\mathcal{H}_{\ell,g}$ denote the set of query heads belonging to GQA group g in layer ℓ. We assign the shared projections the maximum score among their member heads:

$$\kappa^{K/V}_{\ell,g} = \max_{h \in \mathcal{H}_{\ell,g}} \kappa_{\ell,h}, \qquad \text{faith}^{K/V}_{\ell,g} = \max_{h \in \mathcal{H}_{\ell,g}} \text{faith}_{\ell,h}.$$

This conservative aggregation treats the shared K/V parameters as reasoning-critical whenever any associated query head is critical.

4.2 Estimating Fast-Thinking Update Sensitivity

The preceding analysis characterises each head's importance to the reasoning underlying the final prediction, but not how safely its parameters can be modified. Around the trained slow-thinking solution, the loss may be flat along some parameter directions, tolerating larger changes with little effect, but sharp along others, where even small perturbations can substantially alter the model's behaviour. Since the fast-thinking update modifies each head along a specific direction and magnitude, its risk depends on the loss sensitivity along that update. Following Fisher-weighted merging [30], we estimate the sensitivity of the slow-thinking model using the diagonal empirical Fisher of θ_S. We then weight this sensitivity by the squared fast-thinking update to obtain a component-level risk measure, which we term *head-level update sensitivity* (see Appendix B.4 for the derivation).

For each calibration example $x \in \mathcal{D}_{\text{slow}}$, let $Y(x) = (y_1, \ldots, y_{|Y(x)|})$ denote its complete generated response. Because a head may influence token predictions throughout the response, we compute its sensitivity using the average teacher-forced loss over $Y(x)$:

$$\ell_{\theta_S}(x) = -\frac{1}{|Y(x)|} \sum_{t=1}^{|Y(x)|} \log p_{\theta_S}(y_t \mid y_{<t}, \text{prompt}(x)), \tag{4}$$

where $\text{prompt}(x)$ denotes the prompt portion of x. Averaging over $|Y(x)|$ prevents longer responses from dominating the estimate. For each parameter $\theta^{(j)}_S$, we compute the diagonal empirical Fisher as

$$F_j = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_{x \in \mathcal{D}_{\text{slow}}} \left(\frac{\partial \ell_{\theta_S}(x)}{\partial \theta^{(j)}_S} \right)^2. \tag{5}$$

To obtain head- and group-level sensitivities, we partition Fisher entries by the structure of the attention projection matrices W_Q, W_K, W_V, and W_O (query, key, value, output). W_Q and W_O are divided by query head, while W_K and W_V are divided by shared GQA group. Under our convention, each head in W_O corresponds to the columns of its output in the concatenated multi-head representation.

Let c index a single-projection slice: a Q or O slice for query head (ℓ,h), or a K or V slice for GQA group (ℓ,g). We define its update sensitivity as the Fisher-weighted squared fast-thinking update:

$$d_c = \sum_{j \in c} F_j \left(\Delta_{F,j}\right)^2. \tag{6}$$

A larger d_c indicates loss sensitivity to that portion of the fast-thinking update, calling for more conservative merging (derivation in Appendix B.5). To match merge-coefficient granularity, we average slice sensitivities: $d^{QO}_{\ell,h} = (d^Q_{\ell,h} + d^O_{\ell,h})/2$ for query head (ℓ,h), and $d^{KV}_{\ell,g} = (d^K_{\ell,g} + d^V_{\ell,g})/2$ for GQA group (ℓ,g). Hereafter, b indexes either a Q/O head or K/V group, and d_b denotes its sensitivity.

4.3 Allocating Merge Coefficients via Constrained Optimization

Together, reasoning importance and update sensitivity provide the reasoning-grounded basis for selective merging by indicating which heads should be protected and how cautiously each can be modified. We translate these signals into head-aware merge coefficients through a constrained optimisation that transfers as much of the fast-thinking update as possible while limiting the overall risk to recommendation-relevant reasoning.

Concretely, we combine retrieval criticality, decision faithfulness, and update sensitivity into a perturbation weight that represents the risk of perturbing unit b along the fast-thinking update:

$$s_b = \exp(\gamma(\kappa_b + \text{faith}_b))\, d_b,$$

where b indexes either the paired Q/O parameters of a query head or the shared K/V parameters of a GQA group. The exponential factor amplifies update sensitivity of reasoning-critical units, with γ controlling strength (ablations: Appendix B.6; normalisation: B.7). The allocation is robust to uniform-attention bias in faith_b, since any constant offset rescales every s_b equally (Appendix B.3.2).

Given these perturbation weights, we choose merge coefficients that maximise the total applied fast-thinking update while limiting aggregate reasoning-grounded risk. We solve this separately for the Q/O and K/V unit sets, $\mathcal{B}^{QO}$ and $\mathcal{B}^{KV}$, because they operate at different granularities under GQA and exhibit different update and risk scales. For either set $\mathcal{B}$,

$$\max_{\{\alpha_b\}_{b \in \mathcal{B}}} \sum_{b \in \mathcal{B}} w_b \alpha_b \quad \text{s.t.} \sum_{b \in \mathcal{B}} s_b \alpha_b^2 \le \varepsilon, \qquad 0 \le \alpha_b \le \bar{\alpha}, \tag{7}$$

where $w_b = \|\Delta_{F,b}\|_F$ denotes the magnitude of the fast-thinking update for unit b, and $\bar{\alpha}$ caps each merge coefficient, mainly to prevent unstable extrapolation rather than as a sensitive hyperparameter (value in Appendix A.3); ε controls the aggregate perturbation. We set $\varepsilon = \rho \bar{\alpha}^2 \sum_{b \in \mathcal{B}} s_b$, where $\rho \in (0,1]$ specifies the permitted fraction of the perturbation incurred when $\alpha_b = \bar{\alpha}$ for all b.

Since the objective is linear and the constraint is convex quadratic for $s_b \ge 0$, KKT conditions characterise the global optimum

Table 1: Main results across three datasets. Lower is better (↓). Among merging methods, bold and underlined values denote the best and second-best results, respectively. MAE and Tok report 95% t-based confidence intervals (±), while RMSE reports 95% percentile-bootstrap confidence intervals ([ ]).

	Amazon Book			Yelp			Amazon Music		
Method	MAE↓	RMSE↓	Tok↓	MAE↓	RMSE↓	Tok↓	MAE↓	RMSE↓	Tok↓
Qwen2.5-3B-Instruct	0.7504±0.0160	0.9521 [.930,.974]	342.68±1.40	0.9160±0.0230	1.1724 [1.142,1.203]	322.71±1.70	0.6875±0.0290	0.8662 [.824,.908]	334.69±2.50
RecZero (θ_S)	0.6650±0.0180	0.9331 [.906,.959]	313.58±1.25	0.7769±0.0250	1.1054 [1.069,1.141]	314.54±1.30	0.5433±0.0360	0.8510 [.796,.905]	344.28±2.25
TALLRec (θ_F)[1]	0.6994±0.0255	1.1553 [1.122,1.187]	4.0	0.8082±0.0340	1.3509 [1.307,1.394]	4.0	0.7871±0.0310	0.9669 [.933,1.002]	4.0
Arithmetic and pruning-based merging									
Average Merging	0.6880±0.0170	0.9202 [.898,.941]	270.05±1.15	0.7907±0.0235	1.0866 [1.053,1.120]	271.96±1.45	0.5943±0.0325	0.8344 [.786,.882]	280.48±2.20
Task Arithmetic	0.6742±0.0180	0.9333 [.911,.956]	253.13±1.58	0.7723±0.0240	1.0872 [1.054,1.121]	**256.11±2.40**	0.5605±0.0335	0.8265 [.779,.873]	269.44±2.55
DARE	0.7030±0.0170	0.9320 [.910,.954]	272.32±1.20	0.8025±0.0235	1.0968 [1.063,1.130]	275.27±2.45	0.6122±0.0315	0.8409 [.794,.888]	277.55±2.75
DARE + TA	0.6835±0.0175	0.9343 [.912,.956]	252.81±1.53	0.7809±0.0240	1.0921 [1.058,1.125]	265.79±4.30	0.5869±0.0340	0.8486 [.799,.898]	**266.60±2.70**
DARE + TIES[2]	0.7161±0.0210	0.9992 [.976,1.023]	261.39±2.20	0.7591±0.0270	1.0855 [1.051,1.121]	266.21±2.20	0.5947±0.0355	0.8651 [.822,.909]	257.75±3.25
Data-driven merging									
AIM + TA	0.6805±0.0175	0.9272 [.905,.949]	260.46±1.15	0.7735±0.0235	1.0788 [1.046,1.112]	261.62±1.35	0.5678±0.0330	**0.8227 [.775,.871]**	280.71±2.20
ACM + TA	0.6932±0.0165	0.9162 [.894,.938]	282.74±1.15	0.7987±0.0240	1.0995 [1.066,1.132]	290.03±1.65	0.5871±0.0325	0.8333 [.786,.881]	304.72±2.35
RAIN-Merging	0.6632±0.0180	0.9325 [.906,.959]	290.30±1.20	0.7731±0.0250	1.1065 [1.070,1.143]	289.17±1.45	0.5424±0.0365	0.8527 [.799,.905]	315.63±2.25
REAM (ours)	**0.6338±0.0181**	**0.9116 [.888,.936]**	**237.33±1.24**	**0.7564±0.0237**	**1.0649 [1.033,1.098]**	258.19±1.25	**0.5348±0.0351**	0.8324 [.783,.882]	271.49±2.49
Reduction vs. θ_S	*4.7%*	*2.3%*	*24.3%*	*2.6%*	*3.7%*	*17.9%*	*1.6%*	*2.2%*	*21.1%*

(Appendix B.8). When binding, they yield the water-filling solution

$$\alpha_b^\star = \mathrm{clip}\left(\frac{w_b}{2\mu s_b}, 0, \bar{\alpha}\right),$$

where $\mu > 0$ is determined separately for the Q/O and K/V allocations by bisection. For units with $s_b = 0$, we set $\alpha_b^\star = \bar{\alpha}$. This yields coefficients that adapt jointly to the fast-thinking update and its reasoning-aware perturbation weight.

Finally, we construct the merged model by applying the resulting coefficients to the fast-thinking task vector:

$$\theta^\star = \theta_S + \sum_b \alpha_b^\star \Delta_{F,b}, \tag{8}$$

where b spans all merged attention and FFN parameter groups; all remaining parameters are kept at their values in θ_S. Attention coefficients follow from the allocation above. As FFNs lack head structure, we assign each layer the mean Q/O coefficient, $\alpha_\ell^{\star,\mathrm{FFN}} = H^{-1}\sum_{h=1}^{H}\alpha_{\ell,h}^{\star,QO}$, where H is the number of query heads per layer. Because FFNs transform the full hidden representation of each token rather than a head-specific subspace, their updates may affect the reasoning-to-rating mapping more broadly, motivating more conservative merging. In particular, validation shows that merging the final several FFN layers degrades the accuracy–efficiency trade-off, indicating that these layers are especially sensitive to changes in the reasoning-to-rating mapping. We therefore exclude layers 30–35 from FFN merging by setting their coefficients to zero in our experiments; results are reported in §5.4.3.

5 Experiments

We organise our experiments around the following research questions (RQs), addressed in §5.2 to §5.4, respectively:

- **RQ1:** Can general-purpose model merging shorten slow-thinking reasoning traces while preserving recommendation accuracy?
- **RQ2:** Does REAM achieve a better accuracy–efficiency trade-off than existing training-free merging baselines, including both data-driven and arithmetic-based methods?
- **RQ3:** How do REAM's individual design choices contribute to the resulting accuracy–efficiency trade-off?

5.1 Experimental Setup

Models and datasets. We evaluate on Amazon Book, Amazon Music, and Yelp using the Reason4Rec rating-prediction splits [7]. Each instance provides a user's historical ratings and reviews and a target item, with a 1–5 rating as the label. The slow-thinking model θ_S is RecZero [22], fine-tuned from Qwen2.5-3B-Instruct for structured reasoning. The fast-thinking model θ_F is an adapted TALLRec model [4], fine-tuned from the base Qwen2.5-3B-non-instruct to obtain a behaviourally distinct direct-prediction model, as an instruct-initialised model may retain substantial reasoning and instruction-following behaviour even when fine-tuned for direct rating prediction. (Appendix A.1). We use Qwen2.5-3B-Instruct as the shared base θ_B for both task vectors.

Baselines. We compare REAM with the two source models and eight representative training-free baselines spanning arithmetic-based, pruning-based, and data-driven methods (Appendices A.5 and A.6). Among the data-driven baselines, AIM and ACM derive one coefficient per layer from activation statistics, whereas RAIN-Merging operates at head granularity but bases its coefficients on instruction relevance rather than recommendation reasoning—the distinction REAM's allocation is built around. All baselines merge the same θ_S and θ_F using their recommended configurations.

Metrics. We report MAE and RMSE for rating accuracy and mean generated tokens per response (Tok) as a proxy for reasoning verbosity; lower is better for all metrics. Metrics are computed on parseable outputs (Appendix A.3); parse rates are near 100% for all methods except DARE+TIES, as noted in Table 1.

[1] TALLRec outputs a short numerical rating response; token count is not schema-comparable to other methods.
[2] DARE+TIES metrics are computed over successfully parsed outputs only and are not directly comparable.

Implementation details. REAM and all baselines share a common configuration (calibration, hyperparameters, hardware; Appendix A.3), with baseline-specific settings in Appendix A.6 and REAM's computational overhead reported in Appendix A.4.

5.2 Limits of General-Purpose Model Merging

General-purpose model merging provides a training-free means of transferring concise generation from a fast-thinking recommender to a slow-thinking one, but the accuracy–efficiency trade-off remains unclear. We therefore evaluate arithmetic- and pruning-based methods alongside data-driven merging approaches to determine how substantially they shorten reasoning traces and whether these efficiency gains consistently preserve recommendation accuracy.

(1) General-purpose merging shortens reasoning, but does not reliably preserve accuracy. On Book, Task Arithmetic and DARE+TA achieve substantial compression, reducing generation length to 253.13 and 252.81 tokens, respectively, but both worsen MAE relative to θ_S (0.6742 and 0.6835 vs. 0.6650). Average Merging and DARE exhibit the same trade-off, although with more modest compression (Table 1). Importantly, this pattern is not confined to arithmetic- or pruning-based methods: AIM+TA and ACM+TA also worsen MAE to 0.6805 and 0.6932, with ACM+TA underperforming most arithmetic baselines despite incorporating activation statistics. These results show that general-purpose merging can transfer concise generation, but does not consistently preserve the prediction behaviour of the slow-thinking model.

(2) General-purpose merging can alter the distinctive rating behaviour of the slow-thinking model. On Music, most general-purpose merging methods improve RMSE relative to θ_S while worsening MAE. Task Arithmetic, DARE, and DARE+TA achieve RMSE 0.8265–0.8486, compared with 0.8510 for θ_S, but increase MAE from 0.5433 to 0.5605–0.6122. AIM+TA and ACM+TA exhibit the same divergence, with AIM+TA attaining the lowest RMSE among baselines (0.8227) while worsening MAE to 0.5678 (Table 1). Because RMSE penalises large errors more heavily than MAE, a lower RMSE alongside a higher MAE is consistent with a redistribution of errors—fewer extreme deviations but larger absolute errors on average—rather than an overall improvement in accuracy [40]. This redistribution is consequential for slow-thinking recommenders, whose predictive advantage depends on distinguishing fine-grained user–item compatibility and grounding each rating in that assessment [7, 22]. Such a shift does not necessarily preserve this instance-specific judgement, suggesting that the merged model may alter the rating behaviour of θ_S across individual cases.

5.3 Accuracy–Efficiency Trade-off of REAM

Having established the need for more selective merging, we next evaluate whether REAM's reasoning-grounded, head-level allocation achieves a better accuracy–efficiency trade-off than existing model merging methods. Table 1 compares REAM with the two source models and eight baselines across Book, Yelp, and Music.

(1) REAM delivers the most consistent balance between accuracy and efficiency. REAM is the only comparable merging method that maintains accuracy across MAE and RMSE while shortening reasoning on all three datasets. It achieves the best MAE and RMSE on Book and Yelp, the shortest trace on Book, and the best

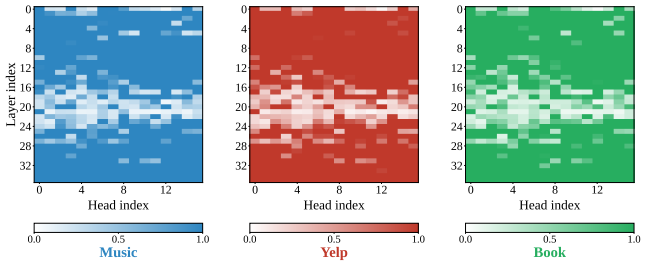


Figure 3: Per-head merge coefficients $\alpha^\star$ across layers and attention heads on Music, Yelp, and Book.

MAE on Music. Although several baselines attain stronger results on isolated metrics, none sustains these gains across both accuracy measures and reasoning length. Specifically, AIM+TA achieves a lower RMSE on Music (0.8227 vs. 0.8324), but worsens MAE relative to REAM (0.5678 vs. 0.5348) and performs less favourably on the other datasets. Taken together, these results establish REAM as the most consistent method across all three recommendation domains.

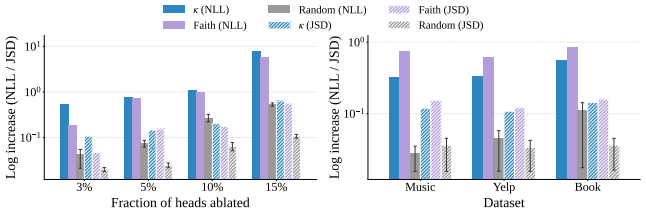
(2) REAM's advantage reflects both reasoning-aware signals and head-level allocation. For reasoning compression in recommender systems, effective coefficient design requires a signal grounded in recommendation reasoning and sufficient granularity to act on it. Concretely, AIM+TA and ACM+TA derive coefficients from activation statistics but assign a single value per layer, thereby applying the same merge strength to heads with substantially different reasoning roles. This coarse allocation yields a consistently weaker accuracy–efficiency trade-off: across all three datasets, AIM+TA and ACM+TA produce longer traces and higher MAE than REAM (e.g., 260.46 and 282.74 vs. 237.33 tokens, and 0.6805 and 0.6932 vs. 0.6338 MAE on Book). The observed head-level heterogeneity helps explain this gap: on Book, the top 5% of retrieval-critical heads hold 78.0% of retrieval mass (Gini= 0.922), whereas decision faithfulness is less concentrated (Gini= 0.553), indicating that reasoning importance is distributed unevenly across heads (Appendix A.9.1). Accordingly, Figure 3 reveals substantial coefficient variation within individual layers. RAIN-Merging operates at head granularity, but derives its coefficients from instruction relevance rather than recommendation reasoning and underperforms REAM on every metric across all three datasets (Table 1). Together, these comparisons show that head-level control is most effective when paired with a reasoning-grounded allocation criterion.

5.4 Impact of Key Components

5.4.1 Component Ablation. We ablate REAM's three allocation signals—retrieval criticality κ, decision faithfulness, and Fisher update sensitivity d—on the full Music and Yelp test sets (Table 2). Removing Fisher sensitivity has the largest effect on both datasets, increasing MAE to 0.5533 on Music (vs. 0.5348) and 0.7616 on Yelp (vs. 0.7564). Removing faithfulness has a smaller effect on Music MAE (0.5350) and slightly improves MAE on Yelp (0.7544), but on both datasets increases RMSE and reasoning length. Together with the near-zero correlation between κ and faithfulness (Spearman $|\rho| \leq 0.018$ in every domain; Appendix A.9.2), these results support complementary roles: Fisher sensitivity captures vulnerability to the update, while κ and faithfulness identify distinct retrieval- and

Table 2: Leave-one-out ablation on Amazon Music and Yelp. ↓ indicates lower is better. Bold marks best value per column.

Config.	Amazon Music			Yelp		
	MAE	RMSE	Tok	MAE	RMSE	Tok
Full	**0.5348**	0.8324	**271.49**	0.7564	**1.0649**	**258.19**
w/o κ	0.5499	0.8320	271.53	0.7599	1.0807	259.55
w/o faith	0.5350	0.8350	271.98	**0.7544**	1.0716	263.72
w/o d	0.5533	**0.8252**	271.89	0.7616	1.0854	262.62



(a) Music, varying fraction ablated. (b) Cross-domain, 5% ablated.

Figure 4: NLL and JSD increase (log scale) from ablating retrieval-critical (κ) vs. random heads.

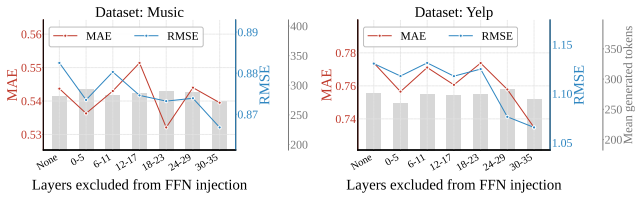


Figure 5: FFN exclusion sweep on Music and Yelp subsets. Bars show Tok; lines show MAE and RMSE.

decision-related risks whose contributions are not fully reflected by any single metric.

5.4.2 Causal Validation of Head-Level Signals. We test whether the heads identified by κ and decision faithfulness are causally important by mean-ablating the top 5% under each signal and comparing held-out NLL and JSD with size-matched random-head ablations averaged over three draws. For NLL, the ratio compares increases from the unablated baseline; for JSD, it directly compares the divergence under critical- and random-head ablation (Appendix A.9.3). Relative to random ablation, ablating κ-critical heads increases NLL/JSD by 11.57×/3.29× on Music, 4.93×/3.96× on Book, and 7.35×/3.19× on Yelp (Figure 4). Faithfulness-critical heads cause similarly large increases of 9.65×/5.64×, 11.16×/5.20×, and 6.63×/4.19×, respectively. Despite these comparable causal effects, the two signals' top-5% head sets are nearly disjoint (Jaccard ≤ 1.75%; Table 8), while each remains stable across datasets (mean cross-domain Jaccard 87.1% for κ and 79.4% for faithfulness; Appendix A.9.1). Together, these results show that the two signals identify complementary and transferable head-level roles rather than redundant rankings.

5.4.3 FFN Exclusion. Because FFNs lack head structure, their merge coefficients are assigned at layer granularity. We assess whether the fast-thinking update can be safely applied across depth by sweeping six-layer exclusion ranges on the Music and Yelp validation subsets (Figure 5). Excluding the final six layers, $\mathcal{L}_{\text{excl}} = \{30, \ldots, 35\}$,

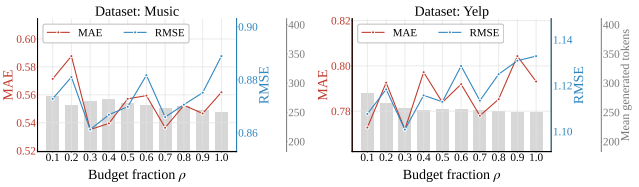


Figure 6: Hyperparameter analysis: MAE, RMSE, and Tok vs. budget fraction ρ, on Music and Yelp subsets.

Table 3: Generalisability of REAM across model scales, base model families, and reasoning backbones (Yelp). ↓ indicates lower is better. CI notation follows Table 1.

Method	MAE↓	RMSE↓	Tok↓
Qwen2.5-7B-Instruct			
RecZero (θ_S)	**0.7626**±0.024	**1.0757** [1.040,1.112]	336.02±1.40
Task Arithmetic	0.8134±0.025	1.1327 [1.098,1.167]	302.37±1.95
REAM (ours)	0.7910±0.024	1.0864 [1.053,1.120]	**288.46**±1.35
Llama-3.2-3B-Instruct			
RecZero (θ_S)	0.8059±0.024	1.0977 [1.069,1.127]	323.79±1.85
Task Arithmetic	**0.7269**±0.030	1.1395 [1.101,1.178]	**254.18**±1.05
REAM (ours)	0.7769±0.026	**1.0895** [1.055,1.126]	257.18±2.20
RecOne (Qwen2.5-3B-Instruct)			
RecOne (θ_S)	0.7395±0.027	1.1310 [1.094,1.168]	365.69±0.65
Task Arithmetic	0.6885±0.028	1.1200 [1.083,1.158]	321.68±1.15
REAM (ours)	**0.6876**±0.028	**1.1137** [1.077,1.151]	**309.70**±1.10

gives the most consistent accuracy–efficiency trade-off: best RMSE on both datasets, best MAE on Yelp, and shortest trace on Music. This suggests late FFNs are especially sensitive to the fast-thinking update, consistent with FFNs' direct role in shaping model outputs [10], where perturbation near the output may disrupt the reasoning-to-rating mapping. We therefore use layers 30–35 as the fixed FFN exclusion window for all Qwen2.5-3B evaluations (§4.3). The broader late-layer exclusion strategy generalises across model scales and families (Appendix B.9).

5.4.4 Hyperparameter Sensitivity. We sweep the budget fraction ρ, which controls how much of the fast-thinking update the allocation can absorb, over $\rho \in [0.1, 1.0]$ on the Music and Yelp validation subsets (Figure 6). Performance is non-monotonic, confirming that ρ is a true allocation budget: too little update limits trace shortening, while too much can perturb reasoning-sensitive components. Our fixed choice, $\rho = 0.3$, consistently lies near the best accuracy–efficiency trade-off on both datasets.

5.4.5 Generalisability. REAM's transferability across model scale, base-model family, and training paradigm is evaluated using the same hyperparameters as Appendix A.3, without re-tuning (Table 3). On Qwen2.5-7B, REAM shortens θ_S's trace from 336.02 to 288.46 tokens and outperforms Task Arithmetic in MAE and RMSE, although it remains less accurate than θ_S. On Llama-3.2-3B, REAM improves on θ_S across all three metrics, and holds a lower RMSE than Task Arithmetic despite Task Arithmetic's shorter trace and lower MAE—consistent with the same MAE–RMSE divergence observed in §5.2, where aggressive compression can come at the cost of rating stability. On RecOne [22], which learns the same typed-segment format through SFT followed by RL rather than RecZero's pure-RL training, REAM outperforms both θ_S and Task Arithmetic

on all three metrics. These results support REAM's transferability across model scales, families, and training paradigms, while leaving room for improvement in some settings.

6 Conclusion

This work shows that reasoning compression in slow-thinking recommender systems depends not only on how much of the fast-thinking update is merged, but also on which components receive it. To our knowledge, REAM is the first merging framework designed for this setting, assigning each head a merge coefficient according to its retrieval criticality, decision faithfulness, and sensitivity to parameter change. The two reasoning signals identify causally important yet largely disjoint head sets, supporting their complementary rather than redundant contributions. Across three recommendation datasets, REAM shortens reasoning while maintaining accuracy, achieving a more favourable accuracy–efficiency trade-off than the evaluated training-free merging baselines. The remaining accuracy gap at larger model scale highlights an important direction for extending selective merging to larger recommenders.

References

[1] Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. arXiv:2503.04697 [cs.CL] https://arxiv.org/abs/2503.04697

[2] Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-Thought: Efficient LLM Reasoning with Adaptive Cognitive-Inspired Sketching. arXiv:2503.05179 [cs.CL] https://arxiv.org/abs/2503.05179

[3] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Chong Chen, Fuli Feng, and Qi Tian. 2023. A Bi-Step Grounding Paradigm for Large Language Models in Recommendation Systems. arXiv:2308.08434 [cs.IR] https://arxiv.org/abs/2308.08434

[4] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, 1007–1014. doi:10.1145/3604915.3608857

[5] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc.

[6] Jundong Chen, Honglei Zhang, Chunxu Zhang, Fangyuan Luo, and Yidong Li. 2025. Breaking the Aggregation Bottleneck in Federated Recommendation: A Personalized Model Merging Approach. arXiv:2508.12386 [cs.DC] https://arxiv.org/abs/2508.12386

[7] Yi Fang, Wenjie Wang, Yang Zhang, Fengbin Zhu, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Reason4Rec: Large Language Models for Recommendation with Deliberative User Preference Alignment. arXiv:2502.02061 [cs.IR] https://arxiv.org/abs/2502.02061

[8] Jingtong Gao, Zeyu Song, Chi Lu, Xiaopeng Li, Derong Xu, Maolin Wang, Peng Jiang, Kun Gai, Qingpeng Cai, and Xiangyu Zhao. 2026. Reinforced Preference Optimization for Reasoning-Augmented Recommendations. arXiv:2605.21967 [cs.IR] https://arxiv.org/abs/2605.21967

[9] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2023. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). arXiv:2203.13366 [cs.IR] https://arxiv.org/abs/2203.13366

[10] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. arXiv:2012.14913 [cs.CL] https://arxiv.org/abs/2012.14913

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* 645, 8081 (2025), 633–638. doi:10.1038/s41586-025-09422-z

[12] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-Budget-Aware LLM Reasoning. arXiv:2412.18547 [cs.CL] https://arxiv.org/abs/2412.18547

[13] Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. ThinkPrune: Pruning Long Chain-of-Thought of LLMs via Reinforcement Learning. arXiv:2504.01296 [cs.CL] https://arxiv.org/abs/2504.01296

[14] Min Hou, Xin Liu, Le Wu, Chenyi He, Hao Liu, Zhi Li, Xin Li, and Si Wei. 2025. WeaveRec: An LLM-Based Cross-Domain Sequential Recommendation Framework with Model Merging. arXiv:2510.26546 [cs.IR] https://arxiv.org/abs/2510.26546

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[16] Zhehao Huang, Yuhang Liu, Baijiong Lin, Yixin Lou, Zhengbao He, Hanling Tian, Tao Li, and Xiaolin Huang. 2026. RAIN-Merging: A Gradient-Free Method to Enhance Instruction Following in Large Reasoning Models with Preserved Thinking Format. arXiv:2602.22538 [cs.LG] https://arxiv.org/abs/2602.22538

[17] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing Models with Task Arithmetic. arXiv:2212.04089 [cs.LG] https://arxiv.org/abs/2212.04089

[18] Jie Jiang, Yang Wu, Qian Li, Yuling Xiong, Hongbo Tang, Xun Liu, Haoze Wang, Jun Zhang, Huan Yu, and Hailong Shi. 2026. SCoTER: Structured Chain-of-Thought Transfer for Enhanced Recommendation. arXiv:2511.19514 [cs.IR] https://arxiv.org/abs/2511.19514

[19] Daniel Kahneman. 2011. *Thinking, Fast and Slow.* Farrar, Straus and Giroux, New York.

[20] Hyunsoo Kim, Jaewan Moon, Seongmin Park, and Jongwuk Lee. 2026. MergeRec: Model Merging for Data-Isolated Cross-Domain Sequential Recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1* (Republic of Korea) *(KDD '26)*. Association for Computing Machinery, New York, NY, USA, 554–565. doi:10.1145/3770854.3780264

[21] Jieyong Kim, Hyunseo Kim, Hyunjin Cho, SeongKu Kang, Buru Chang, Jinyoung Yeo, and Dongha Lee. 2025. Review-driven Personalized Preference Reasoning with Large Language Models for Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*. ACM, 1697–1706. doi:10.1145/3726302.3730055

[22] Xiaoyu Kong, Junguang Jiang, Bin Liu, Ziru Xu, Han Zhu, Jian Xu, Bo Zheng, Jiancan Wu, and Xiang Wang. 2025. Think before Recommendation: Autonomous Reasoning-enhanced Recommender. arXiv:2510.23077 [cs.IR] https://arxiv.org/abs/2510.23077

[23] Frederik Kunstner, Lukas Balles, and Philipp Hennig. 2020. Limitations of the Empirical Fisher Approximation for Natural Gradient Descent. arXiv:1905.12558 [cs.LG] https://arxiv.org/abs/1905.12558

[24] Shijun Li, Yu Wang, Jin Wang, Ying Li, Joydeep Ghosh, and Anne Cocos. 2026. LLM Reasoning for Cold-Start Item Recommendation. In *Proceedings of the ACM Web Conference 2026* (United Arab Emirates) *(WWW '26)*. Association for Computing Machinery, New York, NY, USA, 8409–8412. doi:10.1145/3774904.3792872

[25] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. LLaRA: Large Language-Recommendation Assistant. arXiv:2312.02445 [cs.IR] https://arxiv.org/abs/2312.02445

[26] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2024. How Can Recommender Systems Benefit from Large Language Models: A Survey. arXiv:2306.05817 [cs.IR] https://arxiv.org/abs/2306.05817

[27] Shuqi Liu, Han Wu, Bowei He, Xiongwei Han, Mingxuan Yuan, and Linqi Song. 2025. Sens-Merging: Sensitivity-Guided Parameter Balancing for Merging Large Language Models. arXiv:2502.12420 [cs.CL] https://arxiv.org/abs/2502.12420

[28] Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. CoT-Valve: Length-Compressible Chain-of-Thought Tuning. arXiv:2502.09601 [cs.AI] https://arxiv.org/abs/2502.09601

[29] James Martens. 2020. New insights and perspectives on the natural gradient method. arXiv:1412.1193 [cs.LG] https://arxiv.org/abs/1412.1193

[30] Michael Matena and Colin Raffel. 2022. Merging Models with Fisher-Weighted Averaging. arXiv:2111.09832 [cs.LG] https://arxiv.org/abs/2111.09832

[31] Amin Heyrani Nobari, Kaveh Alim, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. 2025. Activation-Informed Merging of Large Language Models. arXiv:2502.02421 [cs.CL] https://arxiv.org/abs/2502.02421

[32] Yein Park, Minbyul Jeong, and Jaewoo Kang. 2026. Thinking Sparks!: Emergent Attention Heads in Reasoning Models During Post Training. arXiv:2509.25758 [cs.AI] https://arxiv.org/abs/2509.25758

[33] Jung Hyun Ryu, Jaeheyoung Jeon, Jewoong Cho, and Myungjoo Kang 1. 2023. Fisher-Weighted Merge of Contrastive Learning Models in Sequential Recommendation. arXiv:2307.05476 [cs.IR] https://arxiv.org/abs/2307.05476

[34] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. arXiv:2503.16419 [cs.CL] https://arxiv.org/abs/2503.16419

[35] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large Language Models Enhanced Collaborative Filtering. arXiv:2403.17688 [cs.IR] https://arxiv.org/abs/2403.17688

[36] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. arXiv:1905.09418 [cs.CL] https://arxiv.org/abs/1905.09418

[37] Xinfeng Wang, Jin Cui, Yoshimi Suzuki, and Fumiyo Fukumoto. 2025. RDRec: Rationale Distillation for LLM-based Recommendation. arXiv:2405.10587 [cs.CL] https://arxiv.org/abs/2405.10587
[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL] https://arxiv.org/abs/2201.11903
[39] Tianjun Wei, Enneng Yang, Yingpeng Du, Huizhong Guo, Jie Zhang, and Zhu Sun. 2026. MMGRid: Navigating Temporal-aware and Cross-domain Generative Recommendation via Model Merging. arXiv:2601.15930 [cs.IR] https://arxiv.org/abs/2601.15930
[40] Cort J. Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research* 30 (2005), 79–82. https://api.semanticscholar.org/CorpusID:120556606
[41] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv:2203.05482 [cs.LG] https://arxiv.org/abs/2203.05482
[42] Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. 2025. Unlocking Efficient Long-to-Short LLM Reasoning with Model Merging. arXiv:2503.20641 [cs.CL] https://arxiv.org/abs/2503.20641
[43] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2024. A Survey on Large Language Models for Recommendation. arXiv:2305.19860 [cs.IR] https://arxiv.org/abs/2305.19860
[44] Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval Head Mechanistically Explains Long-Context Factuality. arXiv:2404.15574 [cs.CL] https://arxiv.org/abs/2404.15574
[45] Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable Chain-of-Thought Compression in LLMs. arXiv:2502.12067 [cs.CL] https://arxiv.org/abs/2502.12067
[46] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of Draft: Thinking Faster by Writing Less. arXiv:2502.18600 [cs.CL] https://arxiv.org/abs/2502.18600
[47] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-Merging: Resolving Interference When Merging Models. arXiv:2306.01708 [cs.LG] https://arxiv.org/abs/2306.01708
[48] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2025. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities. arXiv:2408.07666 [cs.LG] https://arxiv.org/abs/2408.07666
[49] Junyao Yang, Jianwei Wang, Huiping Zhuang, Cen Chen, and Ziqian Zeng. 2026. RCP-Merging: Merging Long Chain-of-Thought Models with Domain-Specific Models by Considering Reasoning Capability as Prior. arXiv:2508.03140 [cs.CL] https://arxiv.org/abs/2508.03140
[50] Yuxuan Yao, Shuqi Liu, Zehua Liu, Qintong Li, Mingyang Liu, Xiongwei Han, Zhijiang Guo, Han Wu, and Linqi Song. 2025. Activation-Guided Consensus Merging for Large Language Models. arXiv:2505.14009 [cs.CL] https://arxiv.org/abs/2505.14009
[51] Zixuan Yi and Iadh Ounis. 2025. A Multi-modal Recipe for Improved Multi-domain Recommendation. In *Advances in Information Retrieval (ECIR 2025)* (Lucca, Italy). Springer-Verlag, Berlin, Heidelberg, 421–436. doi:10.1007/978-3-031-88708-6_27
[52] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. arXiv:2311.03099 [cs.CL] https://arxiv.org/abs/2311.03099
[53] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. arXiv:2305.07001 [cs.IR] https://arxiv.org/abs/2305.07001
[54] Jiaqi Zhang, Junliang Yu, Zongwei Wang, Wei Yuan, Tong Chen, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. 2025. Towards Reasoning-Aware Recommender Systems: A Survey in the LLM Era. *TechRxiv* (2025). doi:10.36227/techrxiv.176287939.92578520/v2
[55] Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2025. Reinforced Latent Reasoning for LLM-based Recommendation. arXiv:2505.19092 [cs.AI] https://arxiv.org/abs/2505.19092

A Implementation Details

A.1 Fast-Thinking Model Adaptation

TALLRec [4] was originally trained with LoRA [15] for binary recommendation feedback: given a user's interaction history and a target item, the model predicts whether the user would like the item, e.g., `Output: Yes.`

We adapt TALLRec to serve as the fast-thinking model θ_F in two ways. First, we change the task from binary feedback to direct rating prediction. The adapted model outputs a single rating on a 1–5 scale without intermediate reasoning, making its output directly comparable to the final RATE value produced by θ_S. Appendix A.2 shows this adapted output alongside RecZero's structured reasoning trace on the same instance.

Second, we fully fine-tune all parameters of TALLRec rather than using LoRA, since our merging approach requires task-vector slices expressed as direct differences over full weight matrices — W_Q, W_K, W_V, W_O, and the FFN weights — rather than the low-rank adapter factors LoRA produces.

A.2 Prompt Format Examples

Both models receive the same input — a user's history of previously rated items and reviews, followed by the target item — and differ only in how they produce a rating. Prompts below are abbreviated for illustration; the actual input includes the full user history and target item description.

A.2.1 *RecZero (θ_S): Slow-Thinking Structured Reasoning.*

Input: User history with previously rated books and reviews, followed by the target item:

Target item: Shifting Fates (Urban Fantasy Shifter Romance); Rating: **?**

Output:

<analyze_user> The user prefers emotionally engaging romance, morally complex characters, strong world-building, and unconventional romantic settings. </analyze_user>

<analyze_item> The target item is an urban fantasy shifter romance with a protective female lead and survival-focused premise. </analyze_item>

<match> The target item matches the user's preference for strong romantic tension, fantasy settings, and distinctive world-building, although the multi-author format may introduce some uncertainty. </match>

<rate> 4.0 </rate>

RecZero generates a four-segment output containing a three-segment reasoning trace before producing the final rating.

A.2.2 *TALLRec (θ_F): Fast-Thinking Direct Prediction.*

Instruction: Predict rating for the last item.

Input: Same user history and target item as above.

Output: 4.0/5.0

TALLRec predicts the rating directly without generating intermediate reasoning, producing only a short numerical response.

A.3 Experimental Configuration

For all merging methods, task vectors are computed relative to the shared base model θ_B. Coefficient-based methods are evaluated using the common candidate range reported in Appendix A.6. REAM's reasoning-importance and Fisher-sensitivity statistics are estimated separately for each dataset using a calibration set of $|\mathcal{D}_{\text{slow}}| = 500$ examples. We fix $\bar{\alpha} = 1$, $\gamma = 3$, $\rho = 0.3$, and $\delta = 10^{-8}$ across all datasets. All experiments are conducted on a single NVIDIA A40 GPU.

Parseability criteria. For models with typed reasoning, an output is parseable if it contains all required segments and a valid `<rate>...</rate>` span with an extractable rating; for TALLRec, only an extractable numeric rating is required.

A.4 Computational Overhead of REAM

Table 4 reports the cost of REAM's full pipeline — retrieval-criticality and decision-faithfulness estimation, Fisher-sensitivity computation, and coefficient allocation — measured on a single NVIDIA A40 GPU, the same hardware used throughout our experiments (Appendix A.3). The full pipeline completes in approximately 41 minutes (0.69 GPU-hours) per dataset. Fisher-sensitivity estimation and the Q/O and K/V aggregation dominate this cost, accounting for 73.4% of the total, while retrieval-criticality and faithfulness estimation account for 25.2%, and the constrained-optimization allocation step is negligible at 1.4%.

This cost is substantially lower than training the two source models. RecZero (θ_S) requires 4.7 hours of RL training on 2 A40 GPUs (9.4 GPU-hours), and TALLRec (θ_F) requires 6.3 hours of supervised fine-tuning on 1 A40 GPU (6.3 GPU-hours), for a combined 15.7 GPU-hours. REAM's entire pipeline therefore costs approximately 4% of this combined training budget, and—unlike training a new slow-thinking recommender—is incurred only once per dataset, requires no gradient updates to either source model, and involves no changes to decoding.

Table 4: Computational cost of REAM's training-free pipeline vs. fine-tuning the slow- and fast-thinking source models from their respective pretrained checkpoints. GPU-hours account for the number of GPUs used per training run.

Stage	Wall-clock time	GPUs	GPU-hours
Reasoning-aware ($R_{\ell,h}$ + faith)	10.4 min	1	0.17
Update sensitivity	30.2 min	1	0.50
Allocation + merge	0.6 min	1	0.01
REAM total	**41.2 min**	1	**0.69**
RecZero (θ_S) training	4.7 h	2	9.4
TALLRec (θ_F) training	6.3 h	1	6.3
Combined training cost	—	—	**15.7**

A.5 Baseline Descriptions

Average Merging, Task Arithmetic (TA) [17], DARE [52], DARE+TA, and DARE+TIES [47] are implemented using the Long-to-Short codebase [42][1]; AIM [31], ACM [50], and RAIN-Merging [16] use their official codebases.[2]

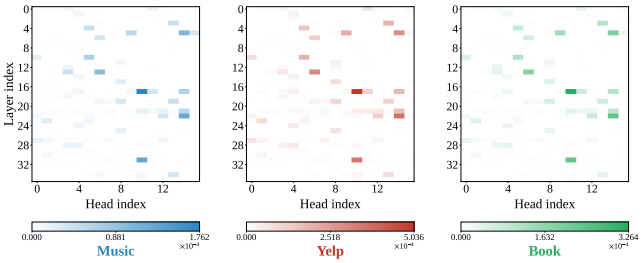


Figure 7: Retrieval criticality $R_{\ell,h}$ per head, across layers, on Music, Yelp, and Book.

Average Merging directly averages θ_S and θ_F's parameters. **TA** adds task vectors (parameter differences from a shared base) with a uniform coefficient. **DARE** randomly drops and rescales task-vector entries before merging; we evaluate it alone, combined with TA (DARE+TA), and combined with TIES (DARE+TIES). **TIES** trims small entries and resolves sign conflicts across task vectors before merging; TIES's own trimming fraction is denoted k_{TIES}. TIES alone consistently failed to produce usable output in our preliminary testing — the same sign-conflict instability likely explains DARE+TIES's lower parse rate noted in Table 1 — so we report only DARE+TIES as the representative TIES-based baseline.

AIM sets layer-wise coefficients from base-model activation magnitudes; **ACM** instead uses mutual information between base and fine-tuned activations. Both are evaluated with TA. **RAIN-Merging**, the closest head-level baseline, projects updates onto the null space of reasoning-token activations and scales head coefficients by instruction-attention alignment — grounded in instruction relevance rather than each head's role in evidence retrieval and decision grounding.

A.6 Baseline Hyperparameters

For each baseline, we use the hyperparameters recommended in its original release.

Table 5: Hyperparameters used for each baseline, taken from the original method's recommended configuration.

Method	Hyperparameters
Average Merging	—
Task Arithmetic	$\alpha = 0.7$
DARE	$p = 0.3$
DARE + TA	$p = 0.3$, $\alpha = 0.7$
DARE + TIES	$p = 0.3$, $k_{\text{TIES}} = 0.8$, $\alpha = 1.0$
AIM + TA	$\omega = 0.4$
ACM + TA	$t = 0.7$
RAIN-Merging	1000 samples, last 27 layers, ridge = 10^{-4}

A.7 Applicability to Other Reasoning Templates

REAM does not depend on RecZero's specific segment names or four-stage schema. For another structured template, its head-level

[1] https://github.com/hahahawu/Long-to-Short-via-Model-Merging
[2] AIM: https://github.com/ahnobari/ActivationInformedMerging; ACM: https://github.com/starrYYxuan/ACM; RAIN-Merging: https://github.com/K1nght/RAIN-Merging.

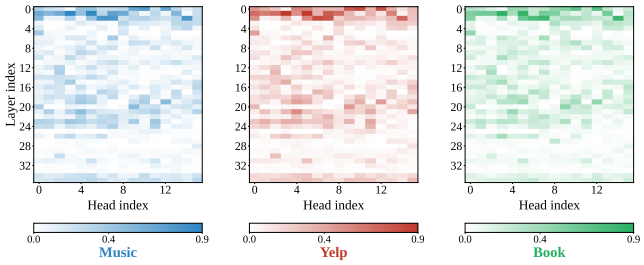


Figure 8: Decision faithfulness $\text{faith}_{\ell,h}$ per head, across layers, on Music, Yelp, and Book.

analysis can be applied to identifiable regions containing the information required at each reasoning stage, including retrieved, collaborative, or multimodal context. The region connecting the preceding reasoning to the final prediction can likewise be identified through perturbation analysis. We evaluate REAM only under RecZero's schema; fully unstructured traces would require an additional segmentation procedure, which we leave to future work.

A.8 Calibration Set Size and Estimate Stability

To assess whether $|\mathcal{D}_{\text{slow}}| = 500$ calibration samples are sufficient for stable retrieval-criticality estimates, we conduct two complementary analyses on Music.

Subsample stability. We recompute $R_{\ell,h}$ using random subsamples of 100, 200, 300, and 400 examples, with three seeds per size, and compare their top-5% critical-head sets and full rankings with those obtained from the full 500-sample set. Table 6 and Figure 9 report the mean Jaccard overlap and Spearman rank correlation. Both increase with sample size: at $n = 300$, the top-head overlap reaches 0.956 and the rank correlation reaches 0.989, while $n = 400$ recovers the full top-5% set and achieves a correlation of 0.995.

Table 6: Agreement between subsample and full-set estimates of $R_{\ell,h}$ on Music, averaged over three seeds.

Subsample size	Mean top-5% Jaccard	Mean Spearman ρ
100	0.933	0.964
200	0.933	0.979
300	0.956	0.989
400	1.000	0.995
500 (full)	1.000	1.000

Bootstrap stability. We further draw 1,000 bootstrap samples with replacement from the full 500-example calibration set and recompute $R_{\ell,h}$ for each draw. The top-5% critical heads (29 of 576) have a median relative confidence-interval width of 14.95%, compared with 20.88% across all heads. Moreover, 96.6% of these heads reappear in the bootstrap top-5% set in at least 900 of the 1,000 draws.

Together, these analyses show that retrieval-criticality estimates stabilise before 500 samples and are most reliable for the heads REAM is designed to protect.

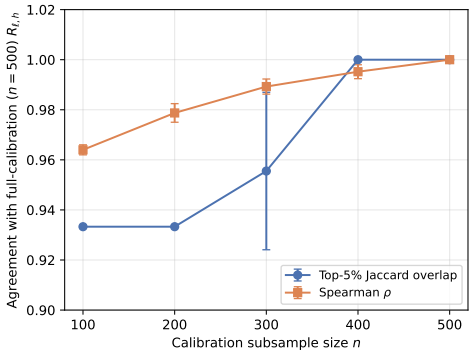


Figure 9: Agreement between subsample and full-set estimates of $R_{\ell,h}$ across calibration sizes on Music.

A.9 Retrieval Criticality and Decision Faithfulness: Sparsity and Causal Validation

We analyse $R_{\ell,h}$ and $\text{faith}_{\ell,h}$ (§4.1) on $\mathcal{D}_{\text{slow}}$, comprising 500 validation traces with correct ratings and all reasoning segments. For $R_{\ell,h}$, the evidence region $K^{(s)}(x)$ is defined once per trace and shared across generation steps.

A.9.1 Sparsity and Cross-Domain Stability. Retrieval criticality is highly concentrated yet distributed across model depth (Figure 7), while decision faithfulness exhibits a different spatial pattern (Figure 8): its mass is less concentrated but the top-5% heads are more localised within early-to-middle layers. Table 7 summarizes both signals' concentration and layer span across the three datasets. On Book, the top 5% of heads (29 of 576) account for 78.0% of retrieval mass, substantially exceeding both uniform allocation (Gini = 0) and a random-noise null (i.i.d. $U(0,1)$, expected Gini $\approx 1/3$). Both signals' top-5% head sets are stable across datasets: mean cross-domain Jaccard is 0.871 for $R_{\ell,h}$ and 0.794 for $\text{faith}_{\ell,h}$.

Table 7: Sparsity of the two head-level signals across datasets. Top-5% mass is the share of total retrieval (or faithfulness) mass held by the top 5% of heads.

Signal	Domain	Top-5% mass	Gini	Layer span
$R_{\ell,h}$ (retrieval)	Music	77.6%	0.920	3–34
	Book	78.0%	0.922	3–34
	Yelp	77.1%	0.920	3–34
$\text{faith}_{\ell,h}$ (faithfulness)	Music	22.6%	0.548	0–23
	Book	23.6%	0.553	0–23
	Yelp	24.0%	0.556	0–23

A.9.2 Independence Between Signals. Table 8 reports the Spearman correlation and top-5% head-set overlap between $\kappa_{\ell,h}$ and $\text{faith}_{\ell,h}$ in each domain. Correlation is negligible everywhere, and the top-5% head sets are correspondingly near-disjoint — consistent with the layer-distribution contrast in Table 7: the two signals capture distinct, largely independent reasoning roles.

A.9.3 Causal Validation. To confirm these heads are causally, not merely correlationally, critical, we mean-ablate them. For head (ℓ, h), let $o_{\ell,h}(x,t) \in \mathbb{R}^{d_h}$ denote its output slice at output-position token t of sample x. We precompute a reference mean activation

Table 8: Independence between $\kappa_{\ell,h}$ and $\text{faith}_{\ell,h}$: Spearman correlation and top-5% head-set overlap.

Domain	Spearman ρ	p-value	Top-5% Jaccard
Music	0.018	0.667	0.0% (0 shared)
Book	−0.001	0.976	0.0% (0 shared)
Yelp	−0.012	0.769	1.75% (1 shared)

over $\mathcal{D}_{\text{slow}}$,

$$\bar{o}_{\ell,h} = \frac{1}{\sum_x T_x} \sum_{x \in \mathcal{D}_{\text{slow}}} \sum_{t=1}^{T_x} o_{\ell,h}(x,t), \tag{9}$$

and overwrite each ablated head's output with this constant at every evaluation position, for every sample. This removes the head's input-dependent signal while preserving its average contribution to the residual stream; we avoid zero- or uniform-attention substitution, both of which collapse generation entirely.

Tables 9 and 10 report the resulting NLL and JSD ratios (ablated vs. a size-matched random-head baseline, averaged over three random draws) on a held-out set for both signals. Both effects confirm the respective head sets are causally important for the model's reasoning, despite selecting largely disjoint heads (Table 8).

Table 9: Causal validation using teacher-forced NLL. ΔNLL denotes the increase from the corresponding unablated baseline under top-5% signal-selected head ablation or size-matched random-head ablation averaged over three draws. Ratio = $\Delta\text{NLL}_{\text{selected}}/\Delta\text{NLL}_{\text{random}}$. Ratios are computed from unrounded values.

Signal	Domain	$\Delta\text{NLL}_{\text{selected}}$	$\Delta\text{NLL}_{\text{random}}$	Ratio
Retrieval criticality	Music	0.324	0.028	11.57×
	Book	0.562	0.114	4.93×
	Yelp	0.341	0.046	7.35×
Decision faithfulness	Music	0.751	0.078	9.65×
	Book	0.861	0.077	11.16×
	Yelp	0.623	0.094	6.63×

Table 10: Causal validation (JSD), same setup as Table 9. Ratio = critical/random.

Signal	Domain	Critical	Random	Ratio
$R_{\ell,h}$-critical	Music	0.119	0.036	3.29×
	Book	0.143	0.036	3.96×
	Yelp	0.108	0.034	3.19×
Faith-critical	Music	0.154	0.027	5.64×
	Book	0.159	0.031	5.20×
	Yelp	0.121	0.029	4.19×

B Method Design Justifications and Proofs

B.1 Token Weighting and Filtering for Retrieval Criticality

For each successful retrieval contributing to $R_{\ell,h}$ (§4.1), we weight the retrieved token w_t by $\tau_x(w_t) = \text{tf}_x(w_t)\text{idf}(w_t)$, measuring its informativeness within the sample's source text. This gives greater credit to rare, content-bearing tokens that distinguish a particular user or item, while downweighting common tokens that appear across many calibration samples. The following paragraphs define the term-frequency and inverse-document-frequency components.

Stopword and punctuation exclusion. Decoding steps corresponding to stopwords or punctuation tokens are excluded from Cond.1 and Cond.2 entirely, before the retrieval check is evaluated — not merely down-weighted — since such tokens trivially satisfy the retrieval criterion almost anywhere in the evidence region and would otherwise inflate $R_{\ell,h}$ with content-free successes.

Tokenisation and vocabulary. Both tf and df are computed over the model's subword vocabulary rather than whitespace-delimited words. Each source text is tokenised once, and every resulting token ID is treated as a distinct term. A token is excluded from both statistics if its decoded form either contains no alphanumeric character or, after lowercasing, appears in the NLTK English stopword list [5] — the same criterion used to filter retrieval events and the evidence-region tokens in $|K^{(s)}(x)|$.

Term frequency. For each sample x, we define

$$\text{tf}_x(w) = \frac{\text{count}(w, \text{doc}(x))}{\sum_{w'} \text{count}(w', \text{doc}(x))},$$

where $\text{doc}(x)$ concatenates the user-history and target-item text fields, excluding the separate average-rating fields attached to each (`user_avg_rating`, `target_item_avg_rating`), and the denominator counts all retained content tokens.

Inverse document frequency. We compute inverse document frequency over the calibration set $\mathcal{D}_{\text{slow}}$, treating each sample's document $\text{doc}(x)$ as one document in the corpus:

$$\text{idf}(w) = \log \frac{1+N}{1+\text{df}(w)} + 1,$$

where $N = |\mathcal{D}_{\text{slow}}|$ and $\text{df}(w)$ is the number of samples' documents containing w. Smoothing the numerator and denominator and adding 1 outside the logarithm keeps the weight finite and positive for both rare and universally occurring tokens.

B.2 Segment-Level Predictive Importance

We anchor decision faithfulness to MATCH because this segment is where user- and item-side evidence is synthesised into a compatibility judgement. If the RATE decision is grounded in the reasoning trace, it should rely on this judgement rather than bypassing it and returning to earlier evidence or prompt cues.

We validate this choice by measuring how much each reasoning segment affects the rate-token distribution. For each segment $s \in \{\text{USER}, \text{ITEM}, \text{MATCH}\}$, we replace all tokens in that segment with a neutral filler of identical length while keeping the rest of the trace unchanged. We then measure the shift in the output distribution at the RATE tokens. A larger shift indicates that the removed segment carries more predictive information for the final decision. We quantify this shift with Jensen–Shannon divergence:

$$I_s = \frac{1}{|D|} \sum_{x \in D} \frac{1}{|T_{\text{RATE}}(x)|} \sum_{t \in T_{\text{RATE}}(x)} \text{JSD}\big(p_t(\cdot \mid x) \,\|\, p_t(\cdot \mid x^{(-s)})\big), \tag{10}$$

where $x^{(-s)}$ denotes the trace with segment s replaced, and $T_{\text{RATE}}(x)$ denotes the decision-token positions.

Table 11 reports I_s with 95% bootstrap confidence intervals. Across all datasets, MATCH produces the largest distributional shift, 3.3–5.1× higher than either USER or ITEM. Its confidence intervals do not overlap with those of the other segments, indicating that the dominance of MATCH is statistically reliable.

Table 11: Segment importance I_s measured by mean JSD at the RATE tokens, with 95% bootstrap confidence intervals. The MATCH segment has the strongest effect on the final decision distribution across all datasets.

Dataset	USER	ITEM	MATCH
Book	0.0076 [0.0064, 0.0089]	0.0071 [0.0062, 0.0080]	0.0287 [0.0245, 0.0330]
Music	0.0101 [0.0086, 0.0117]	0.0077 [0.0065, 0.0089]	0.0520 [0.0448, 0.0594]
Yelp	0.0107 [0.0097, 0.0117]	0.0118 [0.0109, 0.0128]	0.0392 [0.0358, 0.0429]

The analysis shows that MATCH is the most direct segment-level carrier of predictive information for the final decision, consistent with its role as the compatibility-synthesis step.

B.3 Length Sensitivity of Decision Faithfulness

Decision faithfulness uses total attention mass, not per-token density, because the signal is meant to measure how much MATCH evidence enters the decision-time representation, not how much attention MATCH receives per token. We verify that this mass-based definition is robust to region-length sensitivity.

B.3.1 Length-normalised variants. Our main definition uses total attention mass because decision faithfulness is intended to measure evidence contribution to the decision representation. A length-normalised alternative instead measures attention per token. Since PROMPT, ANALYZE, and MATCH differ substantially in length, we evaluate this alternative definition directly.

For each region r, let $|\Omega_r(x,t)|$ be the number of positions in that region for trace x and decision token t. Density-faith first normalises the region mass at each decision token by the corresponding region length, then averages over tokens and calibration traces:

$$m^{r,\mathrm{dens}}_{\ell,h} = \frac{1}{|\mathcal{D}_{\mathrm{slow}}|} \sum_{x\in\mathcal{D}_{\mathrm{slow}}} \frac{1}{|T_{\mathrm{RATE}}(x)|} \sum_{t\in T_{\mathrm{RATE}}(x)} \frac{\sum_{\tau\in\Omega_r(x,t)} \mathrm{Att}^{(x)}_{\ell,h}[t,\tau]}{|\Omega_r(x,t)|}. \tag{11}$$

This uses each sample's own region length, rather than a fixed dataset-level length. The density-based faithfulness score is

$$\mathrm{faith}^{\mathrm{dens}}_{\ell,h} = \frac{m^{\mathrm{MATCH,dens}}_{\ell,h}}{m^{\mathrm{PROMPT,dens}}_{\ell,h} + m^{\mathrm{ANALYZE,dens}}_{\ell,h} + m^{\mathrm{MATCH,dens}}_{\ell,h} + \delta}. \tag{12}$$

The $\sqrt{n}$-faith variant follows the same construction, replacing $|\Omega_r(x,t)|$ with $\sqrt{|\Omega_r(x,t)|}$ before forming the analogous ratio $\mathrm{faith}^{\sqrt{n}}_{\ell,h}$.

Table 12 shows that mass-based faithfulness achieves the lowest MAE and RMSE on both datasets. Length-normalised variants change head rankings but do not improve downstream performance.

Table 12: Length-sensitivity robustness of decision faithfulness on a matched held-out subset. Density-faith normalises attention mass by region length, while $\sqrt{n}$-faith applies a weaker correction.

Domain	Variant	MAE	RMSE
Yelp	Mass-based faithfulness	0.7513	1.0685
	Density-faith	0.7544	1.0751
	$\sqrt{n}$-faith	0.7575	1.0785
Music	Mass-based faithfulness	0.5292	0.8255
	$\sqrt{n}$-faith	0.5308	0.8317
	Density-faith	0.5311	0.8335

B.3.2 Uniform-attention baseline and allocation invariance.

Uniform-attention baseline. A head with no genuine regional preference would still assign nonzero attention to MATCH, since MATCH occupies part of the visible context. Let m^r_0 denote the attention mass that region r would receive under uniform attention, which depends only on region length. The corresponding baseline for decision faithfulness is

$$\eta = \frac{m^{\mathrm{MATCH}}_0}{m^{\mathrm{PROMPT}}_0 + m^{\mathrm{ANALYZE}}_0 + m^{\mathrm{MATCH}}_0}. \tag{13}$$

This value is identical across heads because it depends only on region sizes, not on head behaviour. In our calibration traces, $\eta \approx 0.036$ on Music and $\eta \approx 0.048$ on Yelp.

Allocation invariance of baseline subtraction. The allocation is invariant to any constant shift applied to faith_b, which we use here to show that subtracting the uniform-attention baseline η has no effect on the merged model. For a shift $\widetilde{\mathrm{faith}}_b = \mathrm{faith}_b - \eta$, substituting into the perturbation weight gives

$$\begin{aligned} \widetilde{s}_b &= \exp(\gamma(\kappa_b + \mathrm{faith}_b - \eta))\, d_b \\ &= \exp(\gamma(\kappa_b + \mathrm{faith}_b))\, e^{-\gamma\eta} d_b = e^{-\gamma\eta} s_b. \end{aligned} \tag{14}$$

Every block's perturbation weight is rescaled by the same constant $e^{-\gamma\eta}$, independent of b. Since the injection budget is a fixed fraction of the total perturbation weight,

$$\varepsilon = \rho\, \bar{\alpha}^2 \sum_{b\in\mathcal{B}} s_b, \tag{15}$$

the budget rescales by the same factor, so the water-filling solution $\alpha^\star_b$ is unchanged before and after the shift. This argument holds for any constant η and any definition of faith_b, so recentering the faithfulness signal — for the uniform-attention baseline or otherwise — does not change the merged model, supporting total decision-time attention mass as the operational measure of decision faithfulness.

B.4 Derivation of the Fisher-Weighted Update-Sensitivity Surrogate

B.4.1 Local Quadratic Expansion. In §4.2, the teacher-forced loss is computed using the slow-thinking model parameters θ_S. To examine how this loss changes as the parameters are perturbed during merging, we denote by $\ell_\theta(x)$ the teacher-forced loss computed using an arbitrary parameter vector θ:

$$\ell_\theta(x) = -\frac{1}{|Y(x)|} \sum_{t=1}^{|Y(x)|} \log p_\theta(y_t \mid y_{<t}, \mathrm{prompt}(x)). \tag{16}$$

Setting $\theta = \theta_S$ recovers the loss defined in §4.2. We average this loss over the calibration set:

$$\bar{\ell}(\theta) = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_{x \in \mathcal{D}_{\text{slow}}} \ell_\theta(x). \tag{17}$$

Let b index a merge unit: either the paired Q/O parameters of a query head or the paired K/V parameters of a GQA group, and let $\mathcal{B}$ denote the set of all such units. For each $b \in \mathcal{B}$, let $\Delta_{F,b}$ denote the fast-thinking update restricted to all parameters belonging to unit b.

Given merge coefficients $\{\alpha_b\}_{b \in \mathcal{B}}$, the total parameter change applied to the slow-thinking model is

$$\Delta_\theta(\boldsymbol{\alpha}) = \sum_{b \in \mathcal{B}} \alpha_b \Delta_{F,b},$$

where $\boldsymbol{\alpha}$ collects the coefficients assigned to all merge units. The resulting merged model is

$$\theta(\boldsymbol{\alpha}) = \theta_S + \Delta_\theta(\boldsymbol{\alpha}).$$

Full injection of the fast-thinking update corresponds to $\alpha_b = 1$ for every merge unit b.

To assess how the merged model $\theta(\boldsymbol{\alpha})$ perturbs the calibration loss, we expand $\bar{\ell}$ around θ_S. A second-order Taylor expansion gives

$$\begin{aligned} \bar{\ell}(\theta(\boldsymbol{\alpha})) - \bar{\ell}(\theta_S) &= \nabla_\theta \bar{\ell}(\theta_S)^\top \Delta_\theta(\boldsymbol{\alpha}) \\ &\quad + \frac{1}{2} \Delta_\theta(\boldsymbol{\alpha})^\top H_S \Delta_\theta(\boldsymbol{\alpha}) + O\big(\|\Delta_\theta(\boldsymbol{\alpha})\|_2^3\big), \end{aligned} \tag{18}$$

where $H_S = \nabla_\theta^2 \bar{\ell}(\theta_S)$ is the Hessian of the calibration loss at the slow-thinking model. Although θ_S is already trained, the gradient $\nabla_\theta \bar{\ell}(\theta_S)$ need not vanish because $\mathcal{D}_{\text{slow}}$ is a filtered calibration set rather than the original training set. We omit the first-order term because it is signed and can be negative or cancel across merge units depending on the alignment between $\Delta_\theta(\boldsymbol{\alpha})$ and the calibration gradient, making it unsuitable as a non-negative perturbation cost. By contrast, under a positive-semidefinite (PSD) curvature approximation, the quadratic term is non-negative and measures the local curvature along the merge-update direction. We therefore use it as a non-negative local sensitivity surrogate:

$$C_H(\boldsymbol{\alpha}) = \frac{1}{2} \Delta_\theta(\boldsymbol{\alpha})^\top H_S \Delta_\theta(\boldsymbol{\alpha}). \tag{19}$$

This quantity captures the local curvature of the calibration loss along the direction of the merge-induced parameter change. A larger value signals that the fast-thinking update perturbs the slow-thinking model along a direction of higher loss curvature, and therefore carries greater risk to model behaviour.

B.4.2 From Hessian to a Diagonal Empirical-Fisher Surrogate.

Hessian decomposition. For a negative log-likelihood objective, the Hessian contains two sources of curvature, following the generalised Gauss–Newton decomposition of [29]. The first captures how the loss curves with respect to the model logits and how parameter changes propagate to those logits; this forms the generalised Gauss–Newton component. The second arises from the nonlinear dependence of the logits on the model parameters and forms a residual term.

For each example x, let $z_\theta(x)$ denote the vector obtained by stacking the logits used at all teacher-forced response positions, and let $J_\theta(x) = \partial z_\theta(x)/\partial\theta$ denote its Jacobian with respect to the model parameters. Applying the chain rule twice and averaging over the calibration set gives

$$\begin{aligned} H_S &= \nabla_\theta^2 \bar{\ell}(\theta_S) \\ &= \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_{x \in \mathcal{D}_{\text{slow}}} \Bigg[\underbrace{J_{\theta_S}(x)^\top \, \nabla_z^2 \ell(z;x)\big|_{z=z_{\theta_S}(x)} \, J_{\theta_S}(x)}_{G_S(x)} \\ &\qquad + \underbrace{\sum_k \frac{\partial \ell(z;x)}{\partial z_k}\bigg|_{z=z_{\theta_S}(x)} \nabla_\theta^2 z_{\theta,k}(x)\big|_{\theta=\theta_S}}_{R_S(x)} \Bigg] \\ &= G_S + R_S, \end{aligned} \tag{20}$$

where the index k runs over all scalar entries of the stacked logit vector, and $G_S = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_x G_S(x)$, $R_S = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_x R_S(x)$.

Model Fisher. We next relate G_S to the Fisher information matrix, which measures how sensitive the model's predicted next-token distribution is to changes in its parameters. At teacher-forced decoding position t of example x, let $c_t(x) = (\text{prompt}(x), y_{<t})$ denote the fixed context preceding that position. Given this context, the model produces a logit $z_{\theta,k}(x,t)$ for each vocabulary token k, and the softmax function converts these logits into the next-token distribution

$$p_\theta(y \mid c_t(x)) = \frac{\exp\big(z_{\theta,y}(x,t)\big)}{\sum_{k \in \mathcal{V}} \exp\big(z_{\theta,k}(x,t)\big)},$$

where $\mathcal{V}$ denotes the vocabulary. Let $J_\theta(x,t) = \partial z_\theta(x,t)/\partial\theta$ denote the Jacobian of the logit vector at position t, and define the score function $s_\theta(y,x,t) = \nabla_\theta \log p_\theta(y \mid c_t(x))$. The token-level conditional model Fisher at this position is

$$\begin{aligned} \mathcal{F}_S(x,t) &= \mathbb{E}_{y \sim p_{\theta_S}(\cdot \mid c_t(x))} \left[s_{\theta_S}(y,x,t) s_{\theta_S}(y,x,t)^\top \right] \\ &= \sum_{y \in \mathcal{V}} p_{\theta_S}(y \mid c_t(x)) \, s_{\theta_S}(y,x,t) s_{\theta_S}(y,x,t)^\top, \end{aligned}$$

where the second form follows because the output vocabulary is discrete. Thus, the model Fisher considers all possible next tokens, weighted by the probabilities assigned to them by the slow-thinking model, rather than using only the observed target token y_t. Averaging over the same calibration examples and response positions as the teacher-forced loss gives

$$\mathcal{F}_S = \frac{1}{|\mathcal{D}_{\text{slow}}|} \sum_{x \in \mathcal{D}_{\text{slow}}} \frac{1}{|Y(x)|} \sum_{t=1}^{|Y(x)|} \mathcal{F}_S(x,t). \tag{21}$$

Gauss–Newton equals Fisher. We next establish that, for categorical negative log-likelihood with softmax logits, the model Fisher matrix $\mathcal{F}_S$ coincides exactly with the generalised Gauss–Newton matrix G_S [29]. Consider a fixed example x and decoding position t, and treat the per-position loss as a function of a free logit vector z. Let $p = p_\theta(\cdot \mid c_t(x))$ denote the corresponding softmax distribution, and let $\ell_t(z) = -\log p_y$ be the loss for observed token y. The Hessian of this loss with respect to the logits is

$$\nabla_z^2 \ell_t(z) = \text{diag}(p) - pp^\top,$$

which is independent of the observed token. Meanwhile, the log-probability score is

$$\nabla_z \log p(y) = e_y - p,$$

where e_y denotes the one-hot vector for token y. Taking its expected outer product under $y \sim p$ yields

$$\mathbb{E}_{y\sim p}\left[(e_y - p)(e_y - p)^\top\right] = \mathbb{E}_{y\sim p}\left[e_y e_y^\top\right] - pp^\top = \mathrm{diag}(p) - pp^\top.$$

The logit-space curvature of the loss therefore equals the covariance of the logit-space score. Mapping both quantities into parameter space through $J_{\theta_S}(x,t)$ gives $G_S(x,t) = J_{\theta_S}(x,t)^\top[\mathrm{diag}(p) - pp^\top]J_{\theta_S}(x,t) = \mathcal{F}_S(x,t)$. Moreover, because the teacher-forced loss averages the negative log-likelihood over response positions, $G_S(x) = \frac{1}{|Y(x)|}\sum_{t=1}^{|Y(x)|} G_S(x,t)$, and averaging over the calibration set therefore yields $G_S = \mathcal{F}_S$.

Discarding the residual. Following the standard generalised Gauss–Newton approximation, we discard the residual curvature term R_S: near a well-trained solution such as θ_S, the per-token prediction error entering R_S is small, so its contribution to the Hessian is expected to be small relative to G_S. The Hessian is then approximated by the positive-semidefinite model Fisher:

$$H_S = G_S + R_S \approx G_S = \mathcal{F}_S. \tag{22}$$

Empirical-Fisher approximation. Computing the model Fisher $\mathcal{F}_S$ exactly would require evaluating the score for every possible vocabulary token at every response position. This is infeasible for a large language model. We therefore replace it with the empirical Fisher, which uses the observed teacher-forced responses in the calibration set.

For each example x, define the gradient of its teacher-forced loss at the slow-thinking parameters as $g(x) = \nabla_\theta \ell_\theta(x)|_{\theta=\theta_S}$. The empirical Fisher is the average outer product of these gradients:

$$\widehat{\mathcal{F}}_{\mathrm{emp}} = \frac{1}{|\mathcal{D}_{\mathrm{slow}}|}\sum_{x\in\mathcal{D}_{\mathrm{slow}}} g(x)g(x)^\top. \tag{23}$$

Unlike the model Fisher in Eq. (21), this estimator uses the observed response $Y(x)$ rather than averaging over tokens drawn from the model's predictive distribution. It is therefore not exactly equal to either $\mathcal{F}_S$ or H_S, but provides a tractable positive-semidefinite curvature surrogate [23].

Storing the full empirical-Fisher matrix remains infeasible because its size grows quadratically with the number of model parameters. We therefore retain only its diagonal entries, $\widehat{\mathcal{F}}_{\mathrm{emp}} \approx \mathrm{diag}(F_1, \ldots, F_{|\theta|})$, where

$$F_j = \frac{1}{|\mathcal{D}_{\mathrm{slow}}|}\sum_{x\in\mathcal{D}_{\mathrm{slow}}}\left(\left.\frac{\partial \ell_\theta(x)}{\partial \theta_j}\right|_{\theta=\theta_S}\right)^2.$$

Each F_j measures how strongly the calibration loss responds locally to changes in parameter θ_j. A larger value indicates that a small change to that parameter is more likely to disturb the slow-thinking behaviour captured by the calibration set.

B.5 Derivation of the Head-Level Update Sensitivity

We use the diagonal empirical Fisher to approximate the curvature cost of the merged parameter change $\Delta_\theta(\boldsymbol{\alpha}) = \sum_{b\in\mathcal{B}} \alpha_b \Delta_{F,b}$, where b indexes either a paired Q/O head or a paired K/V group. The resulting sensitivity surrogate is

$$C_F(\boldsymbol{\alpha}) = \frac{1}{2}\Delta_\theta(\boldsymbol{\alpha})^\top \mathrm{diag}(F_1, \ldots, F_{|\theta|})\Delta_\theta(\boldsymbol{\alpha}) = \frac{1}{2}\sum_j F_j\left(\Delta_{\theta,j}(\boldsymbol{\alpha})\right)^2.$$

Each merge unit b comprises two individual projection slices, denoted by $c \in C(b)$: the Q and O slices of a query head, or the K and V slices of a GQA group. Both slices receive the same coefficient α_b. For each slice, its update sensitivity is $d_c = \sum_{j\in c} F_j(\Delta_{F,j})^2$, as defined in Eq. (6). Because the projection slices are disjoint, substituting the merge-induced update gives

$$\begin{aligned} C_F(\boldsymbol{\alpha}) &= \frac{1}{2}\sum_{b\in\mathcal{B}} \alpha_b^2 \sum_{c\in C(b)} d_c \\ &= \sum_{b\in\mathcal{B}} \alpha_b^2 d_b, \end{aligned} \tag{24}$$

where $d_b = \frac{1}{2}\sum_{c\in C(b)} d_c$ is the averaged sensitivity of the two projection slices in merge unit b. This corresponds to $d_{\ell,h}^{QO} = (d_{\ell,h}^{Q} + d_{\ell,h}^{O})/2$ for a query head and $d_{\ell,g}^{KV} = (d_{\ell,g}^{K} + d_{\ell,g}^{V})/2$ for a GQA group, matching the definition in §4.2.

Thus, d_b combines the sensitivity of the slow-thinking model with the magnitude of the fast-thinking update for merge unit b. A larger d_b indicates that the proposed update is more likely to disturb the behaviour captured by the calibration loss. Its contribution to the curvature surrogate scales quadratically with α_b, motivating the perturbation constraint in §4.3.

B.6 Form of the Perturbation-Cost Multiplier

We ablate the link function used to map retrieval criticality and decision faithfulness into a multiplier on Fisher risk. The main method uses

$$s_b = m_b d_b, \qquad m_b = \exp(\gamma(\kappa_b + \mathrm{faith}_b)).$$

We compare this exponential multiplier with two alternatives using the same signals:

$$m_b^{\mathrm{lin}} = \max(0.05,\ 1 + \gamma(\kappa_b + \mathrm{faith}_b)),$$
$$m_b^{\mathrm{sig}} = 1 + \gamma(\sigma(z(\kappa_b)) + \sigma(z(\mathrm{faith}_b))).$$

Here $z(x) = (x - \bar{x})/s_x$ standardises each signal across heads, and $\sigma(x) = 1/(1+e^{-x})$ maps it to $(0,1)$. Since κ_b and faith_b are already bounded in $[0,1]$, this variant stretches relative differences by z-scoring, then compresses them through the sigmoid. The linear floor is only a numerical safeguard for near-zero denominators in the water-filling solution.

The three forms differ mainly in dynamic range: the exponential form is multiplicative and unbounded, so blocks with jointly high retrieval criticality and decision faithfulness receive substantially larger perturbation weights than either signal alone would produce.

Table 13: Dynamic range of the perturbation multiplier under each link function, measured across all heads on Music.

Link function	Min	Max	Ratio
Exponential	2.3	108	~47×
Linear	1.8	5.7	~3.2×
Sigmoid	2.7	6.3	~2.3×

As shown in Table 13, the exponential form produces a substantially wider spread of perturbation weights than the linear and sigmoid alternatives. This allows the allocation to separate high-cost blocks more sharply, whereas the alternatives compress most blocks into a narrower range.

Table 14: Perturbation-multiplier ablation on Yelp ($n = 400$).

Link function	MAE	RMSE
Exponential	**0.7348**	**1.0656**
Sigmoid	0.7617	1.0970
Linear	0.7686	1.1030

Table 14 compares the three multiplier forms on the same matched Yelp subset. The exponential form achieves the lowest MAE and RMSE, supporting its use in the main method. Because this ablation uses a smaller matched subset, the values are reported separately from the full test-set production results.

B.7 Normalisation Choices in the Perturbation Cost

The perturbation cost

$$s_b = \exp(\gamma(\kappa_b + \text{faith}_b))\, d_b$$

combines three quantities with different meanings and scales. We therefore normalise them differently rather than applying a uniform preprocessing rule.

Retrieval criticality. The raw retrieval criticality $R_{\ell,h}$ has no natural scale and is highly skewed. Since retrieval criticality enters the exponent, using raw $R_{\ell,h}$ would allow extreme-valued heads to dominate the allocation. We therefore map $R_{\ell,h}$ through the log–z-score–sigmoid transform defined in §4, producing a bounded and comparable value κ_b before exponentiation.

Decision faithfulness. We use faith_b in its raw form. Unlike $R_{\ell,h}$, decision faithfulness is already bounded and directly interpretable as the share of tracked decision-time attention directed to MATCH. We therefore apply no additional transformation.

Fisher update sensitivity. We also retain d_b on its original Fisher-weighted scale. Unlike κ_b and faith_b, d_b is not a relative score but an estimate of update-induced perturbation: loss sensitivity weighted by the squared update magnitude. Nonlinear or rank transforms would discard this magnitude and weaken the link between the perturbation constraint and the Fisher-weighted cost. Uniform rescaling leaves the solution unchanged, since it rescales s_b and ε equally; but transforming relative magnitudes changes the allocation.

B.8 KKT Derivation of the Water-Filling Solution

The allocation problem in §4.3 is

$$\max_{\alpha} \sum_{b\in\mathcal{B}} w_b\alpha_b \qquad \text{s.t.} \qquad \sum_{b\in\mathcal{B}} s_b\alpha_b^2 \le \varepsilon, \qquad 0 \le \alpha_b \le \bar{\alpha}.$$

Since the objective is linear and the feasible set is convex when $s_b \ge 0$, the KKT conditions are sufficient for global optimality.

The coefficients α_b share a single perturbation budget ε, so they cannot be chosen independently. We introduce a multiplier $\mu \ge 0$ for the shared constraint; for fixed μ, the problem separates across blocks, and μ is later adjusted until the resulting coefficients satisfy the budget condition. The Lagrangian objective is

$$\sum_{b\in\mathcal{B}} \left(w_b\alpha_b - \mu s_b\alpha_b^2\right) + \mu\varepsilon.$$

Thus each coordinate solves

$$\max_{0\le\alpha_b\le\bar{\alpha}} \; w_b\alpha_b - \mu s_b\alpha_b^2.$$

For $s_b > 0$ and $\mu > 0$, the unconstrained stationary point satisfies

$$\frac{\partial}{\partial\alpha_b}\left(w_b\alpha_b - \mu s_b\alpha_b^2\right) = w_b - 2\mu s_b\alpha_b = 0, \qquad \alpha_b(\mu) = \frac{w_b}{2\mu s_b}.$$

Applying the box constraint gives the water-filling solution

$$\alpha_b^\star = \text{clip}\left(\frac{w_b}{2\mu s_b}, 0, \bar{\alpha}\right).$$

The multiplier μ is not a fixed hyperparameter; it is solved separately for each merge. In the binding case, $\mu > 0$ is found by bisection

$$\sum_{b\in\mathcal{B}} s_b(\alpha_b^\star)^2 = \varepsilon.$$

If full injection already satisfies the perturbation constraint, then the constraint is slack. Complementary slackness sets $\mu = 0$, and the optimal solution is $\alpha_b^\star = \bar{\alpha}$ for all blocks.

When $s_b = 0$, block b carries no perturbation penalty. If $w_b > 0$, the objective is maximised by setting $\alpha_b^\star = \bar{\alpha}$; if $w_b = 0$, all values in $[0, \bar{\alpha}]$ are equivalent, and we set $\alpha_b^\star = \bar{\alpha}$ by convention.

B.9 Generalisability of FFN Exclusion Across Model Scale and Family

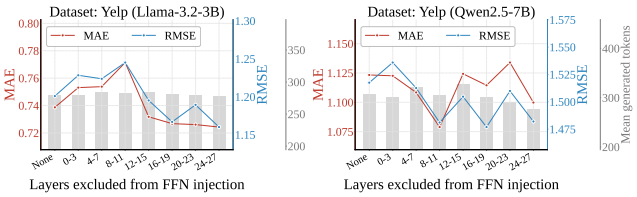


Figure 10: FFN exclusion sweep on Yelp for Llama-3.2-3B-Instruct (left) and Qwen2.5-7B-Instruct (right), both using four-layer exclusion windows. Bars show mean generated tokens; lines show MAE and RMSE.

To assess late-layer FFN exclusion beyond the 3B setup in §5.4.3, we repeat the Yelp sweep using Llama-3.2-3B-Instruct and Qwen2.5-7B-Instruct (Figure 10). Both contain 28 layers, compared with 36 in Qwen2.5-3B. We therefore exclude four-layer windows to preserve a comparable fraction of model depth, rather than reusing six-layer windows for the deeper backbone. The window length is therefore scaled to the backbone depth, while its location is selected by the validation sweep. Once selected, the resulting window is fixed for all evaluations using that backbone.

On Llama-3.2-3B, excluding progressively later layers improves both MAE and RMSE, with layers 24–27 yielding the best accuracy–efficiency trade-off. This is consistent with the main-text finding that late FFNs are particularly sensitive to the fast-thinking update. On Qwen2.5-7B, however, the pattern is less monotonic: excluding either layers 8–11 or 24–27 gives competitive accuracy, whereas

excluding layers 20–23 produces the worst MAE and RMSE among the tested ranges. These results support excluding late FFN layers as a robust default across model scales and families: doing so consistently improves the accuracy–efficiency trade-off relative to unrestricted merging, even though the single most sensitive range shifts modestly across backbones. Accordingly, late-layer exclusion serves as the default strategy, but the exact exclusion window should be validated once for each new backbone rather than transferred directly across architectures.