KARMA: Knowledge graph-based Automated Reasoning Materialization and Alignment

Jinkyeong Choi, Chaebin Jeong, and Donghyeon Park*
Sejong University
Seoul, Republic of Korea
jjinchoi@sju.ac.kr chaibin2000@sju.ac.kr parkdh@sejong.ac.kr

Abstract

Template-based contrastive synthesis is scalable, but its candidates often differ only in a few entity-slots while sequence-level optimization spreads supervision over mostly shared templates. We formalize this as the Resolution Mismatch Problem and propose KARMA, which enumerates schema-constrained paths over domain knowledge graphs and verbalizes them into slot-aligned contrastive candidates. Slot-Parallel Alignment (SPA) then applies a decoupled slot-level objective to route preference supervision to discriminative entity-slots, with slot-aware masked attention serving as an optional packed-evaluation implementation. Across biomedical, computer-science, and chemistry benchmarks, KARMA outperforms base LLM and same-data SFT baselines, and compares favorably with sequence- and token-level preference methods.

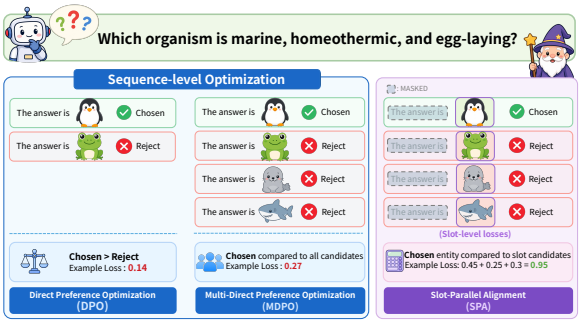


Figure 1: The Resolution Mismatch Problem in preference optimization. (a) DPO compares a single chosen-rejected pair at the sequence level. (b) Multi-DPO extends this to multiple rejected candidates, still at the sequence level. (c) Our Slot-Parallel Alignment (SPA) compares candidates sharing the same template and aligns preference at the resolution of discriminative entity-slots. The loss values illustrate supervision granularity and are not numerically comparable across objectives.

1 Introduction

Recent progress in LLM reasoning has increased the demand for reasoning supervision (Wei et al., 2022; Guo et al., 2025; Jaech et al., 2024). However, scaling such supervision remains bottlenecked by trade-offs among reliability, diversity, and cost (Uesato et al., 2022; Lightman et al., 2024).

Prior approaches address this trade-off in three ways. (1) Human expert annotation yields reliable reasoning chains but is prohibitively expensive at the process level (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024b). (2) Model-generated synthetic data improves scalability (Wang et al., 2023; Xu et al., 2024), but can inherit source-model errors and degrade under recursive use (Gudibande et al., 2023; Shumailov et al., 2024). (3) Rule- or template-based generation reduces cost through programmatic construction, but can restrict data diversity (Riaz et al., 2025). Despite these differences in how reasoning data is generated, a structural limitation emerges once it is used for contrastive supervision: candidate differences are often concentrated in a few entity-slots, while supervision is applied at the full-sequence level.

We formalize this gap as the *Resolution Mismatch Problem* (Fig. 1): preference is often determined by a few discriminative entity-slots, while optimization is applied to the full sequence. *The bottleneck is not how many candidates we compare, but at what resolution we compare them.* Standard DPO (Rafailov et al., 2023) (Fig. 1-a) and its multi-candidate extensions (Fig. 1-b) widen the comparison set but retain sequence-level granularity (Song et al., 2024; Liu et al., 2025b), which can dilute the preference signal associated with informative entity-slots. We examine this behavior directly through gradient analysis, measuring how sequence-, token-, and slot-level objectives

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 20513–20531
October 24-29, 2026 ©2026 Association for Computational Linguistics

distribute sensitivity across entity and template tokens.

Knowledge graphs provide a natural way to address this mismatch. Prior work uses KG paths as structured intermediate traces for reasoning (Luo et al., 2024; Sun et al., 2024; Tan et al., 2025). We use this path structure differently for contrastive data construction: under fixed schemas, alternative paths are verbalized with a shared template, with differences localized to entity-slots. We instantiate this idea in **KARMA** (**K**nowledge graph-based **A**utomated **R**easoning **M**aterialization and **A**lignment), a unified framework whose synthesis pipeline produces such candidates and whose learning component, **Slot-Parallel Alignment (SPA)**, operates directly on them (Fig. 1-c). KARMA uses the KG only at data-construction time; the trained model is evaluated without graph access, making inference-time KG grounding methods complementary rather than direct alternatives. Prior KG-to-text methods verbalize structured facts into natural language (Gardent et al., 2017; Agarwal et al., 2021). The synthesis pipeline verbalizes selected KG paths under a shared template and uses recurrence-based support as a structural prior for preference assignment. This exposes explicit discriminative entity-slots while expanding factual coverage beyond hand-written templates.

On the learning side, SPA applies a decoupled slot-level objective that supervises construction-exposed entity slots while retaining the shared template as a language-modeling signal. For efficient packed evaluation, SPA can optionally use slot-aware masked attention to approximate per-candidate log-likelihoods in a single forward pass without changing the slot-level supervision signal. Unlike prior token-level methods that identify, estimate, or learn token importance (Lin et al., 2025; Liu et al., 2025a; Yang et al., 2025a; Christopoulou et al., 2025; Yang et al., 2026), SPA targets entity-slots that are explicitly exposed by construction in KARMA-synthesized data.

When data and optimization share the same slot-level resolution, preference learning can focus on discriminative evidence while preserving shared templates. We evaluate KARMA across biomedical, chemistry, and computer-science benchmarks, where it improves over base LLM and SFT baselines on most benchmarks and compares favorably with sequence- and token-level preference methods. Our contributions are:

- **Aligned contrastive synthesis from KGs:** We propose a KG-grounded synthesis pipeline that constructs chosen-rejected candidates sharing a verbalization template by design, localizing preference differences to entity-slots and using KG path recurrence as a structural prior for preference assignment.

- **Slot-Parallel Alignment (SPA):** We introduce a preference optimization method whose resolution matches the data: a decoupled slot-level objective applies preference and anchor losses to construction-exposed entity slots while preserving shared templates as a language-modeling signal. We further provide an optional slot-aware masked attention implementation that reduces redundant candidate evaluation without estimating token importance.

2 Related Work

Trade-offs in Reasoning Supervision Reasoning supervision faces a cost-coverage trade-off: human process supervision yields reliable step-level signals but is prohibitively expensive (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024b), model-generated data scales better (Wang et al., 2023; Luo et al., 2025) but can inherit source-model errors and degrade under recursive use (Gudibande et al., 2023; Shumailov et al., 2024), and template-based generation lowers cost but can restrict data diversity (Riaz et al., 2025). When such data is used for contrastive supervision, candidate differences can become concentrated in a few entity-slots. KARMA turns this localized variation into a resource: by grounding candidate construction in KG paths, it expands factual coverage while keeping discriminative variation localized to entity-slots by construction.

Fine-grained Preference Alignment Sequence-level preference optimization applies supervision to sequence-level scores, including pairwise objectives (Rafailov et al., 2023; Cho et al., 2025) and ranking-based objectives over multiple candidates (Song et al., 2024; Liu et al., 2025b). When candidates share most tokens and differ only at a few positions, this granularity does not explicitly localize supervision to the discriminative positions. Token-level methods make preference optimization more fine-grained through token-wise KL or reward regularization (Zeng et al., 2024;

Zhou et al., 2025), while other methods identify or weight important tokens using contrastive estimation, importance sampling, reward estimation, or learned masks (Lin et al., 2025; Liu et al., 2025a; Yang et al., 2025a; Christopoulou et al., 2025; Yang et al., 2026). SPA instead exploits a structural prior from KARMA-synthesized data: discriminative entity-slots are known by construction, so supervision routes to them directly without estimation.

Knowledge Graph Verbalization Prior work verbalizes structured KG facts into natural language for data generation and language-model adaptation (Gardent et al., 2017; Agarwal et al., 2021). KG-augmented reasoning methods instead use graph paths or graph-derived plans to support retrieval and reasoning (Wang et al., 2024a; Luo et al., 2024; Sun et al., 2024; Tan et al., 2025). KARMA uses this structure differently: rather than supplying graph evidence at inference, it constructs multiple training candidates that share a source-target pair, relation schema, and verbalization template, exposing intermediate entity variations as slot-level contrastive supervision.

3 Methodology

3.1 Overview

We propose **KARMA**, a knowledge graph (KG)-grounded contrastive synthesis pipeline, and **SPA**, a slot-level preference optimization method co-designed for KARMA-synthesized data. KARMA enumerates alternative KG paths between fixed source-target entities under a fixed relational schema and verbalizes them into structurally aligned contrastive candidates that share a template scaffold while differing only at discriminative entity-slots. SPA then aligns preference optimization with this entity-slot-level structure through two core learning mechanisms: (i) slot-parallel organization of candidate variants under shared context, and (ii) a decoupled hybrid objective that applies preference supervision to entity slots while preserving the shared template. Slot-aware masked attention is an optional packed-evaluation mechanism that reduces redundant forward passes by approximating per-candidate log-likelihoods in a single forward pass. Together, KARMA and SPA realize a single principle: **align the granularity of supervision with the structural granularity of the data**.

3.2 KARMA Data Synthesis: KG-Grounded Contrastive Candidate Construction

KARMA constructs slot-aligned contrastive candidates from a domain knowledge graph in three steps: it enumerates schema-constrained paths between source-target entity pairs, selects top-K alternatives per candidate pool by balancing support and structural diversity, and verbalizes the selected paths with a schema-specific template shared within the pool. The resulting candidates share the same source and target entities and relation schema, differing only at their intermediate entity slots. This slot-localized variation is the interface to SPA: it confines contrastive information to entity slots, exposing preference signal at the same granularity as the data variation.

3.2.1 Path Enumeration

KARMA first constructs candidate pools by enumerating schema-constrained multi-hop paths from a domain knowledge graph (KG) $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of entities and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{R} \times \mathcal{N}$ is the set of typed edges over relation types $\mathcal{R}$. A relation schema specifies an ordered pattern of typed edges that candidate paths must instantiate. Schemas are exhaustively enumerated rather than selected for individual source-target pairs, as detailed in Appendix A.2. KARMA uses such schemas to collect alternative paths that share the same schema and source and target entities while differing only in their intermediate entities (Fig. 2A).

For a given source entity e_s, target entity e_t, and relation schema, KARMA enumerates a schema-constrained candidate pool $\mathcal{P}$, a source-to-target subgraph of $\mathcal{G}$, whose elements take the form

$$p^m = \left(e_s, e_1^{(m)}, e_2^{(m)}, \ldots, e_{n-1}^{(m)}, e_t\right), \quad (1)$$

where $m = 1, \ldots, M$ indexes paths within the pool, n denotes the number of hops in the relation schema, $e_j^{(m)}$ denotes the entity at the j-th intermediate position of path p^m for $j = 1, \ldots, n-1$, and e_s, e_t denote the source and target entities.

Each such pool corresponds to one source-target pair and serves as the candidate set for one training instance. Paths in the same pool share the same relation schema and source and target entities, but may differ in their intermediate entities. Equivalently, each intermediate slot j induces a set of distinct slot entities $\mathcal{S}_j = \{e_{j,1}, \ldots, e_{j,d_j}\}$, from which each path selects one entity, so $e_j^{(m)} \in \mathcal{S}_j$.

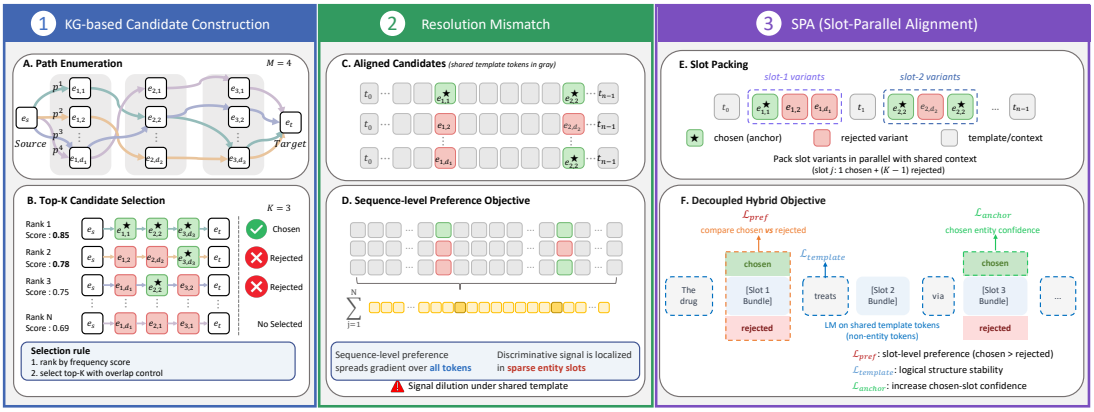


Figure 2: Overview of KARMA and SPA. KARMA synthesizes structurally aligned contrastive candidates from knowledge-graph paths via path enumeration, support-based top-K selection, and template verbalization. Because candidate differences are localized to sparse entity-slots, sequence-level preference optimization can suffer from signal dilution. SPA addresses this mismatch with a decoupled slot-level objective that aligns learning with entity-slot-level supervision. Slot-aware masked attention, when enabled, serves as an optional packed-evaluation implementation that reduces redundant candidate evaluation while leaving the slot-level supervision signal unchanged. The support values in Panel B are normalized within each candidate pool for visualization only; path selection uses the raw recurrence-based scores.

This property allows KARMA to construct contrastive alternatives whose differences are localized to intermediate entity slots.

3.2.2 Top-K Candidate Selection

Given a candidate pool $\mathcal{P}$ of size M, written $\mathcal{P} = \{p^1, \ldots, p^M\}$, KARMA selects K paths by balancing path support and structural diversity. We measure the support of each path by the cumulative recurrence of its intermediate entities within the pool: each intermediate entity contributes its occurrence count across $\mathcal{P}$, and the path's support is the sum of these contributions. Paths whose intermediate entities frequently recur thus receive higher support, providing a training-free evidence prior from repeated graph traversals within the same source-target pool. This recurrence-based support is not a correctness guarantee: the highest-scoring path need not be uniquely correct or preferable.

KARMA then greedily admits paths in descending order of support, subject to an overlap budget δ that limits the number of intermediate entities shared with previously admitted paths (Fig. 2B). This budget enforces structural diversity by preventing near-duplicate candidates. The procedure stops when K paths have been admitted; pools with fewer than K admissible paths are discarded.

We denote the admitted paths by $p_k^\star$ ($k = 1, \ldots, K$), indexed in admission order, with intermediate entities written as $e_j^{(k)}$ for slot j of the k-th selected path. The ordered selected path sequence

$$\mathcal{P}^\star = (p_1^\star, \ldots, p_K^\star) \tag{2}$$

preserves this ordering through verbalization and provides the preference ordering used by SPA: $p_1^\star$ corresponds to the chosen candidate and $p_{2:K}^\star$ to the rejected candidates. This ordering represents relative evidence strength rather than a binary partition of path correctness: rejected candidates may remain valid at shared slots while differing at entity-slots with weaker support. This distinction motivates SPA's slot-level objective, which contrasts discriminative entity-slots without assigning a uniform negative signal to the shared template. We empirically validate this chosen-rejected partition with a GPT-4 judge study in Appendix E, and report end-to-end ablations against random path selection in Section 4.3.

3.2.3 Template Verbalization

Each selected path $p_k^\star \in \mathcal{P}^\star$ is verbalized into a natural-language sequence y^k by populating a schema-specific template with the entities along the path. Verbalization is fully rule-based and uses *no language model in the generation loop*. For each schema, we predefine multiple template variants that differ in wording and synonym choice and uniformly sample one variant for each candidate pool. The sampled template provides a fixed scaffold of textual fragments and entity slots determined by the relation schema, while the entities

of $p_k^\star$ are realized as multi-token spans following a domain-specific rule.

Within a candidate pool, all selected paths share the same source and target entities and the same relation schema, so they are verbalized under an identical template scaffold. Consequently, the resulting ordered candidate sequence $\mathcal{Y} = (y^1, \ldots, y^K)$ differs only at the intermediate entity slots, while all template fragments and endpoint entities remain shared (Fig. 2C). This slot-localized variation is precisely the structural property SPA exploits in the next section. Holding the sampled variant fixed within each pool preserves slot alignment, while sampling across the variant set introduces surface-form diversity across pools.

Repeating this procedure over all retained source-target pairs yields the training corpus

$$\mathcal{D} = \{(x_i, \mathcal{Y}_i)\}_{i=1}^{N}, \qquad \mathcal{Y}_i = (y_i^1, \ldots, y_i^K), \quad (3)$$

where x_i is a prompt constructed from the source and target entities of the i-th instance, and $\mathcal{Y}_i$ is the ordered verbalized candidate set inheriting the chosen/rejected ordering from the corresponding selected path sequence.

3.3 Slot-Parallel Alignment for Slot-Level Preference Learning

SPA realigns preference optimization from sequence-level to slot-level resolution, matching the granularity at which KARMA candidates actually differ. It concentrates supervision on discriminative entity slots through a decoupled hybrid objective: slot-level preference and anchor losses supervise entity tokens, while a template LM loss preserves the shared scaffold. Slot-parallel packing evaluates variants under shared context, and slot-aware masked attention optionally obtains per-candidate log-likelihoods in one forward pass instead of $K+1$, at a small accuracy cost.

3.3.1 Slot-Parallel Packing with Shared Context

For a single training instance, we drop the corpus index i and write $\mathcal{Y} = (y^1, \ldots, y^K)$. Following the path notation in Section 3.2.2, the candidates in $\mathcal{Y}$ differ only at the intermediate slots $j = 1, \ldots, n-1$, where slot j holds the entity $e_j^{(k)}$ in candidate y^k. Let $t_0, \ldots, t_{n-1}$ be the shared template fragments, and let $e_j^{(1)}$ denote the chosen entity at slot j, which we refer to as the *anchor*.

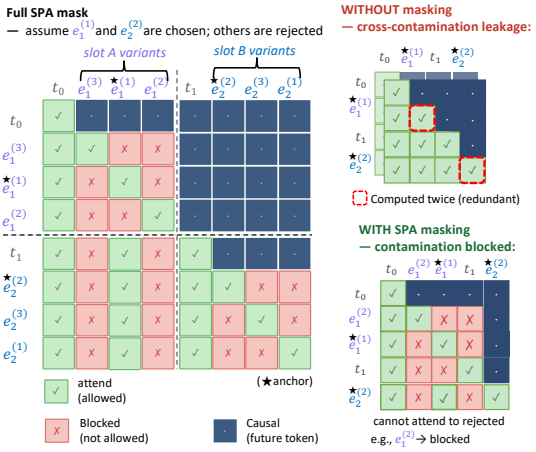


Figure 3: Slot-aware masked attention for packed candidate evaluation. Green cells denote allowed attention, red cells denote blocked attention, and dark cells denote future tokens. Compared with a plain causal mask, the SPA mask lets each variant attend to the shared template scaffold and its own slot state, while restricting previous slots to the anchored prefix so that each candidate's log-likelihood within the packed sequence approximates its standalone value under $K+1$ independent forward passes.

SPA constructs the packed sequence

$$\mathbf{s} = \left[x;\ t_0,\ e_1^{(1:K)},\ t_1,\ \ldots,\ e_{n-1}^{(1:K)},\ t_{n-1}\right], \quad (4)$$

where $e_j^{(1:K)}$ denotes the ordered concatenation of slot-j entity variants $e_j^{(1)}, \ldots, e_j^{(K)}$. As illustrated in Fig. 2E, SPA groups the variants of each intermediate slot into a local slot bundle while sharing the surrounding template fragments. We shuffle the intra-slot variant order during training to prevent positional shortcuts.

3.3.2 Slot-Aware Masked Attention

Since candidates in $\mathbf{s}$ share all template fragments and differ only at the intermediate entity slots, $K+1$ standalone forward passes would redundantly recompute identical template representations. SPA replaces the causal attention mask with a slot-aware mask so that all per-candidate log-likelihoods are obtained from a single forward pass while approximating their standalone values.

We organize the mask by what each query token is allowed to attend to within the causal prefix. Template tokens at fragment t_j attend only to the preceding template fragments and to the *anchor* entity of each past slot. An entity token $e_j^{(k)}$ at slot j in candidate k attends to the preceding template fragments, to the past anchors $e_1^{(1)}, \ldots, e_{j-1}^{(1)}$,

and to itself. Consequently, each candidate's log-likelihood within the packed sequence is computed along its own anchored path, approximating the value it would have under an independent forward pass.

Fig. 3 (left) illustrates the resulting mask for a packed sequence with two intermediate slots and three variants per slot. With a plain causal mask (top right), later tokens can attend to multiple variants from previous slots, so a candidate's log-likelihood within the packed sequence diverges from its standalone value. The SPA mask (bottom right) restricts each candidate to its own anchored path while keeping the template representation shared, so one forward pass over $\mathbf{s}$ approximates $K+1$ independent passes while encoding the template scaffold only once.

3.3.3 Decoupled Hybrid Objective

Given per-candidate log-likelihoods obtained from the packed sequence, applying a sequence-level preference objective to $\mathbf{s}$ would still spread supervision over many template tokens that are identical across candidates. Since KARMA candidates differ only at intermediate entity slots, SPA decouples the objective according to this structure: a slot-level preference loss supervises entity choices, an anchor loss stabilizes the chosen entities, and a template language modeling loss preserves the shared scaffold (Fig. 2F).

Slot-level preference. For each slot j, the anchor entity $e_j^{(1)}$ should be preferred to the rejected entities $e_j^{(k)}$ from candidates $k = 2, \ldots, K$. Let π_θ denote the policy model and π_{ref} a fixed reference model. For an entity span $e_j^{(k)}$, we write $\ell_\theta(e_j^{(k)} \mid \mathbf{s})$ and $\ell_{\text{ref}}(e_j^{(k)} \mid \mathbf{s})$ for its length-normalized log-likelihood under π_θ and π_{ref}, computed under the slot-aware mask. We define the reference-normalized score

$$r_\theta(e_j^{(k)}) = \ell_\theta(e_j^{(k)} \mid \mathbf{s}) - \ell_{\text{ref}}(e_j^{(k)} \mid \mathbf{s}). \quad (5)$$

This score is the policy's log-likelihood gain over the reference on the entity span $e_j^{(k)}$. SPA applies a squared margin loss between the anchor and each rejected entity at the same slot:

$$\mathcal{L}_{\text{pref}} = \frac{1}{(n-1)(K-1)} \times \sum_{j=1}^{n-1} \sum_{k=2}^{K} \left(r_\theta(e_j^{(1)}) - r_\theta(e_j^{(k)}) - \frac{1}{2\beta} \right)^2, \quad (6)$$

where β controls the target margin. Here, k follows the original top-K candidate order: $k = 1$ is the chosen candidate and $k = 2, \ldots, K$ are rejected candidates. This term provides the discriminative signal at the slots where candidates differ, rather than spreading preference supervision over the shared template.

Anchor confidence. The preference term in Eq. 6 constrains relative scores, but it does not directly ensure that the chosen entity has high likelihood. SPA therefore adds an anchor confidence term:

$$\mathcal{L}_{\text{anchor}} = -\frac{1}{n-1} \sum_{j=1}^{n-1} \ell_\theta(e_j^{(1)} \mid \mathbf{s}). \quad (7)$$

This term stabilizes the likelihood of the selected path entities, complementing the relative comparison imposed by $\mathcal{L}_{\text{pref}}$.

Template language modeling. The template fragments encode the relation scaffold shared by all candidates, but they are not discriminative for the slot-level preference comparison. Let $\mathcal{T}$ denote the set of shared template token positions in $\mathbf{s}$, and let s_u denote the u-th token of $\mathbf{s}$. SPA preserves the scaffold with a standard causal language modeling loss:

$$\mathcal{L}_{\text{template}} = -\frac{1}{|\mathcal{T}|} \sum_{u \in \mathcal{T}} \log \pi_\theta(s_u \mid s_{<u}). \quad (8)$$

This term maintains the fluency and logical structure of the verbalization while leaving entity discrimination to the slot-level losses.

Full objective. The final SPA objective combines the slot-level preference term with the two auxiliary losses:

$$\mathcal{L}_{\text{SPA}} = \mathcal{L}_{\text{pref}} + \lambda_{\text{anchor}} \mathcal{L}_{\text{anchor}} + \lambda_{\text{template}} \mathcal{L}_{\text{template}}, \quad (9)$$

where λ_{anchor} and $\lambda_{\text{template}}$ are scalar loss weights. We average $\mathcal{L}_{\text{SPA}}$ over training instances.

4 Experiments

We evaluate KARMA across three knowledge-intensive domains to answer three research questions: (**RQ1**) Does the co-design of KARMA-synthesized contrastive candidates and SPA's slot-level objective outperform both conventional SFT and sequence-level preference optimization on the same data? (**RQ2**) How important is graph-guided

Domain	Metric	Baseline (Base LLM)	Simple SFT		Ours (KARMA)
			(1) Open-Data	(2) KARMA-Data	
Biomedical	MedQA (Acc)	0.362	0.434	0.273	**0.541**
	PubMedQA (Acc)	0.511	0.512	0.229	**0.522**
	MMLU-Bio (Acc)	0.505	0.571	0.381	**0.702**
	MMLU-Pro-Bio (Acc)	0.230	0.324	0.292	**0.532**
	GPQA-Bio (Acc)	**0.700**	0.533	0.333	0.667
Computer Science	MMLU-CS (Acc)	0.651	0.655	0.451	**0.701**
	MMLU-Pro-CS (Acc)	0.195	0.243	0.180	**0.256**
	CSBench (Acc)	0.345	**0.506**	0.372	0.495
Chemistry	MMLU-Chem (Acc)	0.500	0.495	0.373	**0.630**
	MMLU-Pro-Chem (Acc)	0.148	0.178	0.126	**0.195**
	GPQA-Chem (Acc)	0.161	0.172	0.172	**0.204**

Table 1: Main performance comparison between the baseline, simple SFT variants, and our proposed method (KARMA). Results are reported in accuracy (**Acc**). For KARMA, the optimal configuration is reported in Appendix. D. The best and second-best results per row are highlighted.

path selection for synthetic reasoning supervision? (**RQ3**) Does aligning the preference objective's resolution with the entity-slot structure of the data yield consistent gains over sequence-level and token-level alternatives?

4.1 Experimental Setup

We evaluate on three domains: *Biomedical*, *CS*, and *Chemistry*, using 11 benchmarks that mix MMLU subsets (Hendrycks et al., 2021), MMLU-Pro subsets (Wang et al., 2024c), and reasoning-heavy benchmarks (MedQA (Jin et al., 2021), PubMedQA (Jin et al., 2019), GPQA (Rein et al., 2024), CSBench (Song et al., 2025)). The base model is Qwen2.5-7B-Instruct with LoRA fine-tuning, and all evaluations use zero-shot chain-of-thought prompting. Full benchmark lists (C.1), dataset sources (A), and implementation details (B) are provided in Appendix. The SPA loss weights are selected on the Biomedical domain and transferred unchanged to CS and Chemistry without target-benchmark tuning; full training and selection protocols are provided in Appendices B.2 and D.

4.2 Main Results

As shown in Tab. 1, KARMA yields consistent gains over the baseline across the majority of benchmarks, with the largest improvements on reasoning-heavy ones. KARMA also outperforms Open-Data SFT on the majority of benchmarks, indicating that the gains do not merely reflect additional in-domain text but the structural properties of KARMA-synthesized supervision.

The contrast between KARMA-Data SFT and KARMA further isolates the role of the training objective. KARMA-Data SFT uses identical KARMA-synthesized chosen candidates but optimizes them with a flat sequence-level SFT objective, and fails to consistently improve over the baseline. This reflects an inherent incompatibility between KARMA's data structure and sequence-level supervision: chosen candidates from the same source-target pair share almost all tokens except at the intermediate entity-slots, so SFT on these candidates assigns equal weight to all positions and the model is dominantly supervised on the shared template tokens, which form a small set of patterns repeated across many examples. This concentrates SFT updates on a low-diversity response distribution, encouraging reliance on repeated output formats and increasing the risk of forgetting knowledge acquired during pretraining (Yang et al., 2025b; Zhang et al., 2025; Béthune et al., 2025). We view this as a data-side analog of the Resolution Mismatch Problem, where the mismatch arises not between preference signal and optimization granularity, but between data structure and supervision granularity. SPA resolves this by decoupling slot-level preference supervision from template language modeling, converting the same data into the strongest signal in the comparison. This confirms that the gains of KARMA arise from the co-design of structurally aligned data and slot-level supervision, not from data scale or content alone. This answers RQ1: KG-grounded contrastive synthesis improves domain reasoning beyond conventional SFT, but only when the training objective operates at the entity-slot-level resolution at which the candidates actually differ.

4.3 Ablation Study

Ablation of Graph-Guided Path Selection. Tab. 1 shows that the full KARMA framework outperforms baselines, but does not isolate which

Metric	Random	KARMA	Δ
MedQA (Acc)	0.381	**0.541**	+0.160
PubMedQA (Acc)	0.465	**0.522**	+0.057
MMLU-Bio (Acc)	0.555	**0.702**	+0.147
MMLU-Pro-Bio (Acc)	0.338	**0.532**	+0.194
GPQA-Bio (Acc)	0.400	**0.667**	+0.267
Average	0.428	**0.593**	+0.165

Table 2: Ablation on path selection strategy in the biomedical domain. **Random** samples paths from the KG; **KARMA** uses our support-based selection.

component drives the gain. RQ2 asks whether KARMA's support-based path selection itself contributes, or whether any sampling from the same KG would suffice. We replace support-based selection with uniform random sampling, holding all other components fixed (KG, source-target pairs, schema, template, SPA objective, hyperparameters). Tab. 2 shows that KARMA outperforms random sampling on every Biomedical benchmark, with an average gain of +16.5 points.

The gain pattern is itself informative. Improvements are largest on benchmarks that require chaining multiple facts, such as GPQA-Bio (+26.7), MMLU-Pro-Bio (+19.4), and MedQA (+16.0), and remain substantial on knowledge-heavy multiple-choice reasoning (MMLU-Bio +14.7), while PubMedQA, which often admits surface-level factual lookup, shows only a marginal gain. This selectivity is consistent with *the view of KG paths as a structural analogue of reasoning chains*: support-based selection acts as a simple evidence prior, where intermediate entities that recur across the pool reflect connections corroborated by multiple traversals in the graph, yielding chosen-rejected contrasts that are informative for reasoning rather than mere surface distractors. That such a lightweight, training-free heuristic, relying only on entity recurrence statistics already implicit in the KG, suffices to produce these gains answers RQ2: KARMA's advantage comes from *how* paths are selected, and the KG itself already carries much of the structure needed to make that selection effective. We further assess the recurrence-based support prior through an LLM-judge study and analyze its disagreement cases in Appendix E.

Ablation on Preference Objective Resolution. Tab. 3 compares SPA with representative preference optimization baselines operating at different resolutions. The goal is to isolate the effect of preference objective resolution: whether the preference signal is applied at the sequence, token, or slot level.

Domain	Benchmark	DPO (Rafailov et al., 2023)	MDPO	TDPO (Zeng et al., 2024)	KARMA
Computer Science	MMLU-CS	0.651	0.683	0.689	**0.701**
	MMLUPro-CS	0.197	0.249	0.256	**0.256**
	CSBench	0.356	0.446	0.475	**0.495**
Chemistry	MMLU-Chem	0.607	0.570	0.616	**0.630**
	MMLUPro-Chem	0.188	0.189	0.194	**0.195**
	GPQA-Chem	0.172	0.172	0.182	**0.204**

Table 3: Ablation comparing preference optimization objectives at different resolutions. SPA achieves the best or tied-best performance across benchmarks.

All methods are trained on the same KARMA-synthesized candidates and share the same training pipeline and hyperparameters; only the preference objective differs.

We consider three baselines. DPO ($K = 1$) performs preference optimization using pairwise chosen-rejected comparisons. MDPO ($K = 4$) extends this formulation to multiple rejected candidates while still operating at the sequence level. TDPO ($K = 1$) introduces token-level KL regularization, providing a finer-grained signal than sequence-level objectives but without explicitly targeting the entity-slots where KARMA ($K = 4$) candidates differ. In contrast, SPA applies supervision directly at the slot level, aligning the optimization objective with the structural units of KARMA-synthesized data.

The results show that SPA achieves the best or tied-best performance across all benchmarks, improving over the strongest baseline (TDPO) on reasoning-heavy ones such as CSBench (+2.0) and GPQA-Chem (+2.2). Sequence-level objectives assign supervision to the entire candidate, token-level regularization makes the signal more fine-grained, and SPA further aligns it with the entity-slot structure where the actual candidate differences arise. This answers RQ3: aligning the preference objective's resolution with the data's entity-slot structure yields consistent gains over both sequence-level (DPO, MDPO) and token-level (TDPO) alternatives, indicating that resolution alignment, beyond granularity alone, accounts for the improvement.

Ablation of Slot-Aware Masked Attention. Tab. 4 characterizes the accuracy-efficiency trade-off of slot-aware masked attention, which we use as the default packed-evaluation implementation in SPA. The default w/ Mask variant packs all candidates into one sequence and approximates their log-likelihoods in a single forward pass, whereas w/o Mask evaluates candidates independently with $K+1$ forward passes per step. Both variants share identical objectives, data, and hyperparameters;

Domain	Benchmark	w/o Mask	w/ Mask
CS	MMLU-CS	0.711	**0.701**
	MMLUPro-CS	0.258	**0.256**
	CSBench	0.495	**0.495**
Chemistry	MMLU-Chem	0.640	**0.630**
	MMLUPro-Chem	0.198	**0.195**
	GPQA-Chem	0.182	**0.204**
Efficiency (per training step)			
	Step time (s)	15.58	**11.32**
	Peak memory (GiB)	**32.83**	55.39
	Reserved memory (GiB)	61.96	**59.02**
	Forward passes	$2(K+1)$	**2**

Table 4: Ablation of SPA's slot-aware masked attention on task performance and per-step training efficiency. Bold marks the w/ Mask column.

only the attention masking and packing differ. Efficiency is measured on Qwen2.5-7B-Instruct with $K=4$ rejected candidates.

The two variants exhibit a clear trade-off between accuracy and training cost. On the accuracy side, disabling the mask matches or slightly improves performance on five of the six benchmarks, while the w/ Mask configuration remains within one accuracy point of the independent-evaluation variant on those benchmarks and outperforms it on GPQA-Chem. On the efficiency side, the default packed evaluation under the slot-aware mask reduces the number of forward passes per step from $2(K+1)$ to 2, yielding a $1.38\times$ wall-clock speedup. The speedup comes with a substantial increase in peak GPU memory from 32.83 to 55.39 GiB, corresponding to a $1.69\times$ increase, because the packed sequence concentrates all candidates into a single forward pass and materializes the slot-aware mask; reserved memory remains comparable. Thus, slot-aware masked attention is an optional time-memory trade-off rather than the source of SPA's slot-level supervision. Enabling the mask favors wall-clock time, whereas disabling it reduces peak memory and may slightly improve accuracy; the preference, anchor, and template losses remain unchanged in both settings.

5 Conclusion

We propose KARMA, a KG-based pipeline that synthesizes structurally aligned contrastive candidates with explicit entity-slot-level differences. By leveraging KG paths, KARMA provides broad factual coverage while preserving structural alignment across candidates. We further introduce SPA, a slot-level preference optimization method designed for KARMA-style data. SPA addresses the Resolution Mismatch Problem between sequence-level supervision and entity-slot-level candidate variation, while differing from token-level methods by directly targeting the construction-exposed entity-slots where candidates differ. Together, KARMA achieves substantial gains on most knowledge-intensive reasoning benchmarks, while our analyses highlight the complementary roles of support-based KG path selection and entity-slot-level supervision.

6 Limitations

KARMA shares two limitations with prior KG-grounded LLM research (Tan et al., 2025; Sun et al., 2024; Luo et al., 2024; Wang et al., 2024a). First, the synthesis pipeline relies on a domain knowledge graph with sufficient coverage and connectivity to support multi-hop path sampling, so domains without such a KG would require automatic KG construction or hybrid retrieval-based grounding. Second, KARMA assumes that the target capability is expressible through an explicit schema and verbalization template, which restricts the framework to capabilities decomposable into discrete entity-slot structures and leaves open-ended generation outside its current scope.

Beyond these, slot-aware masked attention is an efficiency-oriented implementation choice rather than the source of SPA's supervision signal: the default masked setting reduces per-step forward passes from $2(K+1)$ to 2 and yields a $1.38\times$ wall-clock speedup, while disabling it recovers independent candidate evaluation at higher training cost. Users can choose either setting based on whether wall-clock time or peak accuracy is the binding constraint. Finally, all experiments use Qwen2.5-7B-Instruct with LoRA fine-tuning, and generality across other model families and scales remains to be verified.

Ethical considerations

Potential risks. KARMA-synthesized data is grounded in domain knowledge graphs and verbalized through fixed templates. Although KG paths provide structural support, the synthesized candidates may contain factual inaccuracies inherited from the source graphs or template phrasing. Models fine-tuned with KARMA should not be deployed for clinical, medical, or otherwise high-stakes decision-making without further validation.

Data and license. The domain knowledge graphs (biomedical, computer science, and chemistry) are

sourced from GRBench (Jin et al., 2024), which is publicly available for research use. Qwen2.5-7B-Instruct is released under the Apache 2.0 license. All evaluation benchmarks (MMLU, MMLU-Pro, MedQA, PubMedQA, GPQA, CSBench) are used under their original licenses for research purposes only. The KARMA-synthesized dataset and code will be released under a research-only license upon acceptance, consistent with the upstream data licenses.

Privacy and offensive content. The source knowledge graphs contain public scientific entities and do not include personally identifiable information about individuals. We manually inspected samples of synthesized candidates and did not identify offensive content.

Computing infrastructure. All experiments were conducted on NVIDIA A100 GPU.

Use of AI assistants. We used AI assistants (e.g., ChatGPT, Claude, GitHub Copilot) for writing polish, LaTeX formatting, and boilerplate code generation and debugging. All research ideas, experimental design, analysis, core implementation, and final text are the authors' own work.

Acknowledgments

This research was supported by the 14th Executive Programme for Scientific and Technological Cooperation between the Republic of Italy and the Republic of Korea (2026–2028) grant funded by the National Research Foundation of Korea (NRF) (RS-2025-25460384), and by the Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency (KOCCA) grant funded by the Ministry of Culture, Sports and Tourism (MCST) (RS-2024-00442006). This research was also supported by the AI Seoul Tech Research Support Program of the Seoul Future Foundation.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565.

Louis Béthune, David Grangier, Dan Busbridge, Eleonora Gualdoni, Marco Cuturi, and Pierre Ablin. 2025. Scaling laws for forgetting during finetuning with pretraining data injection. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4020–4042. PMLR.

Jae Hyeon Cho, JunHyeok Oh, Myunsoo Kim, and Byung-Jun Lee. 2025. Rethinking dpo: The role of rejected responses in preference misalignment. In *EMNLP (Findings)*, pages 8159–8176.

Fenia Christopoulou, Ronald Cardenas, Gerasimos Lampouras, Haitham Bou Ammar, and Jun Wang. 2025. SparsePO: Controlling preference alignment of LLMs via sparse token masks. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25477–25503, Suzhou, China. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th international conference on natural language generation*, pages 124–133.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods*

in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pages 2567–2577.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2025. Critical tokens matter: Token-level contrastive estimation enhances LLM's reasoning capability. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37906–37918. PMLR.

Aiwei Liu, Haoping Bai, Zhiyun Lu, Yanchao Sun, Xiang Kong, Xiaoming Wang, Jiulong Shan, Albin Madappally Jose, Xiaojiang Liu, Lijie Wen, and 1 others. 2025a. Tis-dpo: Token-level importance sampling for direct preference optimization with estimated weights. In *International Conference on Learning Representations*, volume 2025, pages 51339–51368.

Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, and 1 others. 2025b. Lipo: Listwise preference optimization through learning-to-rank. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2404–2420.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and 1 others. 2025. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *International Conference on Learning Representations*, volume 2025, pages 49573–49609.

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, volume 2024, pages 14400–14423.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Haris Riaz, Sourav Sanjukta Bhabesh, Vinayak Arannil, Miguel Ballesteros, and Graham Horwood. 2025. Metasynth: Meta-prompting-driven agentic scaffolds for diverse synthetic data generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18770–18803.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

Xiaoshuai Song, Muxi Diao, Guanting Dong, Zhengyang Wang, Yujia Fu, Runqi Qiao, Zhexu Wang, Dayuan Fu, Huangxuan Wu, Bin Liang, and 1 others. 2025. Cs-bench: A comprehensive benchmark for large language models towards computer science mastery. In *International Conference on Learning Representations*, volume 2025, pages 55244–55288.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pages 3868–3898.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-graph: Knowledge graph empowered large language model reasoning. In *Proceedings of the ACM on Web Conference 2025*, pages 3505–3522.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, and 1 others. 2024a. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7813–7835.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language

models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024c. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *International Conference on Learning Representations*, volume 2024, pages 30745–30766.

Kailai Yang, Zhiwei Liu, Qianqian Xie, Jimin Huang, Erxue Min, and Sophia Ananiadou. 2025a. Selective preference optimization via token-level reward function estimation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7032–7056.

Linyao Yang, Jian-Tao Huang, Yafei Lu, Zhenhui Jessie Li, and Guirong Xue. 2025b. The emperor's new reasoning: format imitation overshadows genuine mathematical understanding in sft. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21098–21111.

Ning Yang, Hai Lin, Yibo Liu, Baoliang Tian, Guoqing Liu, and Haijun Zhang. 2026. Token-importance guided direct preference optimization. In *International Conference on Learning Representations*, volume 2026, pages 13888–13910.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58348–58365. PMLR.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025. The best instruction-tuning data are those that fit. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 141172–141208. Curran Associates, Inc.

Wenxuan Zhou, Shujian Zhang, Lingxiao Zhao, and Tao Meng. 2025. T-reg: Preference optimization with token-level reward regularization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27876–27889.

A Dataset Construction Details

A.1 Knowledge Graph Sources

KARMA constructs training data from domain-specific knowledge graphs from GRBench (Jin et al., 2024), summarized in Table 5. Each graph is represented as a typed node-edge structure, where nodes contain textual attributes and outgoing typed neighbors. The biomedical graph uses heterogeneous node types such as diseases, symptoms, compounds, genes, anatomy, side effects, pharmacologic classes, biological processes, molecular functions, cellular components, and pathways. The academic graph uses paper nodes with citation edges, where each paper contains title, abstract, and keyword fields.

Domain	Graph nodes	Edge types	Node attributes
Biomedical	Typed biomedical entities	Typed biomedical relations	Name
Computer Science	Papers	`reference`, `cited_by`	Title, abstract, keywords
Chemistry	Papers	`reference`, `cited_by`	Title, abstract, keywords

Table 5: Summary of domain knowledge graphs used for KARMA synthesis.

A.2 Relation Schemas

A relation schema defines the edge-type pattern used to enumerate paths from a source entity to a target entity. The schema set is constructed by exhaustively enumerating a constrained space of typed multi-hop metapaths, without manually designing or selecting a schema for each source-target pair. In the academic graph, schemas are metapaths over citation edges. We use four-hop schemas by default, generated from all combinations of `reference` and `cited_by`. This produces 16 relation schemas. In the biomedical graph, schemas are typed metapaths over biomedical node and relation types. Exhaustive enumeration under these typed-edge constraints produces 522 metapaths.

A.3 Path Enumeration and Filtering

For each schema, KARMA enumerates paths that follow the prescribed edge-type sequence, as described in Algorithm 1. Each path connects a source entity e_s to a target entity e_t through a sequence of intermediate entities. For a fixed source-target pair, all paths satisfying the same schema form a candidate pool. Candidate pools with fewer than K valid paths are discarded. We remove loops in intermediate positions and apply intermediate caps during traversal to avoid path explosion. For each source-target pair, KARMA considers every applicable schema rather than selecting a single schema as optimal for that pair. A pair may therefore produce multiple candidate pools under different schemas, with quality controlled only after enumeration through pool filtering and support-based Top-K selection.

Algorithm 1 Schema-Constrained Path Pool Construction

Require: Knowledge graph G, schema set $\mathcal{S}$, candidate number K
Ensure: Candidate pools $\mathcal{C}$

$\mathcal{C} \leftarrow \emptyset$
for each schema $s \in \mathcal{S}$ **do**
  for each source entity e_s **do**
    Traverse G according to schema s
    Collect reachable target entities e_t
    for each target entity e_t **do**
      Enumerate all middle paths $\mathcal{M}(e_s, e_t)$
      if $|\mathcal{M}(e_s, e_t)| \geq K$ **then**
        Add $(e_s, e_t, s, \mathcal{M}(e_s, e_t))$ to $\mathcal{C}$
      end if
    end for
  end for
end for
return $\mathcal{C}$

A.4 Support-based Top-K Selection

For each source-target candidate pool, we select K paths using an entity-support score and an overlap constraint, as detailed in Algorithm 2. The support score of a middle path m is the sum of the occurrence counts of its intermediate entities within the same source-target pool:

$$\text{score}(m) = \sum_{e \in m} \text{freq}(e). \tag{10}$$

Here, $\text{freq}(e)$ is the number of paths in the same candidate pool that contain intermediate entity e. The resulting score is a raw, unbounded integer sum. The support values shown in Fig. 2B are normalized within each candidate pool for visualization only; path ranking and selection use the raw scores.

We then greedily select high-score paths while allowing at most δ overlapping intermediate entities with the already selected paths. In the academic-domain generator, we use $K = 4$ and $\delta = 1$. Recurrence-based support serves as a training-free evidence prior rather than a correctness guarantee. An intermediate entity that recurs across multiple

Algorithm 2 Support-based Top-K Path Selection

Require: Middle paths $\mathcal{M}(e_s, e_t)$, candidate number K, overlap budget δ
Ensure: Selected paths $\mathcal{M}^\star$

if $|\mathcal{M}(e_s, e_t)| < K$ **then**
  return $\emptyset$
end if
Count entity frequencies $c(e)$ over all intermediate entities in $\mathcal{M}(e_s, e_t)$
Score each path by $s(m) = \sum_{e \in m} c(e)$
Sort paths in descending order of $s(m)$
$\mathcal{M}^\star \leftarrow \emptyset, \quad U \leftarrow \emptyset$
for each path m in sorted order **do**
  $E_m \leftarrow \{e : e \in m\}$
  if $|E_m \cap U| \leq \delta$ **then**
    Add m to $\mathcal{M}^\star$
    $U \leftarrow U \cup E_m$
  end if
  if $|\mathcal{M}^\star| = K$ **then**
    break
  end if
end for
if $|\mathcal{M}^\star| < K$ **then**
  return $\emptyset$
end if
return $\mathcal{M}^\star$

paths receives support from multiple graph traversals within the same pool, but the highest-scoring path need not be uniquely correct or preferable. Accordingly, the selected ordering represents relative evidence strength among structurally aligned candidates.

A.5 Template Verbalization

Each selected path is verbalized into a natural-language candidate using a schema-specific template. Verbalization is fully rule-based and uses no language model in the generation loop. For each schema, we predefine multiple template variants that differ in wording and synonym choice and uniformly sample one variant for each candidate pool. The sampled variant is held fixed within the pool, preserving slot alignment, while sampling across variants introduces surface-form diversity across pools. Entity mentions are marked with bold delimiters, e.g., `**entity**`. These marked spans define the entity slots used by SPA. The selected paths for the same source-target pair share the same template scaffold and differ only in their intermediate entity slots.

A.6 Dataset Statistics

Table 6 reports dataset statistics. For each domain, we report the number of nodes and edges in the source knowledge graph, the number of relation schemas used for path enumeration, the number of generated training samples, and the number of candidates per pool size K.

Domain	#Nodes	#Edges	#Schemas	#Samples	K
Biomedical	47K	4.23M	522	1M	4
Computer Science	5.26M	65M	16	1M	4
Chemistry	1.85M	41M	16	1M	4

Table 6: Dataset statistics for KARMA-synthesized training data.

B Training Details

B.1 Model and LoRA Configuration

All experiments use Qwen2.5-7B-Instruct as the base model. We fine-tune the model with LoRA adapters. The LoRA rank is 32 and the LoRA alpha is 32. Adapters are inserted into the attention and feed-forward projection modules. The full configuration is reported in Table 7. The reference model π_{ref} used by SPA and all DPO-family baselines is obtained by disabling the LoRA adapter on the policy model π_θ; that is, π_θ and π_{ref} share identical base parameters and differ only in whether the trained adapters are active.

B.2 Training Protocol

All training runs share the same general-purpose hyperparameters (learning rate, optimizer, batch size, epochs, gradient clipping, LoRA configuration) across baselines and KARMA, so that performance differences reflect the training objective rather than tuning. Method-specific hyperparameters, such as λ_{anchor}, $\lambda_{\text{template}}$, and β for SPA, are set as in Table 8 for KARMA, and the corresponding original defaults are used for DPO, MDPO, and TDPO. We do not perform validation-based checkpoint selection; the final epoch checkpoint is used for evaluation. Each domain dataset is used for training only, and evaluation is performed on the held-out benchmarks listed in Table 11. The SPA loss weights are selected using the Biomedical-domain sensitivity analysis and then applied unchanged to Computer Science and Chemistry. We do not tune these weights separately for the target benchmarks in either transferred domain.

All results reported in this paper, including the main results in Table 1 and all ablations, are com-

Configuration	Value
Base model	Qwen2.5-7B-Instruct
Fine-tuning method	LoRA
LoRA rank	32
LoRA alpha	32
Target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj` `gate_proj`, `up_proj`, `down_proj`
Precision	bfloat16
Max sequence length	4096
Gradient checkpointing	Enabled
Attention implementation	SDPA
Reference model π_{ref}	Policy model with LoRA adapter disabled

Table 7: Model and LoRA configuration.

Hyperparameter	Value
Learning rate	5×10^{-5}
Epochs	3
Per-device batch size	8
Gradient accumulation	1
Max gradient norm	1.0
λ_{anchor}	0.5
$\lambda_{\text{template}}$	0.1
Entity margin parameter β	0.1
Variant shuffling	Enabled

Table 8: SPA training hyperparameters. The preference-loss weight λ_{pref} is fixed to 1.0 in all experiments based on the sensitivity analysis in Table 14, and is therefore absorbed into $\mathcal{L}_{\text{pref}}$ in Eq. 9.

puted as the mean of 5 independent runs with different random seeds. The standard deviation is below 0.01 in every reported configuration and is provided alongside each mean in Tables 1, 2, 3, and 4.

B.3 Compute Infrastructure

All experiments are run on NVIDIA A100 GPU. A single SPA training run for one domain (1M samples, 3 epochs) takes approximately `40` GPU-hours on `1` GPU. The total compute budget across 5 random seeds, three domains, all baseline methods, and ablation studies is on the order of a few thousand A100 GPU-hours.

B.4 Baseline Implementations

We compare KARMA with the base model, open-data SFT, KARMA-data SFT, and preference-optimization baselines, as listed in Table 9. All fine-tuned baselines share the same base model, LoRA configuration, learning rate, batch size, epochs, and optimizer as KARMA; they differ only in the training objective and, where applicable, in the structure of the input data. Each baseline retains its original method-specific hyperparameters (e.g., the KL coefficient in DPO, the token-level regularization weight in TDPO) at the values recommended in their respective papers.

Baseline	Description
Base LLM	Qwen2.5-7B-Instruct without task-specific fine-tuning
Open-Data SFT	SFT on open-domain or domain instruction data
KARMA-Data SFT	SFT on KARMA-generated chosen responses
DPO	Pairwise sequence-level preference optimization
MDPO	Multi-rejected sequence-level preference optimization
TDPO	Token-level preference optimization baseline
KARMA / SPA	Slot-level preference optimization on KARMA candidates

Table 9: Baseline implementation summary. All baselines share the same base model, LoRA configuration, and general-purpose training hyperparameters.

B.5 Slot-aware Mask Implementation

SPA packs candidate variants into a single sequence by replacing each entity slot with a local bundle of chosen and rejected variants. The attention mask starts from a causal mask and then blocks cross-candidate contamination according to Table 10. PAD separators inserted between variants are blocked as both queries and keys.

Query token	Allowed keys	Blocked keys
Template token	Previous templates + previous anchors	Rejected variants
Anchor entity token	Previous templates + previous anchors + itself	Other variants in same slot
Rejected entity token	Previous templates + previous anchors + itself	Other variants in same slot
PAD separator	None	All tokens
Future token	None	Causal future positions

Table 10: Slot-aware attention mask rules used in SPA.

B.6 SPA Loss Computation

The full SPA loss computation procedure is summarized in Algorithm 3. Given a packed sequence with slot-aware masking, the policy and reference models compute log-probabilities for each variant. Slot-level preference and anchor confidence losses are accumulated across slots, and a template language modeling loss is computed on non-entity tokens.

C Evaluation Details

C.1 Benchmark Sources

We evaluate on biomedical, chemistry, and computer-science benchmarks. The benchmark groups are summarized in Table 11.

C.2 Prompt Templates

All benchmarks are evaluated using the same model wrapper and benchmark-specific prompt templates, summarized by category in Table 12.

Algorithm 3 SPA Loss Computation

Require: Packed sequence s, slot variants $\{e_j^{(k)}\}$, policy model π_θ, reference model π_{ref}
Ensure: SPA loss $\mathcal{L}_{\text{SPA}}$

```
Compute policy log-probabilities under the slot-aware mask
Compute reference log-probabilities with the frozen reference model
for each slot j do
    Compute reference-normalized score r_θ(e_j^(k)) for each variant
    Compare anchor e_j^(1) against rejected variants e_j^(k), k > 1
    Accumulate slot-level preference loss L_pref
    Accumulate anchor confidence loss L_anchor
end for
Compute template language modeling loss L_template on non-entity tokens
L_SPA ← L_pref + λ_anchor L_anchor + λ_template L_template
return L_SPA
```

Domain	Benchmark	Subset / category
Biomedical	MMLU-Bio	Anatomy, biology, medicine, genetics, nutrition, virology
	MMLU-Pro-Bio	Biology
	MedQA	Full benchmark
	PubMedQA	Full benchmark
	GPQA-Bio	Biology
Chemistry	MMLU-Chem	College chemistry, high school chemistry
	MMLU-Pro-Chem	Chemistry
	GPQA-Chem	Chemistry
Computer Science	MMLU-CS	CS, computer security, high school CS, machine learning
	MMLU-Pro-CS	Computer Science
	CSBench	Reasoning task type

Table 11: Benchmark groups used for domain evaluation.

C.3 Answer Extraction

Generated responses are normalized to benchmark-specific answer formats before scoring. For multiple-choice tasks, the predicted option is extracted from the generated text and compared against the gold option. For label-based QA tasks such as yes/no/maybe classification, the generated text is mapped to the corresponding normalized label.

C.4 Inference Configuration

The inference configuration used across all benchmarks is reported in Table 13.

Benchmark type	Prompt format
Multiple-choice QA	Question + answer choices + instruction to select one option
Yes/no/maybe QA	Question + instruction to output the normalized label
Reasoning QA	Question + optional chain-of-thought style instruction

Table 12: Prompt template categories used in evaluation.

Configuration	Value
Base model	Qwen2.5-7B-Instruct
Adapter handling	LoRA adapter loading with optional merge
Batch size	8 by default
Max new tokens	512
Number of shots	0 by default
Distributed evaluation	Supported with `torchrun`
Output format	JSON result file

Table 13: Inference configuration used for evaluation.

D Hyperparameter Sensitivity

Table 14 reports sensitivity to the loss weights λ_{pref}, λ_{anchor}, and $\lambda_{\text{template}}$ on the Biomedical domain. The analysis examines whether SPA is robust to the relative weighting of slot-level preference learning, anchor confidence, and template language modeling. Based on this analysis, we fix $\lambda_{\text{pref}} = 1.0$ in all main experiments and absorb it into $\mathcal{L}_{\text{pref}}$ in the final objective (Eq. 9), tuning only λ_{anchor} and $\lambda_{\text{template}}$. The selected configuration is $(\lambda_{\text{pref}}, \lambda_{\text{anchor}}, \lambda_{\text{template}}) = (1.0, 0.5, 0.1)$, which achieves the best average accuracy across MedQA, PubMedQA, and MMLU-Pro-Bio.

λ_{pref}	λ_{anchor}	$\lambda_{\text{template}}$	MedQA	PubMedQA	MMLU-Pro-Bio	Avg
1.0	1.0	1.0	0.494	0.235	0.169	0.299
1.0	1.0	0.5	0.541	0.287	0.242	0.357
1.0	1.0	0.1	0.541	0.522	0.362	0.475
1.0	0.5	1.0	0.447	0.313	0.362	0.374
1.0	0.5	0.5	0.588	0.261	0.193	0.347
1.0	0.5	0.1	**0.541**	**0.522**	**0.532**	**0.532**
1.0	0.1	1.0	0.517	0.444	0.169	0.377
1.0	0.1	0.5	0.565	0.261	0.556	0.460
1.0	0.1	0.1	0.565	0.235	0.507	0.436
0.5	1.0	1.0	0.494	0.365	0.193	0.351
0.5	1.0	0.5	0.306	0.339	0.217	0.288
0.5	1.0	0.1	0.470	0.391	0.266	0.376
0.5	0.5	1.0	0.494	0.444	0.338	0.425
0.5	0.5	0.5	0.612	0.339	0.242	0.397
0.5	0.5	0.1	0.659	0.365	0.411	0.478
0.5	0.1	1.0	0.541	0.313	0.604	0.486
0.5	0.1	0.5	0.588	0.183	0.580	0.450
0.5	0.1	0.1	0.588	0.496	0.507	0.530
0.1	1.0	1.0	0.565	0.209	0.290	0.354
0.1	1.0	0.5	0.447	0.287	0.314	0.349
0.1	1.0	0.1	0.329	0.339	0.121	0.263
0.1	0.5	1.0	0.494	0.104	0.145	0.248
0.1	0.5	0.5	0.470	0.261	0.048	0.260
0.1	0.5	0.1	0.494	0.183	0.266	0.314
0.1	0.1	1.0	0.400	0.183	0.314	0.299
0.1	0.1	0.5	0.423	0.235	0.507	0.389
0.1	0.1	0.1	0.517	0.183	0.314	0.338

Table 14: Sensitivity of SPA to the loss weights λ_{pref}, λ_{anchor}, and $\lambda_{\text{template}}$ on the Biomedical domain. The selected configuration $(\lambda_{\text{pref}}, \lambda_{\text{anchor}}, \lambda_{\text{template}}) = (1.0, 0.5, 0.1)$ (bold) achieves the highest average accuracy across MedQA, PubMedQA, and MMLU-Pro-Bio. Based on this analysis, λ_{pref} is fixed to 1.0 in the final objective.

Setting	Value
Judge model	GPT-4
Sampled candidate pools	10,000
Candidates per pool (K)	4
Chance baseline (uniform over K)	25.0%
Agreement with KARMA chosen path	74.8%
Near-equivalent disagreements	80.1%
Fully divergent disagreements	19.9%
Rationale-candidate mismatch	40.9%
Tie handling	Counted as disagreement

Table 15: GPT-4 validation of KARMA chosen-path selection. For each sampled pool, GPT-4 ranks all candidates without preference labels; agreement denotes the fraction of pools in which GPT-4's top-ranked candidate matches the chosen path selected by support-based Top-K. Near-equivalent and fully divergent cases are reported over disagreement pools. Rationale-candidate mismatch denotes disagreements in which the judge's explanation refers to an entity from a candidate other than the one selected.

E Chosen Path Validation

We conduct an LLM-judge study to assess whether recurrence-based support provides a meaningful relative evidence signal among candidates. For each sampled candidate pool, GPT-4 receives the source-target pair, the shared schema scaffold, and all K verbalized candidates without access to the KARMA preference labels. The judge selects the candidate whose intermediate entities form the most factually and semantically plausible reasoning chain. We measure agreement between this selection and the chosen path produced by support-based Top-K. Ties are conservatively counted as disagreement.

As reported in Table 15, GPT-4 agrees with KARMA's chosen path on 74.8% of 10,000 sampled pools, compared with a 25.0% chance baseline for $K = 4$. This agreement provides external evidence that recurrence-based support captures path plausibility, but it does not establish the selected path as uniquely correct. The LLM judge is itself an imperfect evaluator and is used here as an auxiliary validation signal rather than ground truth.

We therefore analyze the disagreement pools. In 80.1% of disagreements, the judge-selected and KARMA-selected paths share one or two of their three intermediate entities and differ only at the remaining entity-slots. For example, the paths `CA4` $\rightarrow$ `Hydroflumethiazide` $\rightarrow$ `hypertension` and `CA4` $\rightarrow$ `Furosemide` $\rightarrow$ `hypertension` differ only at the drug slot while representing closely related mechanisms. Only 19.9% of disagreement pools contain fully divergent paths.

Judge noise also contributes to the residual disagreement. In 40.9% of disagreement pools, the judge's rationale mentions an entity belonging to a candidate other than the candidate it selects. Consequently, disagreement cannot be interpreted solely as failure of the recurrence prior. The results instead characterize a fine-grained regime in which multiple candidates may be partially defensible and differ in relative evidence strength. This is the regime targeted by SPA: shared or aligned entity-slots are preserved, while preference supervision is concentrated on the entity-slots where candidates diverge.

F Qualitative Analysis

Table 16 shows a representative example of a KARMA-generated candidate pool from the biomedical domain, and Table 17 provides the corresponding path-only view.

The candidate pool in Tables 16 and 17 illustrates why preference objective resolution matters. All candidates share the source entity, target entity, relation schema, template scaffold, and all intermediate entities except the Compound slot. A sequence-level objective assigns one preference label to each complete sequence despite this localized difference, whereas SPA applies contrastive supervision directly to the differing Compound entities while retaining the shared scaffold as a language-modeling signal.

G Cross-Backbone and Matched-K Comparison

We evaluate whether SPA's gains persist under a second model family and whether they can be explained solely by the number of candidates. We repeat the Chemistry-domain comparison using Llama-3.2-3B with the same KARMA-synthesized data and training pipeline. TDPO is evaluated with both $K = 1$ and the matched $K = 4$ setting, while SPA uses $K = 4$.

Increasing TDPO from $K = 1$ to $K = 4$ improves its average accuracy from 0.279 to 0.295, showing that additional candidates provide useful supervision. Under the same $K = 4$ candidate budget, SPA reaches 0.339 average accuracy and outperforms TDPO on all three benchmarks. The matched comparison indicates that SPA's gain is not explained solely by the number of candidates, while the result on a second model family provides initial evidence of cross-backbone generality.

Field	Example
Instruction	Using the following biomedical data, explain how the biological cascade initiating from Anatomy 'internal carotid artery' reaches Biological Process.
Chosen candidate	This path represents a key hypothesis explaining the biological causality between **'internal carotid artery'** and **'phospholipase C-activating G-protein coupled receptor signaling pathway'**. **[Detailed Mechanism]**: From a clinical perspective, **'internal carotid artery'** forms a biological interaction network with **migraine**, **'migraine'** represents a condition where **Ergotamine** is administered for symptomatic relief or cure, **'Ergotamine'** forms a biological interaction network with **HTR2A**, **'HTR2A'** is involved in **phospholipase C-activating G-protein coupled receptor signaling pathway**, regulating physiological homeostasis. Furthermore ultimately leading to this conclusion.
Rejected candidate 1	This path represents a key hypothesis explaining the biological causality between **'internal carotid artery'** and **'phospholipase C-activating G-protein coupled receptor signaling pathway'**. **[Detailed Mechanism]**: From a clinical perspective, **'internal carotid artery'** forms a biological interaction network with **migraine**, **'migraine'** represents a condition where **Cyproheptadine** is administered for symptomatic relief or cure, **'Cyproheptadine'** forms a biological interaction network with **HTR2A**, **'HTR2A'** is involved in **phospholipase C-activating G-protein coupled receptor signaling pathway**, regulating physiological homeostasis. Furthermore ultimately leading to this conclusion.
Rejected candidate 2	This path represents a key hypothesis explaining the biological causality between **'internal carotid artery'** and **'phospholipase C-activating G-protein coupled receptor signaling pathway'**. **[Detailed Mechanism]**: From a clinical perspective, **'internal carotid artery'** forms a biological interaction network with **migraine**, **'migraine'** represents a condition where **Methylergometrine** is administered for symptomatic relief or cure, **'Methylergometrine'** forms a biological interaction network with **HTR2A**, **'HTR2A'** is involved in **phospholipase C-activating G-protein coupled receptor signaling pathway**, regulating physiological homeostasis. Furthermore ultimately leading to this conclusion.
Rejected candidate 3	This path represents a key hypothesis explaining the biological causality between **'internal carotid artery'** and **'phospholipase C-activating G-protein coupled receptor signaling pathway'**. **[Detailed Mechanism]**: From a clinical perspective, **'internal carotid artery'** forms a biological interaction network with **migraine**, **'migraine'** represents a condition where **Cyclobenzaprine** is administered for symptomatic relief or cure, **'Cyclobenzaprine'** forms a biological interaction network with **HTR2A**, **'HTR2A'** is involved in **phospholipase C-activating G-protein coupled receptor signaling pathway**, regulating physiological homeostasis. Furthermore ultimately leading to this conclusion.

Table 16: Example of a generated candidate pool.

H Gradient Analysis

We directly examine whether each preference objective concentrates its training signal on the entity-slots where candidates differ. DPO, TDPO, and SPA are trained on the same contrastive candidate pools. DPO and TDPO consume each pool as pairwise comparisons against the same chosen path, while entity annotations are used only post hoc to group tokens for analysis and do not modify either baseline objective.

For each method, we measure per-token objective sensitivity separately for chosen-entity tokens, rejected-entity tokens, and shared template tokens. Because absolute gradient magnitudes depend on objective scaling, each group is normalized by the mean sensitivity of template tokens within the same method. The resulting ratios are comparable across token groups within a method, but not as absolute magnitudes across methods. The All Entity / Template ratio is computed as a per-token mean over all entity tokens rather than as the arithmetic average of the chosen and rejected ratios. Results are averaged over nine checkpoints from training steps 100 to 900.

DPO assigns nearly uniform sensitivity to chosen entities, rejected entities, and shared template tokens. TDPO places relatively stronger sensitivity on rejected entities but does not concentrate it on the chosen entity-slots. SPA assigns $2.18\times$ the template-token sensitivity to chosen entity tokens while keeping rejected-entity sensitivity near the template level. This analysis provides direct empirical evidence that SPA routes supervision toward the construction-exposed entity-slots rather than uniformly amplifying all tokens.

Preference label	Candidate path
Chosen	internal carotid artery → migraine → **Ergotamine** → HTR2A → phospholipase C-activating GPCR signaling pathway
Rejected 1	internal carotid artery → migraine → **Cyproheptadine** → HTR2A → phospholipase C-activating GPCR signaling pathway
Rejected 2	internal carotid artery → migraine → **Methylergometrine** → HTR2A → phospholipase C-activating GPCR signaling pathway
Rejected 3	internal carotid artery → migraine → **Cyclobenzaprine** → HTR2A → phospholipase C-activating GPCR signaling pathway

Table 17: Path-only view of the candidate pool in Table 16. All four candidates share the same Anatomy, Disease, Gene, and Biological Process nodes, while differing only in the Compound node (bold), illustrating the entity-slot resolution at which preference learning must operate.

Method	MMLU-Chem	MMLU-Pro-Chem	GPQA-Chem	Average
Base	0.488	0.143	0.182	0.271
TDPO ($K = 1$)	0.468	0.145	0.225	0.279
TDPO ($K = 4$)	0.495	0.153	0.236	0.295
SPA ($K = 4$)	**0.553**	**0.179**	**0.284**	**0.339**

Table 18: Cross-backbone and matched-candidate comparison on the Chemistry domain using Llama-3.2-3B.

Method	Chosen / Template	Rejected / Template	All Entity / Template
DPO	1.00±0.05	0.99±0.06	0.99±0.06
TDPO	0.80±0.03	1.19±0.04	1.00±0.04
SPA	**2.18±0.34**	0.96±0.12	**1.36±0.18**

Table 19: Per-token objective sensitivity normalized by the mean template-token sensitivity within each method. Ratios are interpreted only within a method.

I SPA Loss Component Ablation

We ablate the three components of SPA on the Chemistry domain using Qwen2.5-7B-Instruct. The preference term provides relative supervision between chosen and rejected entity variants, the anchor term increases the absolute likelihood of chosen entities, and the template term preserves the shared language scaffold.

Configuration	MMLU-Chem	MMLU-Pro-Chem	GPQA-Chem	Average
Base	0.500	0.148	0.161	0.270
Preference only	0.580	0.182	0.204	0.322
Preference + Anchor	0.613	0.181	0.161	0.318
Preference + Template	0.606	**0.196**	0.161	0.321
Full SPA	**0.630**	0.195	**0.204**	**0.343**

Table 20: Ablation of SPA's preference, anchor, and template loss components on the Chemistry domain.

The preference term alone improves average accuracy from 0.270 to 0.322, showing that slot-level contrastive supervision provides the primary gain. The anchor and template terms improve different evaluation axes when added individually. The full objective achieves the highest average accuracy, the best MMLU-Chem result, a tied-best GPQA-Chem result, and performance within 0.001 of the best MMLU-Pro-Chem result. This pattern indicates that the three components play distinct and complementary roles.