SERM: Self-Evolving Relevance Model with Agent-Driven Learning from Massive Query Streams

Anonymous ACL submission

Abstract

Due to the dynamically evolving nature of real-world query streams, relevance models struggle to generalize to practical search scenarios. A sophisticated solution is self-evolution techniques. However, in large-scale industrial settings with massive query streams, this technique faces two challenges: (1) informative samples are often sparse and difficult to identify, and (2) pseudo-labels generated by the current model could be unreliable. To address these challenges, in this work, we propose a **S**elf-**E**volving **R**elevance **M**odel approach (SERM), which comprises two complementary multi-agent modules: a *multi-agent sample miner*, designed to detect distributional shifts and identify informative training samples, and a *multi-agent relevance annotator*, which provides reliable labels through a two-level agreement framework. We evaluate SERM on a large-scale industrial platform, which serves billions of user requests daily. Experimental results demonstrate that SERM can achieve significant performance gains through iterative self-evolution, as validated by extensive offline multilingual evaluations and online testing.

1 Introduction

Search relevance is central to modern information retrieval, aiming to rank documents that best satisfy a user query (Yin et al., 2016; Li et al., 2015). With the explosion of information on platforms such as Google and TikTok, effective relevance modeling has become increasingly important (Chen et al., 2024). Traditional approaches encode queries and documents into vectors and learn a scoring function (Gao et al., 2020; Zou et al., 2021), while recent work leverages large language models (LLMs) to directly generate relevance judgments (Zhuang et al., 2024; Ye et al., 2025).

Despite recent progress, relevance modeling still suffers from significant generalization limitations. This stems from the dynamic and continuously evolving nature of real-world query distributions, which makes it difficult for relevance models to generalize effectively to practical search scenarios. For instance, on online search platforms, users often issue queries containing newly emerged expressions or cultural references, such as *remember me pets arriving on 10/27*. Such queries encode nuanced meanings that models often fail to capture, leading to mismatches between user intent (i.e., *commemorating deceased pets*) and retrieved content (i.e., *generic pets returning home*).

There is much work on addressing this limitation by pre-training models on large-scale document corpora (Zou et al., 2021; Zhang et al., 2023; Ma et al., 2024). However, such approaches primarily capture domain-specific knowledge from static data and fail to account for the fact that real-world query streams continuously introduce novel expressions and emerging linguistic patterns.

A more sophisticated approach is self-evolution techniques, including self-training (Gulcehre et al., 2023) and self-reflection (Huang et al., 2023), where a model leverages its own predictions on unlabeled user queries to enhance its generalization. While such approaches have shown promise, applying self-evolution in industrial-scale search scenarios poses two fundamental challenges: (**C1**) informative samples are sparse and difficult to identify within massive query streams, and (**C2**) pseudo-labels generated by the current model may be unreliable, potentially leading to error accumulation.

To address these challenges, we explore strategies for self-evolving relevance models that can continuously adapt to massive query streams. To this end, we develop an LLM-based relevance model that, given a query and a document, generates both a relevance score (e.g., 0–4) and the corresponding rationale. Building on this foundation, in this work, we propose a **S**elf-**E**volving **R**elevance **M**odel approach (SERM), which comprises two complementary multi-agent modules: a

multi-agent sample miner and a *multi-agent relevance annotator*. Specifically, the sample miner monitors incoming queries, detects distributional shifts driven by diverse user behaviors, and selects informative training samples where the model lacks sufficient knowledge to make accurate predictions (tackling **C1**). The annotator then produces reliable learning signals for these samples through a two-level agreement framework, enabling the model to iteratively refine itself and adapt to emerging user intents (tackling **C2**). To the best of our knowledge, we are the first to investigate the self-evolution of relevance models using massive query streams.

We employ a large-scale industrial search platform as a testbed to evaluate the effectiveness of our SEEM. Specifically, we conduct experiments on massive query streams to demonstrate how our SERM approach enables continuous model evolution. The experimental results demonstrate that SERM enables effective and scalable self-evolution of relevance models under industrial-scale settings involving millions of query–document pairs. Notably, after three iterations of self-evolution, SERM achieves a +2.99 point improvement in NDCG@1 over the baseline model. Moreover, unlike self-training, which relies solely on the model's own predictions and often suffers from severe error propagation, SERM provides reliable labels that mitigate this issue, enabling consistent and sustained performance gains.

2 Preliminaries

2.1 Task Formulation

Given a query q and a collection of candidate documents $D = \{d_1, d_2, \cdots, d_m\}$, where m denotes the number of documents, the goal of a relevance model is to compute a score for each document (MacAvaney et al., 2019; Nogueira et al., 2019). These scores are used to rank documents, allowing the system to return results that best match the query intent. Here, we focus on document search as a representative case of relevance modeling, though the term *document* can also encompass other searchable content such as videos or images.

2.2 Continual Pre-Training

As shown in Figure 1(a), the development of search relevance models typically follows a two-stage recipe: continual pre-training followed by supervised fine-tuning (SFT). There are two commonly used approaches for continual pre-training. One simple approach is to pre-train the model on a large corpus of documents. This allows the model to better capture domain-specific semantics and patterns (Zou et al., 2022; Wu et al., 2024a).

A second approach is to model relationships between document attributes. For instance, before supervised training, we can generate tasks where the model predicts certain document attributes like post content (Zhang et al., 2023).

2.3 Supervised Fine-Tuning

After continual pre-training, we use labeled data to train the model further through SFT. At this stage, modeling approaches can be broadly categorized into *discriminative* and *generative* methods.

Discriminative Relevance Modeling. The discriminative modeling approach uses a pre-trained encoder to represent the query q and document d as feature vectors, which are then fed into a scoring function to produce a relevance score. Training typically follows either a pairwise ranking objective or a regression-based objective. For the pairwise ranking objective, given a non-relevant document d_a, the model is trained to assign a higher score to d_b than to d_a through a Bradley–Terry loss function (Bradley and Terry, 1952), as follows

$$\mathcal{L}_{\mathrm{d}} = -\mathbb{E}_{(q,d_a,d_b)\sim\mathcal{D}_r} \Big[\log \sigma\big(f(q,d_b) - f(q,d_a)\big)\Big] \quad (1)$$

where $D_r = (q, d_a, d_b)$ denotes the labeled pairwise dataset, $f(\cdot)$ is the relevance scoring model, and $\sigma(\cdot)$ is the sigmoid function. Additionally, when graded relevance labels $y \in \mathcal{Y}$ are available, where $\mathcal{Y}$ denotes the set of label tokens (e.g., $\{0, 1, 2, 3\}$), a regression objective can also be adopted by minimizing the mean squared error between the predicted score and the label.

Generative Relevance Modeling. The generative modeling approach utilizes the generation capability of LLMs to directly produce relevance judgments. Instead of encoding the query and document into feature vectors and training a scoring function, the model is trained to generate a discrete relevance label. The training objective is formulated as a cross-entropy loss over the target label tokens:

$$\mathcal{L}_{\mathrm{g}} = -\mathbb{E}_{(q,d,y)\sim\mathcal{D}_r} \log \Pr_\theta(y|q,d) \quad (2)$$

where $\Pr_\theta(\cdot)$ is the probability distribution defined by an LLM with parameters θ. This modeling ap-

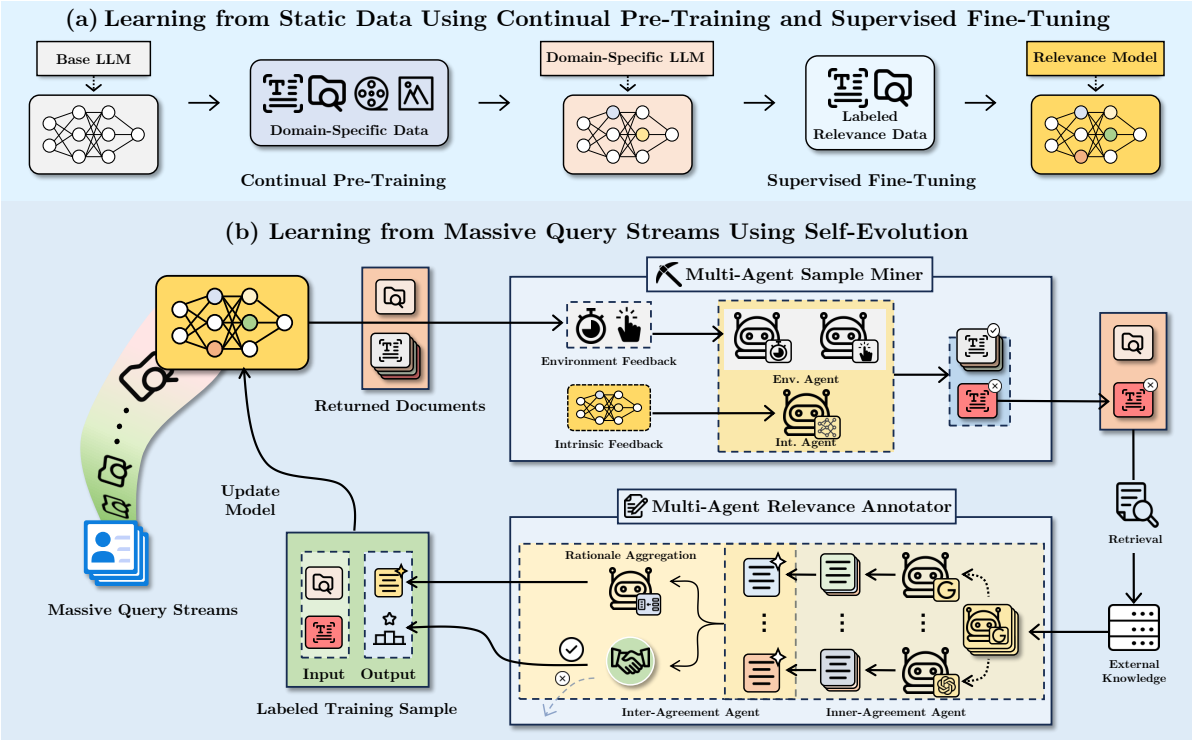


Figure 1: **(a) Learning from Static Data:** The conventional training recipe is to first apply continual pre-training and then perform supervised fine-tuning on static labeled data. **(b) Learning from Massive Query Streams:** The proposed SERM uses a multi-agent sample miner to identify informative samples and a multi-agent relevance annotator to generate reliable labels, enabling continuous model evolution with massive query streams.

proach enables the model not only to predict a relevance score but also to generate interpretable rationales before the score, thereby enhancing robustness in dynamic search scenarios (Zhuang et al., 2024; Ji et al., 2025). Once trained, the model can be directly deployed to generate relevance scores for unseen query–document pairs.

3 Self-Evolving Search Relevance Models

Our goal is to achieve self-evolution, enabling the model to continuously adapt and refine its relevance predictions based on evolving user queries and behaviors. To achieve this, we propose the SERM, which consists of two multi-agent modules: the multi-agent sample miner (MSM) and the multi-agent relevance annotator (MRA), as illustrated in Figure 1(b). The following subsections describe these modules in detail.

3.1 Multi-Agent Sample Miner

The MSM is designed to identify informative and challenging query–document pairs from massive query streams to drive the self-evolution of the relevance model. The basic idea is that not all samples contribute equally to model improvement: easy or well-predicted pairs provide little additional information, whereas those that reveal the model's weaknesses are more valuable for further learning. To this end, the MSM employs multiple complementary agents, as described below.

3.1.1 Environmental Feedback Agents

We design two agents that capture inconsistencies between the relevance model and its surrounding environment. These two agents use feedback signals from user interactions and auxiliary models to identify query–document pairs that are potentially informative and challenging for further learning:

User Feedback Agent. We design an agent that employs user interaction signals, including clicks and dwell time, to identify informative query–document pairs for training relevance models. This agent evaluates each pair (q, d) based on two criteria. First, it detects strong positive user engagement, determined by $\big(\mathbf{1}\text{click}(q,d) = 1\big) \vee \big(U(q,d) > \tau_u\big)$, where $\mathbf{1}\text{click}(\cdot)$ indicates whether document d was clicked for q, $U(q,d)$ is a dwell-time–based engagement metric, and τ_u is the engagement threshold. Second, it assesses the pair's difficulty for the current relevance model by checking whether $f(q,d) < \tau_c$, where τ_c is the confidence threshold. When both conditions are met,

the agent outputs a feedback signal highlighting the discrepancy between strong user interest and low model confidence, thereby recommending the pair as a valuable candidate for guiding self-evolution.

Click Model Feedback Agent. We design a click model feedback agent that augments user feedback by compensating for the biases and sparsity of raw click signals. In practice, user clicks are often influenced by position and presentation biases (Bar-Ilan et al., 2009) and tend to be sparse or delayed, particularly for tail queries or newly introduced documents (Chuklin et al., 2022). To mitigate these issues, the agent leverages a pre-trained click model R_{cm}, which estimates the probability that a user would click a candidate document given a query. Instead of relying solely on raw click logs, the agent evaluates each query–document pair by comparing the predicted click probability from R_{cm} with the relevance model's confidence. If the predicted click probability exceeds a threshold τ_{cm}, the agent treats the pair as clicked, i.e., $\mathbf{1}_{\text{click}}(q, d) = 1$.

3.1.2 Intrinsic Feedback Agent

The intrinsic feedback agent uses internal signals of the relevance model to identify query–document pairs that best reveal its weaknesses. Specifically, we design two feedback signals: model disagreement and model uncertainty.

Model Disagreement. We prompt the LLM to generate K relevance judgments with accompanying rationales using temperature sampling, producing a set of scores $\{f^k(q, d)\}_{k=1}^{K}$. The disagreement among these judgments is quantified as

$$\text{MD}(q, d) = \max_{i,j} \left| f^i(q, d) - f^j(q, d) \right| \qquad (3)$$

where a larger value indicates greater inconsistency in the model's predictions.

Model Uncertainty. We compute the model's prediction uncertainty using the entropy of the relevance label distribution:

$$\begin{aligned} \text{MU}(q, d) \quad = \quad & -\sum_{y} \Pr_\theta(y \mid q, d) \\ & \times \log \Pr_\theta(y \mid q, d) \end{aligned} \qquad (4)$$

where higher entropy denotes lower confidence and suggests that the pair is harder for the model. By jointly considering both disagreement and uncertainty, the agent identifies samples with high inconsistency and low confidence, which are particularly informative for guiding the model's self-evolution.

Using the designed agents, the MQM module selects query–document pairs that meet the specified conditions. Specifically, for each new query, it samples n candidate documents identified by each agent, forming a set of trainable query–document pairs. Duplicate documents are removed during this process. The union of these sampled pairs constitutes a collection of hard and informative cases, which are then passed to the MRA module to generate reliable learning signals.

3.2 Multi-Agent Relevance Annotator

Building on the samples identified by the MSM, we propose a two-level agreement framework to enhance the accuracy and robustness of automated relevance annotation using multiple large-scale models. This framework operates in two stages: inner-agreement and inter-agreement, ensuring that the selected query–document pairs are annotated with both reliable labels and coherent rationales.

Inner-Agreement Agent. The inner-agreement agent operates by leveraging multiple LLMs (e.g., `GPT-4o` and `Gemini2.5-Pro`) to perform relevance annotation. Acting as autonomous evaluators, the agent first retrieves external knowledge relevant to the given query–document pair and then reasons about its relevance using the retrieved context. To ensure robustness, the agent employs a multi-path chain-of-thought strategy (Thomas et al., 2024), generating multiple independent reasoning paths for each pair. These reasoning paths produce a set of candidate relevance labels, which the agent consolidates via majority voting to arrive at a stable label prediction. By internally reconciling diverse reasoning paths, the inner-agreement agent mitigates the randomness of single-pass LLM outputs and improves intra-agent reliability.

Inter-Agreement Agent. The inter-agreement agent operates by reconciling the outputs of multiple inner-agreement agents. After receiving stable label predictions from each agent, it filters the results by retaining only those query–document pairs on which the agents reach consensus, thereby constructing a high-confidence annotated dataset. Beyond label selection, the inter-agreement agent also assumes responsibility for rationale generation. To this end, it collects all reasoning paths that support the agreed-upon relevance label and consolidates them into a single, coherent explanation. By enforcing cross-agent consensus and producing unified rationales, the inter-agreement agent enhances

annotation quality and ensures that only reliable signals are passed forward for self-evolution.

Importantly, although MRA employs external feedback, it fundamentally differs from knowledge distillation (Kim and Rush, 2016). Please refer to Appendix B.1 for a detailed discussion of these differences. In essence, we can consider the MRA module functions as a form of evolutionary feedback that iteratively corrects and refines the model's outputs, rather than transferring static knowledge from a fixed teacher. This design aligns with recent work on self-evolving systems, where external tools or auxiliary models are employed to refine model predictions and provide feedback signals for sustained self-improvement (Gou et al., 2024; Zhou et al., 2024).

4 Experiments

We evaluated our self-evolving approach on an online social platform, focusing on a document search task and employing the widely used Qwen2.5-7B and Qwen2.5-1.5B models. Notably, the platform handles a massive daily volume of user queries from multiple countries, offering a realistic and challenging testbed for our research problem.

4.1 Datasets

For continual pre-training, we used a corpus of 100B tokens, primarily derived from the platform's document collection. For SFT, we employed 3.6M labeled query–document pairs, each labeled according to its relevance score (ranging from 0 to 3, indicating bad, fair, good, and excellent, respectively). During self-evolution, each iteration sampled approximately 700K user query–document pairs from the platform's database to run SERM, with iterations spaced two weeks apart to ensure sufficient shifts in the user query distribution. Note that since the platform serves users across multiple countries, our datasets were multilingual, with the language distribution shown in Table 4 in the Appendix. Additionally, given that this work focuses on evolving relevance models from massive real-world user queries, existing open-source datasets did not meet our research needs. Consequently, all datasets used in this study were collected from real-world industrial application scenarios.

4.2 Settings

All of the trained relevance models, including the teacher and distilled models, employ a generative modeling approach, as described in Eq. 2. For the self-evolution phase, we executed three iterations, progressively augmenting the training dataset with each iteration. We used `GPT-4o` and `Gemini2.5-Pro` in the MRA module to generate three reasoning paths for each query–document pair. During each iteration, we mixed the newly generated data, previously generated data, and the original SFT data, and retrained the model to prevent catastrophic forgetting (Luo et al., 2025). More training settings are shown in Appendix A.

4.3 Evaluation

We conducted offline testing using an in-house test set that categorizes languages into three families: Germanic, Romance, and Minor Languages (He et al., 2024), with the distribution shown in Table 4. This test set was specifically chosen to better reflect the language distribution and real-world application scenarios of our models. We reported model performance based on NDCG@1, NDCG@4, and relevance accuracy (Acc.). Additionally, in industrial-scale search scenarios, relevance models must often be distilled into smaller models to meet strict latency requirements (Yao et al., 2022). Therefore, we also evaluated the performance of small models distilled from different trained relevance models. Specifically, we used Qwen2.5-0.5B as the small model and performed distillation using all of our SFT data, employing the Kullback-Leibler divergence-based distillation method as described in Ye et al. (2025)'s work.

4.4 Baselines

Our baseline for comparison was the traditional training pipeline, consisting of continual pre-training followed by SFT on static data (denoted as *CT+SFT*). Additionally, to demonstrate the effectiveness of our multi-agent module, we compared it with the self-training approach. In this baseline, instead of using our MRA module for annotation, we relied on the relevance model to annotate the data itself (denoted as *Self-Training*). It is worth noting that during the self-training process, we ensured that the hyperparameters and query-document pairs used were consistent with those used in SERM to make a fair comparison.

4.5 Offline Evaluation Results

We conduct offline evaluations on the trained relevance models using a static, large-scale test set. The results are listed in Table 1. First, compared to CT+SFT, we observe that both self-training and

Method	Germanic			Romance			Minor Language		
	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.
Training with Qwen2.5-7B Model									
CT+SFT	84.74	86.68	55.30	85.61	87.33	50.74	82.02	84.02	52.89
Self-Training									
Iteration 1	84.87	86.75	55.74	85.82	87.43	51.40	82.30	84.12	53.83
Iteration 2	84.95	86.78	55.85	85.72	87.35	51.85	82.29	84.12	54.25
Iteration 3	84.78	86.72	55.63	85.58	87.33	51.85	82.20	84.08	54.45
+Distillation	83.86	86.21	54.88	84.75	86.68	50.98	81.47	83.50	53.73
SERM									
Iteration 1	87.04	87.53	56.50	87.72	88.12	52.74	84.55	84.96	54.24
Iteration 2	87.27	87.60	57.40	88.01	88.22	53.25	84.79	85.05	54.40
Iteration 3	**87.56**	**87.71**	**57.79**	**88.14**	**88.27**	**53.51**	**84.99**	**85.12**	**55.07**
+Distillation	86.78	87.27	56.92	87.68	87.84	52.74	84.43	84.78	54.42
Training with Qwen2.5-1.5B Model									
CT+SFT	84.59	86.63	54.75	85.99	87.44	50.24	81.75	83.91	51.81
Self-Training									
Iteration 1	84.91	86.77	55.22	86.07	87.51	51.08	82.00	84.02	53.09
Iteration 2	84.93	86.75	55.49	85.98	87.47	51.22	82.10	84.05	53.32
Iteration 3	85.04	86.79	55.66	85.86	87.43	51.60	82.19	84.09	53.74
+Distillation	84.17	86.34	54.77	84.88	86.79	50.83	81.65	83.49	52.86
SERM									
Iteration 1	86.64	87.38	55.57	87.46	88.02	51.98	84.11	84.81	53.51
Iteration 2	87.03	87.52	56.57	87.68	88.09	52.55	84.46	84.93	54.07
Iteration 3	**87.30**	**87.62**	**56.78**	**87.83**	**88.16**	**53.25**	**84.75**	**85.03**	**54.35**
+Distillation	86.54	86.97	56.06	86.96	87.33	52.70	84.40	84.60	53.68

Table 1: Performance of relevance models on various language families. The best result in each group is in bold. "+Distillation" denotes the distillation of the relevance model from the third iteration to a small LLM (0.5B). "ND@1" and "ND@4" denote NDCG@1 and NDCG@4, respectively.

SERM show significant improvements, regardless of whether the model is Qwen2.5-7B or Qwen2.5-1.5B. This confirms that incorporating large volumes of user query streams into the training process effectively enhances model performance. Additionally, we observe that SERM outperforms self-training in terms of both stability and accuracy. This highlights the effectiveness of the multi-agent framework and the integration of external knowledge in improving model robustness. Interestingly, we find that noise can be easily introduced into the self-training process. For example, in the Qwen2.5-7B model, the performance of the third iteration on the Germanic language family significantly drops compared to the second iteration (e.g., NDCG@1 drops from 84.95 to 84.78). We conjecture that this is due to error propagation during the self-training iterations (Zhu et al., 2023). In contrast, SERM consistently demonstrates stable improvements, which we attribute to the integration of external knowledge and annotations that mitigate such issues. Finally, when comparing the distilled models, we see that the model distilled from SERM Iteration 3 outperforms the one distilled from Self-Training Iteration 3. This further validates the effectiveness of the SERM approach in delivering superior relevance model performance.

4.6 Online Testing Results

Online A/B Testing. We conduct an A/B test on the small model distilled from our 7B model (SERM Iteration 3), with the results summarized in Table 2. From the results, we observe that our model can achieve a significant improvement in 14-day retention, with a +0.0359% gain and a p-value of 0.0278, indicating enhanced long-term user engagement. We also find that the model can improve user satisfaction, as reflected by a slight decrease

Metric	Gain	P-value
User Negative Feedback	-1.2081%	0.0001
Change Query Ratio	-0.0839%	0.0023
Change Query Ratio (Longtail)	-0.1312%	0.0015
14-Day Retention	+0.0359%	0.0278

Table 2: Performance metrics from an online A/B testing using the student model distilled from the Qwen2.5-7B model trained with SERM. The definitions of the metrics are provided in Appendix A.3.

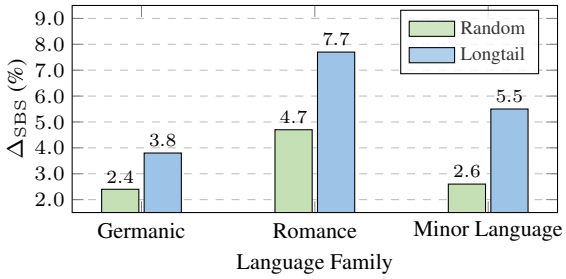


Figure 2: Side-by-side manual evaluation results comparing our proposed SERM with the baseline.

in user negative feedback by -1.2081% (p-value: 0.0001)[1]. These findings show the effectiveness of the SERM approach in enhancing both user retention and satisfaction through the self-evolution of the model using massive query streams.

Side-by-Side Manual Evaluation. We further conduct a side-by-side (SBS) evaluation on our crowdsourcing platform, where annotators directly compare the outputs of the experimental system against the baseline to determine which provides better relevance. Detailed evaluation settings are described in Section A.3. The advantage ratio of a given strategy is computed as:

$$\Delta_{\text{SBS}} = \frac{G - B}{G - B + S} \tag{5}$$

where G denotes the number of cases in which the experimental strategy is preferred over the baseline, B denotes the number of cases in which the baseline is preferred, and S denotes the number of cases in which annotators see no clear difference (i.e., a tie). A higher Δ_{SBS} indicates a stronger advantage of the experimental strategy over the baseline. The SBS results are summarized in Figure 2. We can observe that SERM consistently outperforms the baseline on both random and long-tail test samples across all language families.

[1]On our platform, improvements of 0.01% in metrics such as change query ratio and 14-day retention are considered significant due to the large user base (see Appendix A.3 for details). This is also consistent with prior work on industrial-scale search relevance evaluation (Zhou et al., 2025).

Method	Germ.	Roma.	Minor.
CT+SFT	86.68	87.33	84.02
SERM	**87.71**	**88.27**	**85.12**
w/o User Feedback	87.01	87.93	84.41
w/o CM Feedback	86.93	88.10	84.39
w/o Model Disagreement	86.95	87.74	84.57
w/o Model Uncertainty	87.17	87.51	84.69
w/o Inner Agreement	87.41	88.03	84.97
w/o Inter Agreement (GPT)	86.48	86.76	83.57
w/o Inter Agreement (Gemini)	86.84	87.47	84.36

Table 3: Ablation study on the MSM module of SERM using the Qwen2.5-7B model.

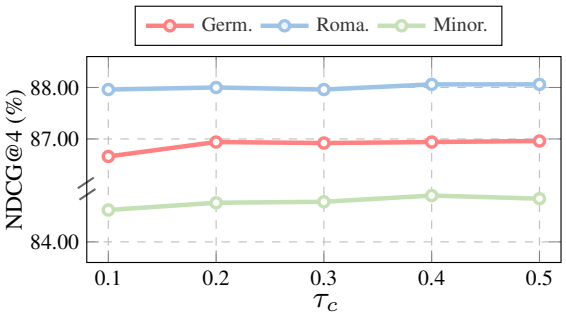
4.7 Ablation Studies

To evaluate the contribution of each agent within the MSM and MRA modules, we conduct an ablation study. Specifically, we re-run the SERM while removing one component at a time, including the user-feedback agent, click-model feedback agent, model disagreement, model uncertainty, inner-agreement agent, and inter-agreement agent. When the inner-agreement agent is removed, each LLM generates only a single reasoning path. When the inter-agreement agent is removed, we skip the external agreement filtering and instead directly aggregate all reasoning results from different LLMs via majority voting.

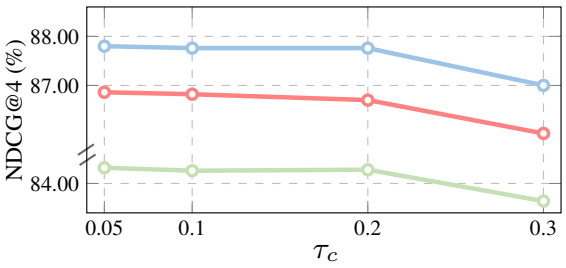
Table 3 summarizes the results. First, the results confirm that each component contributes valuable and complementary signals, enabling the agents to more effectively identify informative samples. The non-overlapping nature of these signals highlights the importance of integrating all components to achieve optimal self-evolution performance. Second, we find that both the inner-agreement and inter-agreement agents provide notable gains, underscoring their role in producing more reliable labels for the model's iterative evolution. Interestingly, we see that although using a single GPT model alone for annotation can significantly degrade SERM's performance due to its weaker capability on relevance tasks, GPT still proves beneficial for agreement analysis. This observation aligns with recent findings in *weak-to-strong generalization* (Burns et al., 2024), where a weak model can help a strong model improve further.

4.8 Effect of τ_c and τ_{cm} on Performance.

We further evaluate the performance of SERM under different hyperparameter settings. Specifically, we conduct experiments using the Qwen2.5-7B model with varying thresholds for the relevance model (τ_c) and the click model (τ_{cm}). For τ_c, we



(a) Relevance Model



(b) Click Model

Figure 3: Performance on different thresholds.

test values in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and for τ_{cm}, we test values in $\{0.05, 0.1, 0.2, 0.3\}$. We use a validation set with the same distribution as the test set to evaluate model performance under each configuration. The results are presented in Figure 3. Based on these experiments, we select $\tau_c = 0.4$ and $\tau_{\text{cm}} = 0.1$ as the default thresholds for SERM in subsequent self-evolution experiments.

5 Related Work

Relevance Modeling. Search relevance aims to rank items, such as documents, images, and videos, that best satisfy a user's query intent. Early approaches relied on pre-trained encoders (e.g., BERT (Devlin et al., 2019)) and discriminative models (Gao et al., 2020; Zou et al., 2021), while recent advances, inspired by the success of LLMs (Ouyang et al., 2022; Wang et al., 2024), have shifted toward generative approaches. For instance, recent works have prompted LLMs to generate the relevance of either a single query–document pair or a query with multiple candidate documents (Qin et al., 2024; Zhuang et al., 2024). To further enhance LLM performance on search relevance tasks, some works have used labeled data to fine-tune the models, improving their ability to align with human-labeled relevance judgments (Ma et al., 2024). Building on this, other works employed reinforcement learning to better exploit the reasoning capabilities of LLMs in relevance modeling (Tang et al., 2025; Zhuang et al., 2025). More recently, to address the limitations of LLMs in capturing domain-specific patterns, researchers explored continual pre-training on large-scale search data (Zhang et al., 2023; Ye et al., 2025). However, these approaches rely mainly on static data and overlook the potential of massive query streams in real-world applications.

Self-Evolving Large Language Models. This work joins a large body of research demonstrating that LLMs can evolve themselves using unlabeled data (Huang et al., 2023; Wang et al., 2023). One approach in this direction is *self-training*, where the model leverages its own predictions on unlabeled data as pseudo-labels and iteratively retrains itself (Huang et al., 2023; Gulcehre et al., 2023). Despite its simplicity and effectiveness in semi-supervised and relatively stable settings, self-training often suffers from error propagation when the pseudo-labels are noisy or unreliable (Wang et al., 2021; Mahmood et al., 2024). Subsequent research has sought to address these limitations by enriching the learning signals beyond simple pseudo-labels, for example, by incorporating model-generated rationales (Madaan et al., 2023; Lu et al., 2023) or reward-based feedback (Gulcehre et al., 2023) to better guide the learning process.

Multi-Agent Collaboration. The emergence of LLMs has opened new possibilities for multi-agent collaboration. In LLM-based multi-agent collaboration, each agent is instantiated as an LLM-powered entity capable of reasoning and planning (Qian et al., 2024; Guo et al., 2024). Systems such as AutoGPT, CAMEL (Li et al., 2023), and AutoGen (Wu et al., 2024b) demonstrate that LLM agents can assume diverse roles and accomplish complex tasks through dialogue-based coordination. In this work, we extend multi-agent collaboration to assist the self-evolution of relevance models.

6 Conclusion

We have explored how to learn from massive query streams to improve the generalization of relevance models. We have proposed a self-evolving relevance model approach, called SERM, which integrates a multi-agent sample miner and a multi-agent relevance annotator to enable continuous model adaptation. We validated the effectiveness of SERM on a large-scale industrial document search platform through both offline and online testing. The experimental results show that SERM can continuously enhance model performance.

Limitations

In this section, we discuss some limitations of this work as follows:

- *Dependence on Query Streams*. Our experiments rely on large-scale query logs from a single industrial platform. While this provides a realistic testbed for studying dynamic search environments, it limits the immediate reproducibility and generalizability of our findings. To mitigate this limitation, we will release a curated subset of the data and the offline test set after publication to support reproducibility and further research. All released data will be carefully filtered to exclude any user-identifiable information and harmful content.

- *Evaluation Scope*. Search tasks are inherently diverse, encompassing not only document retrieval but also modalities such as image and video search. Although the proposed SERM framework is, in principle, applicable across these modalities, it is infeasible to exhaustively evaluate it on each task one by one. Therefore, this work primarily focuses on document search, which we consider a representative and widely adopted scenario across modern platforms.

- *Dependence on Heuristics or Implicit Thresholds*. Several components of SERM, such as informative sample selection in the MSM module and agreement filtering in the MRA module, rely on heuristics or implicitly defined thresholds (e.g., confidence scores or agreement ratios). Nevertheless, in this work, we provide detailed and comprehensive guidelines for setting these hyper-parameters (see Section 4.8), which are designed based on practical considerations in industrial document search scenarios. Additionally, empirical results indicate that SERM is not overly sensitive to these hyperparameter choices, as stable performance improvements are observed across a reasonable range of settings.

Ethics Statement

This work leverages large-scale real-world query streams, which naturally raise privacy and ethical concerns. We strictly follow anonymization and aggregation protocols and ensure that all data is legally obtained and free of harmful content. As a result, the work does not pose specific ethical risks.

References

Judit Bar-Ilan, Kevin Keenoy, Mark Levene, and Eti Yaari. 2009. Presentation bias is significant in determining user preference for search results—a user study. *Journal of the American Society for Information Science and Technology*, 60(1):135–149.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. 2024. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Zeyuan Chen, Haiyan Wu, Kaixin Wu, Wei Chen, Mingjie Zhong, Jia Xu, Zhongyi Liu, and Wei Zhang. 2024. Towards boosting llms-driven relevance modeling with progressive retrieved behavior-augmented prompting. *ArXiv preprint*, abs/2408.09439.

Aleksandr Chuklin, Ilya Markov, and Maarten De Rijke. 2022. *Click models for web search*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding bert rankers under distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, pages 149–152.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *ArXiv preprint*, abs/2308.08998.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 8048–8057.

Yifei He, Alon Benhaim, Barun Patra, Praneetha Vaddamanu, Sanchit Ahuja, Parul Chopra, Vishrav Chaudhary, Han Zhao, and Xia Song. 2024. Scaling laws for multilingual language models. *ArXiv preprint*, abs/2410.12883.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore.

Yuelyu Ji, Zhuochun Li, Rui Meng, and Daqing He. 2025. Reason-to-rank: Distilling direct and comparative reasoning from large language models for document reranking. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2320–2329.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Lihong Li, Jinyoung Kim, and Imed Zitouni. 2015. Toward predicting the outcome of an A/B experiment for search relevance. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 37–46.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, and 1 others. 2023. Self: Self-evolution with language feedback. *ArXiv preprint*, abs/2310.00533.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2421–2425.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1101–1104.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Arif Mahmood, Wenxiong Kang, and 1 others. 2024. Pseudo-label refinement for improving self-supervised learning systems. *ArXiv preprint*, abs/2410.14242.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *ArXiv preprint*, abs/1910.14424.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and 1 others. 2024. Scaling large language model-based multi-agent collaboration. *ArXiv preprint*, abs/2406.07155.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico.

Tian Tang, Zhixing Tian, Zhenyu Zhu, Chenyang Wang, Haiqing Hu, Guoyu Tang, Lin Liu, and Sulong Xu.

2025. Lref: A novel llm-based relevance framework for e-commerce search. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 468–475.

Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2024. Large language models can accurately predict searcher preferences. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 1930–1940.

Chenglong Wang, Hang Zhou, Kaiyan Chang, Bei Li, Yongyu Mu, Tong Xiao, Tongran Liu, and Jingbo Zhu. 2024. Hybrid alignment training for large language models. *ArXiv preprint*, abs/2406.15178.

Qianlong Wang, Zhiyuan Wen, Qin Zhao, Min Yang, and Ruifeng Xu. 2021. Progressive self-training with discriminator for aspect term extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 257–268, Online and Punta Cana, Dominican Republic.

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, Toronto, Canada.

Kaixin Wu, Yixin Ji, Zeyuan Chen, Qiang Wang, Cunxiang Wang, Hong Liu, Baijun Ji, Jia Xu, Zhongyi Liu, Jinjie Gu, and 1 others. 2024a. Cprm: A llm-based continual pre-training framework for relevance modeling in commercial search. *ArXiv preprint*, abs/2412.01269.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024b. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Shaowei Yao, Jiwei Tan, Xi Chen, Juhao Zhang, Xiaoyi Zeng, and Keping Yang. 2022. Reprbert: Distilling BERT to an efficient representation-based relevance model for e-commerce. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 4363–4371.

Dezhi Ye, Jie Liu, Junwei Hu, Jiabin Fan, Bowen Tian, Haijin Liang, and Jin Ma. 2025. Applying large language model for relevance search in tencent. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5171–5181.

Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly Jr., Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 323–332.

Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2023. A two-stage adaptation of large language models for text ranking. *ArXiv preprint*, abs/2311.16720.

Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, and 1 others. 2025. Onerec technical report. *ArXiv preprint*, abs/2506.13695.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE.

Dawei Zhu, Xiaoyu Shen, Michael Hedderich, and Dietrich Klakow. 2023. Meta self-refinement for robust learning with weak supervision. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1043–1058, Dubrovnik, Croatia.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. *ArXiv preprint*, abs/2503.06034.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 38–47.

Lixin Zou, Weixue Lu, Yiding Liu, Hengyi Cai, Xiaokai Chu, Dehong Ma, Daiting Shi, Yu Sun, Zhicong Cheng, Simiu Gu, and 1 others. 2022. Pre-trained language model-based retrieval and ranking for web search. *ACM Transactions on the Web*, 17(1):1–36.

Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model based ranking in baidu search. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 4014–4022.

A Experimental Details

A.1 Setups

Continual Pre-training. Following prior work (Ye et al., 2025), we first performed continual pre-training to help the model adapt to our specific experimental domain. During this stage, we set the learning rate to 1e-4 for both the Qwen2.5-7B and Qwen2.5-1.5B models.

SFT Training. During SFT, we used a learning rate of 3e-6 with a warm-up strategy applied during the first 10% of the training process, and the training epoch was set to 1. In each iteration of SERM and self-training, we also adopted the same learning rate of 3e-6. Note that we applied identical hyperparameters for both the 7B and 1.5B models. We also experimented with adjusting the learning rate according to model size, but observed no significant improvement in performance. The prompt used for SFT training is shown in Figure 5. For each input, the document included its title, hashtags, and summary.

SERM. In our experiments with SERM, we configured the agents as follows. For the user-feedback agent, we set the dwell-time threshold τ_u to 5 seconds and the model confidence threshold τ_c to 0.4. Note that τ_u was determined based on platform-specific statistics derived from real-world business scenarios, while τ_c was selected empirically as the optimal value, as shown in Figure 3. During sample selection, we set n to 4, meaning that for each query, the sample-mining agents collectively select up to four candidate documents to form the training samples. If the number of documents satisfying the agent's selection criteria exceeds n, we randomly sample four; if it is fewer than n, we use all qualifying documents. For the click-model feedback agent, we set the threshold τ_{cm} to 0.1. For the inner-agreement agent, we generated three reasoning paths for each query–document pair to improve label stability and consistency. The prompt used for this process is provided in Figure 6.

A.2 Datasets

We presented the length distributions of the datasets used in the pre-training and SFT stages in Figure 4. Both the continual pre-training and SFT pipelines followed standard industry practices and did not include any procedures specifically designed to favor SERM. Moreover, the self-training baseline was constructed using the same pre-training and SFT setup, which effectively isolated the effects of these stages from the performance gains observed in our experiments. As shown in Table 1, under an identical training recipe with the Qwen2.5-7B model, SERM consistently outperformed the self-training baseline by 1–2 points in NDCG@4 at Iteration 3. These results demonstrated that the improvements achieved by SERM were not attributable to differences in pre-training or SFT data, but instead stemmed from the proposed self-evolving framework itself. Additionally, we reported the language distribution and dataset statistics for both the training and test sets in Table 4. Notably, the SERM training data were selected by the MSM module, with approximately one million query-document pairs used per iteration. Our evaluation was conducted in a multilingual search environment, which posed additional challenges and further highlighted the robustness of the proposed approach.

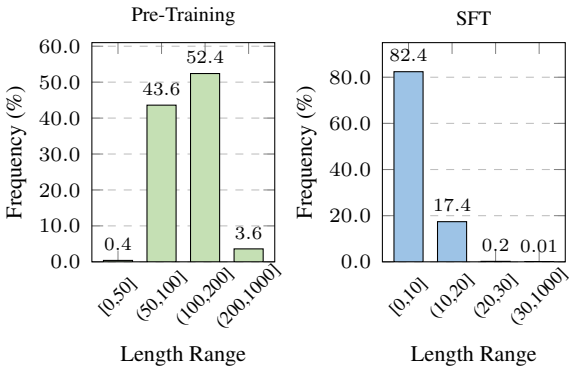


Figure 4: Length distribution of document datasets and queries used in our experiments.

Dataset	Germanic	Romance	Minor Language
	EN DE NL	ES PT FR IT	ID AR JA RU
SFT	1,655,040	1,105,927	840,963
SERM (Iteration 1)	359,249	190,630	140,123
SERM (Iteration 2)	354,551	172,343	159,978
SERM (Iteration 3)	358,090	183,872	160,692
SERM (Iteration 4)	328,506	177,393	144,542
SERM (Iteration 5)	335,001	174,200	160,800
Testing	43,793	23,392	18,262
Testing-v2	57,126	33,391	30,261

Table 4: The language distribution and statistics for the training and test datasets, including data from SFT and self-evolution (SERM), are based on different language families: Germanic, Romance, and Minor Languages, following the classification of He et al. (2024). Note that "Testing-v2" denotes our latest testing set, which incorporates newly collected test cases to evaluate better whether the model continues to self-evolve over time. The construction and curation of the testing-v2 dataset are described in Appendix B.2.

Metric	Absolute Gain	P-value	Unit
User Negative Feedback	-2,416,200	0.0001	counts (number of events)
Change Query Ratio	-2,409,524	0.0023	counts (number of change-query events)
Change Query Ratio (Longtail)	-753,586	0.0015	counts (number of change-query events)
14-Day Retention	+25,130	0.0278	user-days

Table 5: Absolute improvements observed in the online A/B test.

A.3 Evaluation

Inference of Generative Relevance Models. In this work, we employ a probability aggregation approach to derive relevance scores from the generative relevance model. Specifically, given a query–document pair (q, d), the model generates a discrete relevance label token (e.g., $y \in \{0, 1, 2, 3, 4\}$), where the predicted token reflects the model's relevance judgment. To obtain a continuous relevance score that is more informative for downstream ranking, we compute the expectation over the label probabilities as:

$$f(q, d) = \sum_{y \in \mathcal{Y}} y \cdot \Pr_\theta(y \mid q, d) \tag{6}$$

where $\Pr_\theta(y \mid q, d)$ denotes the probability of generating label y under the model parameters θ.

A/B Testing Metrics. We report the following key metrics for evaluating the impact of relevance models in online A/B testing:

- *User Negative Feedback*. This metric captures instances where users provide explicit negative feedback (e.g., reporting irrelevant or unsatisfactory results). A lower value indicates higher immediate user satisfaction.
- *Change Query Ratio*. This metric measures the proportion of cases in which users reformulate or issue a new query for the same information need after the initial search did not meet their expectations. A lower ratio suggests that the system is more effective at fulfilling user intent on the first attempt.
- *14-Day Retention*. This metric tracks whether users continue to return to the platform over a 14-day window. A higher value indicates stronger long-term engagement and improved user loyalty.

In the experimental results, although the absolute percentage improvement appears small, the practical impact is, in fact, substantial due to the extremely large user base involved in our online A/B evaluation. For instance, both the control and treatment groups contain approximately 70 million users. Under such a scale, even a 0.0359% improvement translates to a significant increase in cumulative user activity:

$$0.000359 \times 70{,}000{,}000 = 25{,}130 \text{ user-days} \tag{7}$$

We also present all the absolute improvements observed in our A/B test results in Table 5. From the results, although the percentage appears small, the gain is far from negligible in practice; rather, it represents a meaningful improvement in long-term user engagement within a high-traffic industrial search system. It is also worth noting that such A/B online testing metrics are commonly used in search relevance evaluation, and the corresponding improvements are typically small in absolute terms due to the extremely large user bases involved (Zou et al., 2021; Ye et al., 2025; Zhou et al., 2025).

Side-by-Side Evaluation. In our side-by-side evaluation, the random category refers to a uniformly sampled subset of user queries drawn from the overall production traffic, reflecting the average distribution of common and frequently occurring queries. In contrast, the long-tail category consists of queries that fall into the low-frequency region of the query distribution, *i.e.*, those appearing below a frequency threshold within the same logging period. These long-tail queries are typically rare, emerging, or domain-specific, and are underrepresented in training data, making them substantially more challenging for relevance models to handle. Prior work in information retrieval has similarly emphasized the importance of evaluating models on low-frequency or long-tail queries, as they better reflect robustness and generalization in practical deployments (Ye et al., 2025).

B More Analysis

B.1 Differences from Knowledge Distillation

In this subsection, we discuss how SERM differs from conventional knowledge distillation in two

Method	Germanic			Romance			Minor Language		
	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.
Training with Qwen2.5-7B Model									
CT+SFT	84.74	86.68	55.30	85.61	87.33	50.74	82.02	84.02	52.89
Self-Training									
Iteration 4	84.55	86.63	55.54	85.44	87.26	51.80	82.07	84.03	54.34
Iteration 5	84.44	86.60	55.53	85.42	87.27	51.80	82.01	84.02	54.34
SERM									
Iteration 4	**87.64**	**87.74**	**57.90**	**88.04**	**88.19**	**53.96**	**84.98**	**85.12**	55.49
Iteration 5	87.47	87.68	57.68	87.86	88.18	53.46	84.89	85.08	**55.69**
Training with Qwen2.5-1.5B Model									
CT+SFT	84.59	86.63	54.75	85.99	87.44	50.24	81.75	83.91	51.81
Self-Training									
Iteration 4	84.63	86.65	55.62	85.31	87.23	51.55	81.86	83.97	53.70
Iteration 5	84.70	86.66	55.63	85.46	87.26	51.57	81.80	83.96	53.70
SERM									
Iteration 4	87.32	87.60	57.05	**87.74**	**88.12**	53.39	84.85	85.06	54.58
Iteration 5	**87.43**	**87.64**	**57.22**	87.62	88.08	**53.77**	**84.94**	**85.10**	**55.00**

Table 6: Results from additional SERM iterations 4 and 5.

key aspects. First, *unlike traditional knowledge distillation where a student passively mimics a static teacher on a fixed dataset, SERM acts as an active learner*. The "evolution" is driven by the model's own internal states. Specifically, the MSM uses the model's own uncertainty and disagreement (as shown in Section 3.1.2) to identify where it is failing. In this way, the model itself dictates the curriculum of its learning process. If the model were confident and correct, no evolution would be triggered. Thus, the impetus for improvement is intrinsic, even if the supervision is extrinsic. Second, *while traditional knowledge distillation operates in a static manner, where a fixed teacher transfers knowledge to a student before deployment, our SERM presents a fundamentally different paradigm.* Specifically, unlike knowledge distillation, SERM remains adaptive after deployment and is driven by three forms of dynamism that are intrinsic to industrial search environments: a dynamic environment, dynamic data, and dynamic annotation.

B.2 Results from Additional SERM Iterations

Performance Saturation on the Static Testing Set. Table 6 reports results from SERM iterations 4 and 5 trained on newly collected queries from the most recent month. We observe that performance gains gradually saturate across later iterations, particularly for the self-training baseline, whose metrics remain nearly unchanged from iteration 4 to iteration 5. This suggests that, on this relatively static test distribution, the model has approached the upper bound of achievable performance. Notably, despite this saturation effect, SERM consistently outperforms self-training across all language families and both model scales. Even in later iterations, SERM maintains stable improvements of approximately 1–2 points in NDCG@4 and accuracy, indicating that it continues to provide higher-quality evolutionary signals than standard self-training. These observations naturally raise an important question: *Does the observed saturation indicate that SERM has reached its capacity for further improving relevance models?*

Results on a Newly Collected Testing Set. To further investigate this question, we construct a newly collected testing set, denoted as *testing-v2*, which incorporates more recent and evolving user queries. Specifically, testing-v2 augments the original testing set with newly emerged query cases collected in the most recent month, aiming to better reflect the continuously shifting query distribution in real-world search environments. The language distribution and statistics of the testing-v2 are pre-

Method	Germanic			Romance			Minor Language		
	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.	ND@1	ND@4	Acc.
Training with Qwen2.5-7B Model									
CT+SFT	71.69	71.29	47.22	73.94	73.30	48.94	70.08	70.17	47.42
Self-Training									
Iteration 1	72.07	71.76	47.75	74.02	73.45	49.03	70.29	70.51	47.56
Iteration 2	72.22	72.31	47.82	74.33	73.77	49.31	70.36	70.68	47.65
Iteration 3	72.31	71.99	48.25	74.19	73.54	49.43	70.50	70.69	48.04
Iteration 4	72.46	72.44	48.44	74.31	73.93	50.02	70.76	70.87	48.06
Iteration 5	72.64	72.14	48.47	74.37	73.60	50.12	70.74	70.80	48.50
SERM									
Iteration 1	74.07	73.02	49.96	75.62	75.29	50.62	74.11	74.17	50.48
Iteration 2	75.42	74.10	50.59	75.98	76.87	50.69	74.85	74.41	50.65
Iteration 3	75.43	74.38	50.84	76.58	77.51	50.91	75.10	74.48	50.61
Iteration 4	76.21	74.47	51.15	77.64	77.52	53.54	76.51	75.34	52.76
Iteration 5	**76.67**	**74.57**	**51.33**	**78.04**	**77.76**	**53.56**	**76.63**	**75.88**	**53.28**
Training with Qwen2.5-1.5B Model									
CT+SFT	71.33	70.94	46.65	73.72	72.54	48.69	68.90	70.09	46.53
Self-Training									
Iteration 1	71.94	71.19	47.44	73.84	72.80	48.84	69.50	70.48	47.36
Iteration 2	72.15	72.49	47.52	74.11	73.01	48.62	69.97	70.75	47.45
Iteration 3	72.17	71.36	47.76	73.92	72.96	49.42	69.67	70.75	47.52
Iteration 4	72.26	72.66	47.53	74.06	73.18	48.83	70.16	70.87	48.25
Iteration 5	72.38	71.49	48.14	74.13	73.02	49.70	70.02	70.88	48.01
SERM									
Iteration 1	73.94	73.19	48.70	74.55	76.75	50.95	72.97	72.95	49.86
Iteration 2	74.25	73.49	49.06	74.68	76.78	51.51	73.04	72.98	49.96
Iteration 3	74.49	73.78	49.28	76.30	76.96	51.63	73.09	73.16	50.12
Iteration 4	74.83	73.82	49.87	76.81	77.04	52.44	73.31	74.00	51.08
Iteration 5	**75.24**	**73.83**	**50.32**	**77.49**	**77.09**	**52.75**	**73.81**	**74.30**	**51.59**

Table 7: Results on a newly collected testing set with evolving query distributions.

sented in Table 4. Note that these additional test queries are not selected based on or aligned with the training data used in Iterations 4 and 5, ensuring a fair and unbiased evaluation. We re-evaluate models from Iterations 1 to 5 on testing-v2, and the results are reported in Table 7. Several important observations can be drawn. First, across all language families and both model scales, SERM consistently outperforms the self-training baseline at every iteration, reaffirming the robustness and general effectiveness of the proposed self-evolution framework under distribution shifts. Second, unlike the saturation behavior observed in Table 6, SERM continues to exhibit monotonic or near-monotonic improvements across iterations on testing-v2, particularly in later iterations. This contrast suggests that the previously observed performance plateau is largely attributable to the limited capacity of a fixed test distribution, rather than a degradation or failure of the SERM mechanism itself. In other words, while the model may appear saturated under static evaluation, it continues to acquire new capabilities when assessed against evolving query distributions.

Given a search query and the document textual content, evaluate the relevance to the query and assign a relevance score from 0 to 3. The text contents include the title, hashtags, document summary.
The relevance score should only be selected from 0/1/2/3.

Here is the search query and the textual content.
Query: {query}
Title: {title}
Hashtags: {hashtag}
Summary: {summary}

Figure 5: Template used for training our generative relevance models.

Assume three experts are conducting document relevance judgment; please judge the relevance between the query and the documents.

[Query and document Information]
User query: {query}
Document title: {title}
Document hashtag: {hashtag}
Document summary: {summary}

[Contextual Information]
In the online search system, we have the following contextual information:
- Document click-through rate (CTR): {ctr} — The click-through rate represents the ratio of users who click on the document after viewing it. A higher CTR indicates greater user interest and engagement with the document.
- Document average dwelling time: {dwell_time} — The average dwell time measures the time a user spends on a document after clicking on it. A longer dwell time suggests that the user found the document more relevant and engaging.
- Click model score (0-1): {cm_score} — The click model score estimates the likelihood of a user clicking on a document based on factors such as relevance and user behavior. A higher score indicates a stronger likelihood of the document being clicked.
- Relevance score (0-1): {model_score} — The relevance score is an estimated measure of how well the document satisfies the user's query via a trained relevance model. A higher relevance score indicates a closer match to the query intent.
- Model disagreement (0-3): {disagreement_score} — The model disagreement refers to internal conflicts within the model, where a larger value indicates greater inconsistency in the model's predictions.
- Model uncertainty (0-1): {uncertainty} — The model uncertainty measures the confidence of the model in its predictions. A higher value indicates greater uncertainty.

[Task Definition]
Each expert's task is to generate the relevance score in JSON format. Experts may hold different opinions based on their individual judgments. The JSON object must contain the following keys:
- Rationale: a string, reasoning steps on how to generate the relevance score.
- Score: a score ranging from 0 to 3 by the following *Relevance Definition*.

[Relevance Definition]
- 0: Indicates useless, meaning the doc is worthless to the user.
- 1: Indicates low value, meaning the doc only has a little value and can only satisfy the user's need to a small extent.
- 2: Indicates pass, meaning the doc has at least some reference value, partially meeting user needs.
- 3: Indicates quite perfect, meaning that the doc satisfies the user's needs very well.

[Format Definition]
- Your output must always be a list of JSON objects only. The length of the list is equal to the number of experts.
- Do not use code block markers. Do not explain yourself or output anything else.

Figure 6: Template used for labeling relevance judgments in the MRA module.

Query	tongue piercing
Document	
Title	I'm low-key crying
Summary	Hey, it's been almost three months since I got my tongue piercing. I changed it today like I have a few times before, and it was bothering me. I thought it was because of the new piercing jewelry I put in, but now I can't get it back in at all. Good afternoon. Come by this afternoon and we'll take a look. Bring all the jewelry pieces you have, or any that you know fit you well. What time do you close? We're open until 8:15 PM.
Hashtag	#crying #tonguepiercing
LLM Reasoning #1	
Rationale	The document discusses the author's experience with a tongue piercing, including issues with changing the piercing and seeking advice. This is somewhat relevant to the query 'tongue piercing' as it provides personal insights and experiences related to tongue piercings, but it does not fully satisfy the query as it lacks detailed or instructional content.
Relevance Score	2
LLM Reasoning #2	
Rationale	The document content is related to the query 'tongue piercing' because it talks about a tongue piercing and the author's experience with it. However, it does not provide comprehensive or highly useful information, such as how to get a tongue piercing or care tips, which limits its relevance.
Relevance Score	2
LLM Reasoning #3	
Rationale	While the document mentions a tongue piercing and some issues the author faced, it does not provide substantial or actionable information about tongue piercings. It is somewhat relevant but not highly valuable for someone searching for 'tongue piercing'.
Relevance Score	1

Table 8: We present a case where the LLM generates reasoning and relevance scores for a query–document pair in the MRA module. Three different reasoning paths are provided for the query "tongue piercing" and a document that records a conversation about the author's experience with a tongue piercing issue and seeking related advice.

Query	wild au aughhh
Document	
Title	-wild au-
Summary	cupcakewild au- (long intro) intro:10sec magbago kana uy
Hashtag	#aufornow #enhypenau #aumeaning #wildau #yoyoyoyoyoyo
Human-Annotated Rationale	Analyzing the query 'wild au aughhh', the most important keywords are 'wild' and 'au'. The document's title '-wild au-' and the hashtag '#wildau' directly correspond to these keywords. The term 'au' in the context of TikTok and hashtags like '#enhypenau' often refers to 'Alternate Universe', a popular fanfiction genre. So, the user is likely searching for a 'wild' themed 'Alternate Universe' story or document. This document is explicitly labeled as a 'wild au'. The other hashtags like '#aufornow' and '#aumeaning' further confirm the content is about the 'au' concept. The document is highly relevant and directly addresses the user's search intent.
CT+SFT	0.6193
Self-Training	0.7866 (Iteration 1) $\rightarrow$ 0.6586 (Iteration 2)$\rightarrow$ 0.4854 (Iteration 3)
SERM	0.7378 (Iteration 1) $\rightarrow$ 0.8735 (Iteration 2)$\rightarrow$ 0.9389 (Iteration 3)

Table 9: Case 1 of relevance scores assigned by different models. In contrast to Self-Training, which suffers from performance degradation across iterations, SERM progressively captures the true intent behind the query ("wild au aughhh") and assigns increasingly accurate relevance scores to the document, demonstrating superior query understanding and more effective relevance modeling.

Query	how to prepare garlic and lemon
Document	
Title	Lemon+Garlic
Summary	Lemon + Garlic. Mix lemon with garlic and you will not visit a hospital again. A single cup of this drink is enough to clean your kidneys, skin, and much more. For this powerful drink, we will use a previously washed lemon. Lemon provides a large amount of vitamin C, potassium, and smaller amounts of other vitamins and minerals. Finally, we will use a piece of turmeric.
Hashtag	#powerofherbs #halamanggamot #herbalplants #healthbenefits
Human-Annotated Rationale	The user's query is 'how to prepare garlic and lemon'. The document title is 'Lemon+Garlic' and the summary explicitly mentions mixing lemon with garlic. The summary starts by saying 'Mix lemon with garlic...' and then lists the ingredients needed: 'a previously washed lemon' and 'a piece of turmeric'. Although it doesn't give the exact step-by-step preparation instructions (like chopping, blending, quantities), it clearly indicates that it's a recipe or preparation guide. The document is about preparing a drink with lemon and garlic, which directly aligns with the user's intent to learn how to prepare these two ingredients together. Therefore, the document is highly relevant and likely contains the exact preparation method the user is looking for.
CT+SFT	0.7429
Self-Training	0.7371 (Iteration 1) $\rightarrow$ 0.7059 (Iteration 2) $\rightarrow$ 0.9675 (Iteration 3)
SERM	0.9384 (Iteration 1) $\rightarrow$ 0.9743 (Iteration 2) $\rightarrow$ 0.9839 (Iteration 3)

Table 10: Case 2 of relevance scores assigned by different models. As shown, SERM increasingly assigns higher relevance scores across iterations, demonstrating its ability to capture and align with the query's true intent—identifying that the document provides guidance on preparing a drink with lemon and garlic.