RRADistill: Distilling LLMs' Passage Ranking Ability for Long-Tail Queries Document Re-Ranking on a Search Engine

Nayoung Choi[2†*], Youngjune Lee[1†], Gyu-Hwung Cho[1], Haeyu Jeong[1], Jungmin Kong[1], Saehun Kim[1], Keunchan Park[1], Sarah Cho[1], Inchang Jeong[1], Gyohee Nam[1], Sunghoon Han[1], Wonil Yang[1] and Jaeho Choi[1]
[1]Naver Corporation, [2]Emory University
nayoung.choi@emory.edu, youngjune.lee93@navercorp.com

Abstract

Large Language Models (LLMs) excel at understanding the semantic relationships between queries and documents, even with lengthy and complex long-tail queries. These queries are challenging for feedback-based rankings due to sparse user engagement and limited feedback, making LLMs' relevance ranking ability highly valuable. However, the large size and slow inference of LLMs necessitate the development of smaller, more efficient Small Language Models (SLMs). Recently, integrating ranking label generation into distillation techniques has become crucial, but existing methods underutilize LLMs' capabilities and are cumbersome. Our research, **RRADistill** (**R**e-**R**anking **A**bility **Distill**ation), propose an efficient label generation pipeline and novel SLM training methods for both encoder and decoder models. We introduce an encoder-based method using a Term Control Layer to capture term matching signals and a decoder-based model with a ranking layer for enhanced understanding. Experimental results including A/B testing on NAVER, South Korea's leading search platform, demonstrate effectiveness of our approach in re-ranking for long-tail queries.

1 Introduction

Large Language Models (LLMs), such as ChatGPT (OpenAI, 2022) and GPT-4 (Achiam et al., 2023), have shown remarkable potential across diverse search tasks, including query rewriting (Mao et al., 2023; Dhole and Agichtein, 2024), query and document expansions (Wang et al., 2023a; Mackie et al., 2023; Ma et al., 2023a). As LLMs advance in complex tasks, they also show potential for passage ranking, which involves understanding relationships between queries and multiple documents. Recent studies (Qin et al., 2024; Ma et al., 2023b; Zhuang et al., 2024; Pradeep et al., 2023a) show that instruction-tuned LLMs like GPT-4, Flan-T5 (Chung et al., 2024) and Vicuna (Chiang et al., 2023) effectively handle passage ranking in zero-shot settings. Motivated by these studies, we also conducted zero-shot re-ranking with our in-house LLMs, HyperCLOVA X (HCX) (Yoo et al., 2024) which comprises Large (HCX-L) and Small (HCX-S) variants. We put a query and a list of document snippet texts to HCX, using a Korean translation of list-wise prompt from RankGPT (Sun et al., 2023). We found that HCX-L effectively returns document identifiers in the desired order, excluding low-relevance ones, as depicted in Figure 1. Unlike short-head queries, which are popular and short like keywords, long-tail queries are complex and involve longer, specific phrases (A.1.1). These queries benefit more from relevance than feedback, and semantic than syntactic matching due to their rich semantic content but lack of user feedback. Thus, LLMs' ability to rank complex long-tail queries by relevance is highly valuable.

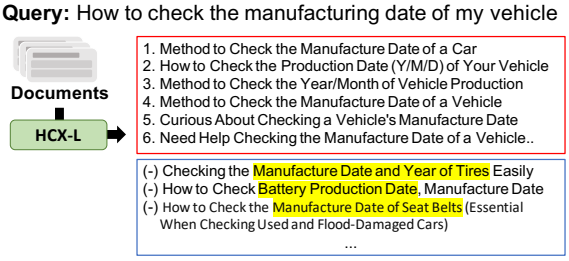

Figure 1: An example of zero-shot inference by HyperCLOVA X (HCX) on retrieved documents. The red box shows HCX's ranked output; the blue box shows excluded documents. Irrelevant parts of excluded documents are highlighted in yellow. The original was in Korean, but translated to English.

However, the slow inference speed challenges the direct use of LLMs in search engine. To address this, we trained a much smaller Language Model (SLM) to retain HCX-L's ranking ability, following a trend known as LLM distillation, similar to RankGPT and TWOLAR (Baldelli et al., 2024).

[†]Both authors contributed equally to this research.
[*]Work done while at Naver.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 627–641
November 12-16, 2024

This involves two stages: 1) Generating ranking label using LLMs, and 2) Training the SLM ranker. When generating ranking label with LLMs, previous studies utilize the list-wise permutation generation method, which inputs a query and a set of documents to LLM and receives an ordered list of document identifiers. However, these approach require sliding windows, which infer multiple times on partial lists due to the prompt length constraints of LLMs, causing a burden. Moreover, previous studies viewed the *missing* phenomenon as a problem, where LLMs fail to include all input documents in the output. However, we observed that in most cases, excluded documents due to *missing* are significantly irrelevant to the query, making it a valuable signal. Consequently, we reframed the *missing* and highlighted its impact. We developed our own label generation pipeline to address these issues, including two key techniques: 1) Pre-rank to filter documents, retaining only those effective to train SLM rankers and bypass the sliding window, 2) Consider *missing* as useful signals, and utilize excluded documents as hard negatives, to train SLM rankers. Our pipeline speeds up labeling and provides compact yet effective training data.

In training SLM rankers, we explored both BERT (Devlin et al., 2019) and GPT (Radford et al., 2019) styles, incorporating our training techniques. For BERT ranker, we integrate a term control layer into the training process to utilize specific term matching signals. For GPT ranker, we developed techniques to effectively utilize classification (whether *relevant* or *irrelevant*) and reasoning (rationale for *relevant* or *irrelevant*) during training, with a light-weight ranking layer. Both rankers incorporate additional training layers, but only specific parts of the model architecture (Encoder plus a classification head for BERT, Decoder plus a dense layer for GPT) are utilized during inference, reducing the burden for service applications.

In this paper, we provide various experiments on our BERT ranker (RRA-BERT) and GPT ranker (RRA-GPT), trained to mimic LLMs' relevance ranking, aiming to improve long-tail search quality. We tested the effectiveness of our methodology through rigorous online and offline evaluations.

2 Methodology

2.1 Label Generation with LLMs

First, we sampled 7,000 long-tail queries from NAVER search logs based on length, complex phrasing and frequency criteria, as detailed in A.1.1. Then, we retrieved 50 documents per query with multiple retrievers of NAVER search engine. Given a query q and retrieved documents $D = [d_1, d_2, ..., d_n]$, we ranked D using our pre-ranker. From pre-ranker ($Ranker_{\text{pre}}$), we obtained the top 10 (D_{top10}) and bottom 10 (D_{bottom10}) documents, and labeled them with HCX-L (Yoo et al., 2024) in a list-wise manner. All document inputs are snippet texts. Figure 2 depicts the overview of our label generation pipeline with LLM. Further details, including the pre-ranker (A.1.2) are described in A.1.

$$\begin{aligned} D' &= Ranker_{\text{pre}}(D) \\ D_{\text{top10}} = D'[:10];\ D_{\text{bottom10}} &= D'[-10:] \\ D_{\text{pre}} &= D_{\text{top10}} \cup D_{\text{bottom10}} \\ D_{\text{ranked}} &= HCX_{\text{L}}(Prompt(q, D_{\text{pre}})) \\ D_{\text{excluded}} &= D_{\text{pre}} \setminus D_{\text{ranked}} \end{aligned} \quad (1)$$

Previous study (Sun et al., 2023) has highlighted the *missing* phenomenon, where LLMs rank only part of the input list, suggesting its frequency varies depending on LLMs. HCX-L also exhibited frequent missing occurrences. However, as shown in Figure 1, we observed that in most cases only relevant documents were included in the output (D_{ranked}), excluding documents with significantly low relevance to the query (D_{excluded}). Hence, we reframed the *missing* as a valuable signal, leveraging excluded documents as hard negatives. Through comparison experiments in Section 3.1.1, training with and without excluded documents, we demonstrated the usefulness of the missing signal. For GPT ranker training, we generated reasoning, which is the rationale for why q and d is *relevant* or *irrelevant*, as described in A.1.3.

2.2 BERT-sytle Distillation: RRA-BERT

Lengthy and complex queries require attention not only to the overall semantic information but also to specific terms that are particularly noteworthy within the query. To address this problem, we propose a novel training approach designed to inject term matching signals between queries and documents as hints into dense representations to effectively enhance performance. BERT-style distillation consists of three components: (1) **Token Selection (TS)** method to select tokens from the document matched to the query. (2) **Term Control Layer (TCL)** that utilizes information of selected tokens as hints for training. (3) **Optimization** that

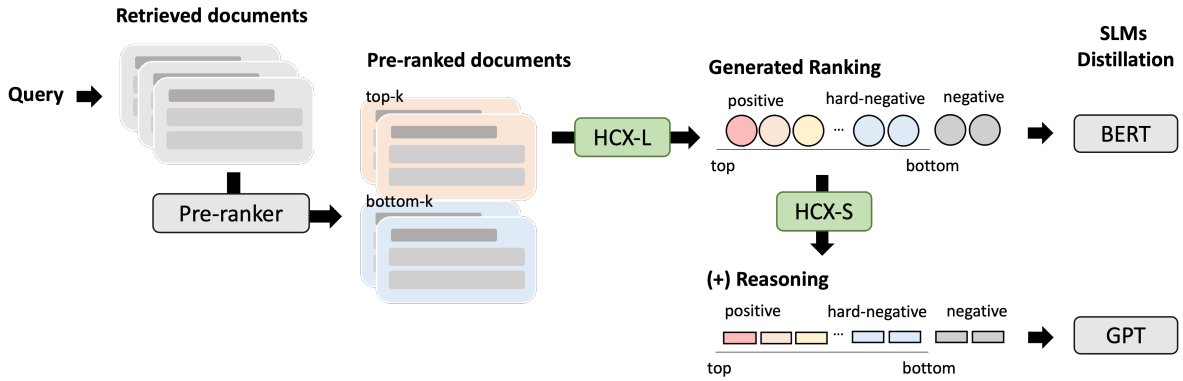

Figure 2: The overview of our label generation pipeline. Negatives are randomly selected from documents totally unrelated to the query.

effectively combines semantic and term information in the training process. The overall structure of BERT-style distillation is in Figure 3.

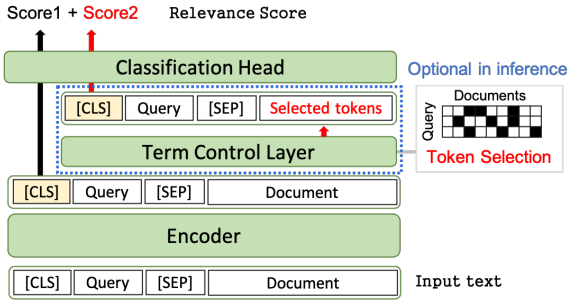

Figure 3: RRA-BERT: The [SEP] token distinguishes the query and the document. The Term Control Layer can be omitted during inference.

2.2.1 Token Selection

We propose a Token Selection (TS) method that captures terms matching signals between the query and the documents as hints, allowing the model to consider both the overall and specific semantics. We select top-k document tokens that match each query token. The process involves identifying and selecting matched tokens $T_{q,d}$ within document d for a given query q using word embeddings ($Embedding_{\text{word}}$), as follows.

$$\begin{aligned} E_q &= Embedding_{\text{word}}(Tokenizer(q)) \\ E_d &= Embedding_{\text{word}}(Tokenizer(d)) \\ Sim_{q,d} &= E_q E_d^T \\ T_{q,d} &= \{\text{Top}_k(Sim_{q,d}[i,:]) | i = 1, \ldots, n_q\} \end{aligned} \quad (2)$$

where n_q is the number of tokens in q. In this process, we select $k \times n_q$ tokens from the document, while excluding duplicate tokens.

2.2.2 Term Control Layer

We propose Term Control Layer (TCL) that effectively integrates the term matching signals into the overall training process. Unlike the overall semantic score (Score1 in Figure 3), TCL utilizes only selected document tokens $T_{q,d}$ to focus on specific tokens. We designed TCL with a multi-head self-attention (Vaswani et al., 2017) mechanism using the last hidden states of encoder as the input, enabling the aggregation of information from each token. The corresponding formula is as follows.

$$\begin{aligned} H_{\text{T}} &= Concat(h_{\text{[CLS]}}, h_q, h_{\text{[SEP]}}, h_{T_{q,d}}) \\ TCL(H_{\text{T}}) &= Attention_{\text{multi-head}}(H_{\text{T}}) \end{aligned} \quad (3)$$

where $h_{\text{[CLS]}}$, h_{q}, $h_{\text{[SEP]}}$ and $h_{T_{q,d}}$ represent the last hidden states of $[CLS]$, the query, $[SEP]$ and $T_{q,d}$, respectively. The number of attention heads is 8.

2.2.3 Optimization

To compute the basic ranking score s_{base} (Score1 in Figure 3), we input the representation $h_{\text{[CLS]}}$ to the classification head (CLF_{head}). Then, we calculate the score s_{TCL} (Score2 in Figure 3) in the same manner, using the TCL-derived $h_{\text{[CLS]}}$ which is TCL$(H_{\text{T}})_{\text{[CLS]}}$ in equation 3. Both calculations share the same CLF_{head}. The final relevance score s is obtained as follows.

$$\begin{aligned} s_{\text{base}} &= CLF_{\text{head}}(h_{\text{[CLS]}}) \\ s_{\text{TCL}} &= CLF_{\text{head}}(TCL(H_{\text{T}})_{\text{[CLS]}}) \\ s &= s_{\text{base}} + \alpha * s_{\text{TCL}} \end{aligned} \quad (4)$$

where α controls the effects of TCL. Finally, given S, a list of outputs s for n documents, we compute the training loss using RankNet (Burges et al., 2005), a pairwise loss function.

$$L = L_{RankNet}(S); \text{ where } S = [s_1, s_2, \ldots, s_n] \quad (5)$$

Our relevance score computation is designed to effectively capture overall semantics with s_{base}, while focusing on specific matching terms with TCL, represented by s_{TCL}. Also, this design improves s_{base} by naturally integrating the term signal into its computation. Therefore, we can remove the TCL during inference to reduce the deployment burden without degrading performance. A detailed analysis is provided in Section 3.1.1.

2.3 GPT-sytle Distillation: RRA-GPT

Existing GPT and T5 (Raffel et al., 2020) rankers primarily use decoder output as a relevance score, calculating from the decoder logits of either the first generated token or the generated target tokens (e.g., *Yes* or *No*). To optimize relevance scores, training tasks typically fall into two types: classification, as implemented in MonoT5 (Nogueira et al., 2020), and ranking, exemplified by RankT5 (Zhuang et al., 2023), TWOLAR, and RankGPT. Furthermore, ExaRanker (Ferraretto et al., 2023), a T5-based ranker, enhances ranking performance by training to generate explanations for *relevant* or *irrelevant*. In this paper, we explored which task combinations help GPT ranker training and whether reasoning enhances the ranking performance.

In previous studies (Sun et al., 2023; Zhang et al., 2023a; Pradeep et al., 2023b), it was surprising to see that despite GPT's much larger parameter size, its performance often matched or fell behind that of BERT and T5 rankers, which suggests GPT lacks a dedicated input encoding(=understanding) module. To address this, we enhanced GPT ranker with a dense layer, which we call a ranking layer. Selecting the appropriate input for this layer is critical, akin to [CLS] token in BERT, to effectively represent the (q, d) relationship. We tested token embeddings from both the input text and the generated texts, <|Response|> and <|Reason|> respectively, as input to the ranking layer. Our final GPT ranker is depicted in Figure 4.

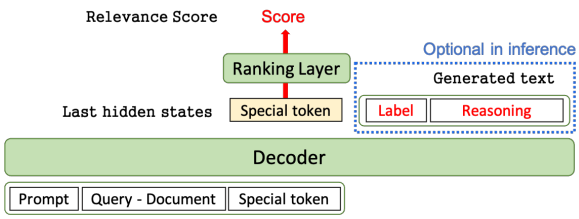

Figure 4: RRA-GPT: The special token here is <|Response|>. The label and reasoning generation can be omitted during inference.

During training, we jointly train relevance ranking, label generation (<|Relevant|> or <|Irrelevant|>), and reasoning. In inference, only the ranking layer is used, without any label generation. The format of our training prompt is shown in Figure 5, but during inference, the prompt includes up to the "<|Response|>:" part only.

```
You are an intelligent assistant that determines the relevance between a
search query and a document. When given a [Search Query] and a
[Document], decide if the document is relevant to the search query.
Respond with "Relevant" or "Irrelevant"

[Search Query] {{query}}
[Document] {{snippet text}}

<|Response|>: {{label}}
<|Reason|> {{reasoning}}
```

Figure 5: Training data format: Replace the red sections $\{\{query\}\}, \{\{snippet\ text\}\}, \{\{label\}\}, \{\{reasoning\}\}$ as needed.

We used significantly small GPT as a backbone for ranker. The backbone has been instruction-tuned, including a task that generates label for a (q, d) pair as either *relevant* or *irrelevant*. We added a total of four special tokens: <|Relevant|>, <|Irrelevant|>, <|Response|> and <|Reason|>, to distinguish them from tokens in q and d during model training. Each special token is initialized with corresponding token embedding; e.g., <|Relevant|> is initialized with "Relevant".

2.3.1 Ranking layer

The relevance score s for a (q, d) pair is obtained through a dense layer as follows.

$$\begin{aligned} X &= Tokenizer(Prompt(q, d)) \\ H &= Decoder(X) \\ h_{<|\text{Resp.}|>} &= H[-1, i];\ \text{where } i \text{ is the index of } <|\texttt{Response}|> \text{ in } X \\ s &= Dense(h_{<|\text{Resp.}|>});\ \text{where } d_{\text{in}} = d_{\text{hidden}}, d_{\text{out}} = 1 \end{aligned} \tag{6}$$

where H refers to all layers of hidden states, and $h_{<|\text{Resp.}|>}$ is the last hidden state of a special token. The *Dense* layer has an input dimension equal to the model's hidden size and outputs a single relevance score. Given a list of outputs s for n documents, where $S = [s_1, s_2, \ldots, s_n]$, the calculation of the loss $L_{\text{RankNet}}(S')$, where $S' = \text{MinMaxScaling}(S)$, follows the BERT ranker training formula as in Equation 5.

2.3.2 Ranking with classification & reasoning

We propose leveraging the decoder's generative abilities for ranking layer training. By simultaneously training the decoder to generate labels and reasoning, we enhance the ranking layer. Label

generation, akin to classification, is as follows.

$$
\begin{aligned}
z &= LM_{\text{head}}(Decoder(X)) \\
z_{\text{rel}} &= z[k];\ \text{where } k \text{ is the token id of } \texttt{<|Relevant|>} \\
z_{\text{irrel}} &= z[j];\ \text{where } j \text{ is the token id of } \texttt{<|Irrelevant|>} \\
p_{\text{rel}} &= \frac{e^{z_{\text{rel}}}}{e^{z_{\text{rel}}} + e^{z_{\text{irrel}}}}, \quad p_{\text{irrel}} = \frac{e^{z_{\text{irrel}}}}{e^{z_{\text{rel}}} + e^{z_{\text{irrel}}}} \\
L_{\text{clf}} &= -[y \log(p_{\text{rel}}) + (1-y) \log(p_{\text{irrel}})]
\end{aligned} \tag{7}
$$

The LM_{head} takes input from hidden layers and generates token probabilities over vocabulary autoregressively. The logit z of the first generated token is always the logit of either <|Relevant|> or <|Irrelevant|>, as training progresses. To infer the class, it is calculated as 1 if $p_{\text{rel}} > p_{\text{irrel}}$ else 0 (1=relevant, 0=irrelevant). The final loss including the generation loss is as follows.

$$
\begin{aligned}
X &= Tokenizer(Prompt(q, d, label[, reasoning])) \\
L_{\text{gen}} &= -\sum_{t=1}^{\text{len}(X)} \log p_{\text{model}}(x_t | x_1, x_2, \ldots, x_{t-1}) \\
L &= L_{\text{gen}} + L_{\text{RankNet}} + L_{\text{clf}}
\end{aligned} \tag{8}
$$

where p_{model} represents the probability that GPT generates the token x_t given the preceding tokens.

3 Experiment

We tested the effectiveness of our label generation and training method, which leverages BERT and GPT structures, alongside following baselines, HCX-L (Yoo et al., 2024), BM25 (Robertson et al., 2009), MonoBERT (Nogueira and Cho, 2019), MonoT5 (Nogueira et al., 2020) and RankGPT (Sun et al. 2023). MonoBERT and MonoT5 utilize our labeled dataset for training. RankGPT's training labels are generated using sliding windows, without our pre-ranking process. While exact parameter sizes of backbones are undisclosed, they follow the order: T5 (small) < BERT << GPT < T5 (large), all of which are below 1 billion parameters. For evaluation, we used our custom NAVER testset along with Korean-translated public testsets for passage re-ranking: MS MARCO (Bajaj et al., 2018), MIRACL (Zhang et al., 2023b), DL19 (Craswell et al., 2020), and DL20 (Craswell et al., 2021). More detailed settings about dataset, metrics, baselines, and implementation are outlined in A.2.

3.1 Results

Table 1 presents the overall performance. **RRA-BERT** excels on our long-tail query testset (NAVER) and matches or surpasses other baselines on public testsets, even outperforming the larger MonoT5. Despite its smaller size, **RRA-GPT** also shows significant improvement, underscoring the effectiveness of our training methods. Moreover, all models trained on our dataset perform comparably to HCX-L, demonstrating the impact of our label generation pipeline. HCX-L relatively underperformed in DL19 and DL20, where many documents necessitated the use of sliding windows (see Table 5). This may be attributed to the finding (Zhang et al., 2023a) that relevant documents are *trapped* in the local block and fail to propagate to the next window. This issue may also clarify why the first-retrieval stage greatly influences LLM reranking (Sun et al., 2023), supporting the efficacy of our window-avoiding labeling approach. Our approach has led to a small efficient model that particularly excels at handling complex queries while also performing well with general queries.

3.1.1 RRA-BERT

We provide an ablation study and analysis of RRA-BERT addressing three questions: **(1) The impact of our label generation method**, **(2) The effectiveness of Token Selection (TS) and Term Control Layer (TCL)**, and **(3) Inference efficiency**. The experimental results are presented in Table 2. **First**, despite having the same architecture, RankGPT (bert) significantly underperforms MonoBERT and RRA-BERT trained on our dataset, confirming the effectiveness of our label generation pipeline. In addition, an ablation study that excludes LLMs' *missing* documents from the training data (`w/o missing`) showed a significant drop in performance, demonstrating that the LLMs' *missing* phenomenon is a useful signal. **Second**, training with TS and TCL (`w/ TS+TCL`) enhances overall performance while reducing the standard deviation without DL20. TS and TCL contributes to both performance improvement and stable training. **Third**, training with TS and TCL but removing TCL during inference (`infer w/o`) did not cause performance degradation. At the same time, removing TCL reduced the inference time by 5.58%. This indicates that TCL enhances s_{base} in Equation 4 by naturally integrating the term signal into its computation. Therefore, we can remove the TCL during inference without degrading performance. As part of our qualitative evaluation, we provide real-world examples of long-tail query ranking results using RRA-BERT in A.4. These examples demonstrate

Table 1: Performance comparison. Best scores per dataset are in **bold**, with second best scores underlined.

	NAVER		MS MARCO		MIRACL		DL19		DL20	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10	nDCG@5	nDCG@10	nDCG@5	nDCG@10	nDCG@5	nDCG@10
BM25	0.427	0.520	0.418	0.524	0.473	0.568	0.350	0.396	0.277	0.284
BERT (naive)	0.535	0.655	0.492	0.567	0.671	0.740	0.584	0.601	0.388	0.419
GPT (vanilla)	0.376	0.473	0.387	0.501	0.323	0.445	0.266	0.307	0.204	0.226
MonoBERT	0.639	0.757	0.533	0.600	**0.696**	**0.759**	0.656	**0.662**	**0.565**	0.560
MonoT5 (large)	0.650	0.759	0.520	0.589	0.668	0.739	0.633	0.652	0.560	**0.565**
RankGPT (bert)	0.589	0.696	0.446	0.542	0.623	0.688	0.557	0.565	0.431	0.434
RankGPT (gpt)	0.432	0.535	0.363	0.487	0.284	0.415	0.295	0.327	0.180	0.201
HCX-L (zero-shot)	-	-	0.523	0.595	0.686	0.733	0.621	0.620	0.480	0.480
RRA-BERT (ours)	**0.655**	**0.776**	**0.543**	**0.607**	0.671	0.743	**0.667**	0.658	0.546	0.536
RRA-GPT (ours)	0.620	0.735	0.491	0.548	0.567	0.660	0.521	0.548	0.417	0.421

Table 2: Ablation study on RRA-BERT (nDCG@5 w/ standard deviation). Best scores per dataset are in **bold**.

	NAVER	MS MARCO	MIRACL	DL19	DL20
`w/o missing`	0.617 (± 0.006)	0.532 (± 0.010)	0.627 (± 0.041)	0.616 (± 0.026)	0.537 (± 0.013)
`w/o TS+TCL`	0.645 (± 0.006)	0.539 (± 0.006)	0.648 (± 0.045)	0.658 (± 0.020)	0.544 (± 0.010)
`w/ TS+TCL`	**0.655** (± 0.001)	**0.543** (± 0.004)	0.671 (± 0.019)	**0.667** (± 0.010)	0.546 (± 0.024)
`infer w/o`	0.654 (± 0.001)	**0.543** (± 0.004)	**0.674** (± 0.019)	0.664 (± 0.012)	**0.550** (± 0.021)

that the model effectively handles long-tail query ranking while still benefiting from improved inference efficiency.

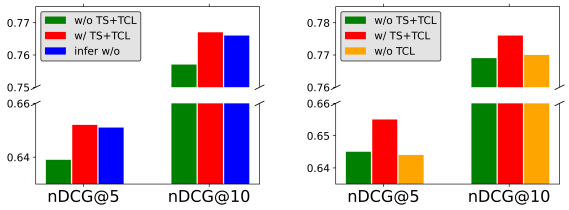

Figure 6: Generalizability study of the TS and TCL with MonoBERT (Left) and effectiveness study of TCL using RRA-BERT (Right) on the NAVER testset.

We conducted further experiments to validate the generalizability and effectiveness of our approach, as shown in Figure 6. Testing on MonoBERT, we observed performance gains with TS and TCL (`w/ TS+TCL`) and no decline without TCL during inference (`infer w/o`), similar to the previous results. This suggests our method reliably enhances performance across ranking models. Additionally, to assess TCL's effectiveness, we input selected document tokens into the encoder, bypassing TCL (`w/o TCL`), and calculated `Score2` in Figure 3 without TCL. The results showed no improvement over `w/ TS+TCL`, confirming the effectiveness of TCL in integrating term signals during training.

3.1.2 RRA-GPT

Here we explore three questions: **(1) The most effective training task combinations**, **(2) The impact of the ranking layer** and whether input or generated tokens are preferable, and **(3) The influence of reasoning on ranking training**. The findings are summarized in Table 3. **First**, models trained with all three tasks: classification (`clf`), ranking (`rank`), and generation (`gen`), showed the best ranking performance. However, adding `clf` or `rank` task individually had no effect in our experiments. **Second**, training a ranking layer was effective, and using the input token embedding `<|Response|>` (abbr. `<|Resp.|>`) was better than using the generated one `<|Reason|>` (abbr. `<|Rsn.|>`), for the input. **Third**, adding the ranking layer greatly improves performance with reasoning, while without it, reasoning worsens ranking, a consistent trend across all our experimental units. We found that simply adding reasoning without extra training layer to GPT does not enhance performance. Undoubtedly, even if reasoning does not directly improve ranking performance, it remains valuable for explainable ranking. Given the challenges of interpreting results from neural models, obtaining explanations for the output of the ranker (i.e., why q and d are relevant or not) is crucial.

Moreover, we observed that jointly training the ranking layer alongside label and reasoning generation yields substantial advantages. Our best model significantly outperformed the `rank only` model shown in Table 3, which was solely trained to optimize relevance scores (`Score` in Figure 4), without label and reasoning generation (`Label`, `Reasoning` in Figure 4). This improvement shows the efficacy of our training approach, maintaining simplicity in inference. Furthermore, the ranking layer accelerates learning. Our best model converges in half the steps compared to the second-best model, which does not use a ranking layer. Specifically, the average training convergence steps were $5666.67(\pm 2624.67)$ and $11333.33(\pm 1885.62)$, re-

*The relevance score s, calculated without a ranking layer, is the same as **GPT** (vanilla), as detailed in A.2.2.

Table 3: Ablation study on RRA-GPT (nDCG@10 w/ standard deviation). Best scores per dataset are in **bold**, with second best scores underlined.

		NAVER	MS MARCO	MIRACL	DL19	DL20
w/o ranking layer						
Task	**Reasoning**					
gen only	False	0.657 (± 0.038)	0.528 (± 0.027)	0.511 (± 0.080)	0.420 (± 0.104)	0.259 (± 0.060)
(+) clf	False	0.571 (± 0.110)	0.523 (±0.030)	0.489 (± 0.115)	0.418 (± 0.090)	0.283 (± 0.070)
(+) rank	False	0.557 (± 0.126)	0.518 (± 0.031)	0.489 (± 0.073)	0.436 (± 0.089)	0.283 (± 0.053)
(+) rank + clf	False	0.706 (± 0.001)	**0.559** (± 0.008)	0.656 (± 0.027)	0.546 (± 0.006)	0.371 (± 0.014)
	True	0.551 (± 0.003)	0.473 (± 0.000)	0.470 (± 0.005)	0.324 (± 0.010)	0.204 (± 0.007)
w/ ranking layer						
Input	**Reasoning**					
<|Resp.|>	False	0.624 (± 0.136)	0.548 (± 0.033)	0.596 (± 0.116)	0.485 (± 0.049)	0.361 (± 0.084)
	False (rank only)	0.647 (± 0.049)	0.533 (± 0.024)	0.510 (± 0.055)	0.431 (± 0.058)	0.278 (± 0.046)
	True	**0.735** (± 0.011)	0.548 (± 0.034)	**0.660** (± 0.030)	**0.548** (± 0.015)	**0.421** (± 0.007)
<|Rsn.|>	True	0.650 (± 0.071)	0.540 (± 0.032)	0.620 (± 0.058)	0.446 (± 0.071)	0.314 (± 0.073)

spectively.

3.1.3 Serving

We compared response latency and ranking performance according to model types and sizes in Figure 7. All results were obtained using a single A100 GPU and model sizes follow the order: T5(small) < BERT < T5(large). As RRA-BERT shows the best ranking performance with reasonable speed we chose it as our final model and successfully deployed using TensorRT-LLM* in real-world scenarios. More details are described in A.3.

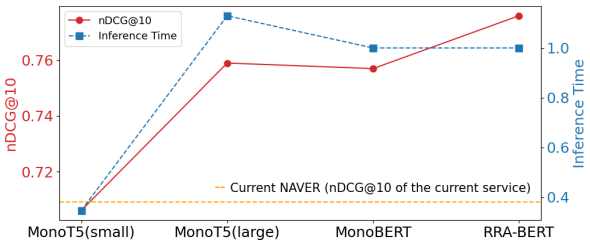

Figure 7: Comparison of inference time (ratio) and nDCG@10 across four models. The yellow line represents the nDCG@10 performance of the current NAVER service.

3.2 A/B Testing

We conducted online and offline A/B tests comparing search results ranked by RRA-BERT with the current search results of NAVER search engine for long-tail queries. In the 7-day online A/B testing, RRA-BERT increased CTR by 5.63%, top-1 document clicks by 5.9%, and dwell time (the duration of time spending on a search result) by 7.97%.

*https://github.com/NVIDIA/TensorRT-LLM

Additionally, we conducted human evaluations by sampling 2,000 queries and scraping search results from both our ranker and the existing one. Human evaluators rated each search result on a scale of 1 to 5, while the ranker remains undisclosed. Compared to the existing ranker, ours achieved an average score increase of 1.41%, with a significant 6.95% boost for top-ranked documents. These results align with the findings of the online A/B tests, demonstrating our ranker's effectiveness in ranking relevant documents at the top of search results.

4 Conclusion

In this paper, we present an efficient label generation pipeline using LLMs that utilizes a pre-ranker to select effective documents and leverages LLMs' *missing* phenomenon as a useful signal. We also present effective training methods to capture long and complex query-document relevance in both BERT and GPT. However, only key parts of the model structure are employed during inference, minimizing the service deployment burden. Through extensive experiments, including A/B testing on NAVER Search, we demonstrate the effectiveness of our approach for long-tail queries re-ranking.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Davide Baldelli, Junfeng Jiang, Akiko Aizawa, and Paolo Torroni. 2024. Twolar: a two-step llm-augmented distillation method for passage reranking. In *European Conference on Information Retrieval*, pages 470–485. Springer.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan

Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *CoRR*, abs/2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

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.

Kaustubh D Dhole and Eugene Agichtein. 2024. Genqrensemble: Zero-shot llm ensemble prompting for generative query reformulation. In *European Conference on Information Retrieval*, pages 326–335.

Fernando Ferraretto, Thiago Laitz, Roberto Lotufo, and Rodrigo Nogueira. 2023. Exaranker: Synthetic explanations improve neural rankers. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2409–2414.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Guangyuan Ma, Xing Wu, Peng Wang, Zijia Lin, and Songlin Hu. 2023a. Pre-training with large language model-based document expansion for dense passage retrieval. *arXiv preprint arXiv:2308.08285*.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023b. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*.

Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2026–2031.

Kelong Mao, Zhicheng Dou, Fengran Mo, Jiewen Hou, Haonan Chen, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1211–1225.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.

OpenAI. 2022. Chatgpt.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. 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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on*

Empirical Methods in Natural Language Processing, pages 14918–14937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Liang Wang, Nan Yang, and Furu Wei. 2023a. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023b. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Kang Min Yoo, Jaegeun Han, Sookyo In, Heewon Jeon, Jisu Jeong, Jaewook Kang, Hyunwook Kim, Kyung-Min Kim, Munhyong Kim, Sungju Kim, et al. 2024. Hyperclova x technical report. *arXiv preprint arXiv:2404.01954*.

Xinyu Zhang, Sebastian Hofstätter, Patrick Lewis, Raphael Tang, and Jimmy Lin. 2023a. Rank-without-gpt: Building gpt-independent listwise rerankers on open-source large language models. *arXiv preprint arXiv:2312.02969*.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023b. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 358–370.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313.

A Appendix

A.1 Details of Label Generation with LLMs

A.1.1 Long-tail query sampling

The NAVER search engine has predominantly handled short keyword-based queries, which constituted the majority of incoming queries as shown in Figure 8. Due to the high volume of popular short queries (short-head), ranking based on query-specific feedback features was effective, with feedback features often proving more significant than relevance features. However, this approach does not apply well to long-tail queries which are long and complex with lower user engagement which is depicted in Figure 9. Individual long-tail queries lacks sufficient user feedback, necessitating a reliance on relevance-based ranking. Since most of the incoming queries were short-head queries that are brief and simple, lacking contextual information, the combination of syntactic matching (like BM25) and feedback features is effective at scale. However, for long-tail queries, relevance is more effective than feedback and semantic matching is more effective than syntactic matching. This is because they lack user feedback but contain rich semantic information.

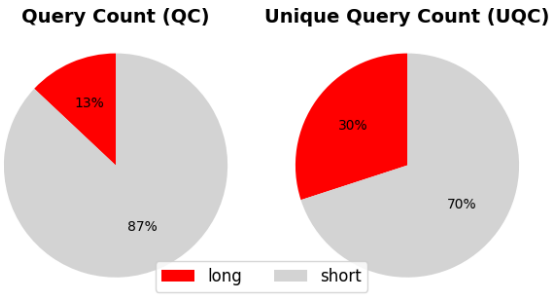

Figure 8: Proportion of long and short queries in our search engine's daily search logs. Queries with a length greater than n are considered long, while others are considered short, based on internal criteria.

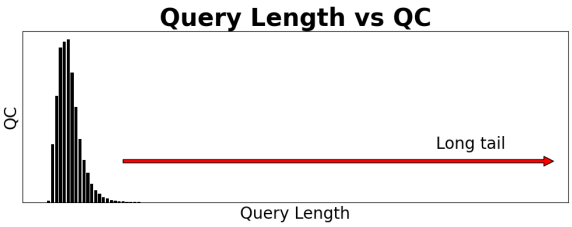

Figure 9: Aggregation of QC based on query lengths from daily search logs on our search engine, showcasing a long-tail distribution. Short queries with high QC are short-head, while long queries with low QC are long-tail.

Now, with the capability of LLMs to understand the relevance between long and complex queries and documents, it has become possible to easily create high-quality training data targeted at long-tail queries. Through LLM distillation, it is now feasible to develop small semantic relevance models that mimic LLMs, thereby improving satisfaction with

long-tail queries that were previously deemed less important. While individual long-tail queries may not be popular, relatively high UQC suggests that users have their own long-tail queries, making improvements to these query search results meaningful. To create specific training sets aligned with our objectives, we sampled queries that were lengthy, complex, infrequently searched based on internal criteria, corresponding to the tail part in Figure 9.

A.1.2 Pre-ranking

Pre-ranking is a crucial step in our ranking label generation pipeline, where all retrieved documents are ranked before being inputted into the LLM. Our pre-ranker, which is BERT-based, was trained on a small dataset of 1,000 queries and corresponding document snippets crawled from NAVER search pages, employing our label generation approach. As several studies (Wang et al., 2023b; Qin et al., 2024; Sun et al., 2023) have shown that the order of inputs significantly influences the performance of LLMs, we initially rank the documents using pre-ranker before inputting them into LLMs. Any rankers can be served as a pre-ranker, and the performance of ours can be found in Table 4.

Table 4: The performance of our pre-ranker measured by nDCG@10

NAVER	MS MARCO	MIRACL	DL19	DL20
0.733	0.567	0.653	0.585	0.493

A.1.3 Reasoning generation

To train the GPT ranker, we utilized HCX-S to create reasoning, which explains the basis for considering q and d as either *relevant* or *irrelevant*.

$$label = \textit{irrelevant} \text{ if } d \in D_{\text{excluded}} \text{ else } \textit{relevant}$$
$$reasoning_{(q,d,label)} = HCX_{\text{S}}(Prompt(q, d, label)) \quad (9)$$

Generating reasoning along with HCX-L list-wise labeling results in either grouping multiple documents together for explanation or omitting reasoning for D_{excluded}. Therefore, we opted for point-wise reasoning generation using HCX-S, a small model of HCX that provides adequate reasoning. Here we utilized a simple prompt like "Explain why the given q and d pair is classified as $label$ (= *relevant* or *irrelevant*)."

A.2 Experimental settings

A.2.1 Datasets & Metrics

The method of constructing training data is in Section 2.1. For evaluation, we set aside 10% of this data. To ensure the robust performance of our models, we also evaluated on public testsets for passage re-ranking: MS MARCO (Bajaj et al., 2018), MIRACL (Zhang et al., 2023b), DL19 (Craswell et al., 2020), and DL20 (Craswell et al., 2021). The statistics of testsets are in Table 5. Since MS MARCO, DL19, and DL20 are in English, we translated them into Korean using NAVER Papago*, which is a Korean machine translation service provided by NAVER. MIRACL is a multilingual dataset, so we directly used its Korean version. We used nDCG (Järvelin and Kekäläinen, 2002) as the evaluation metric, widely used for measuring ranking performance.

Table 5: The number of queries and documents in five testsets.

Count	Ours	MS MARCO	MIRACL	DL19	DL20
Query	700	9,650	213	43	54
Document (avg.)	20	8.20	14.35	107.26	105.30

A.2.2 Baselines

We include the following baselines. MonoBERT and MonoT5 are trained on the dataset created by our label generation pipeline. For RankGPT training labels, we employed sliding windows on D, following the paper (Sun et al., 2023), with a window size of 20 and a step size of 10, without our pre-ranking process.

- **HCX-L** (Yoo et al., 2024): With our teacher model HCX-L which is an instruction-tuned LLM including diverse Korean data, we utilized the list-wise ranking prompt from RankGPT (Sun et al., 2023).
- **BERT** (naive): Before tuning our backbone BERT, we rank based on the cosine similarity of embeddings for q and d.
- **GPT** (vanilla): Before tuning our backbone GPT, we rank using the relevance score $p_{\text{rel}} - p_{\text{irrel}}$ (see Equation 7, changing `<|Relevant|>` to *Relevant* and to `<|Irrelevant|>` to *Irrelevant*).

*https://papago.naver.com/

- **BM25** (Robertson et al., 2009): A probabilistic score for (q, d), calculated from term frequency and inverse document frequency.
- **MonoBERT** (Nogueira and Cho, 2019): A point-wise trained classification model for determining (q, d) relevance, based on BERT.
- **MonoT5** (Nogueira et al., 2020): A T5-based ranker, also employing a classification based approach.
- **RankGPT** (Sun et al., 2023): BERT and GPT-based ranker trained in a pair-wise manner to mimic list-wise LLM ranking.

A.2.3 Implementation details

We used internally pre-trained small Korean LMs, specifically RoBERTa (Liu et al., 2019) for RRA-BERT and GPT (Radford et al., 2019) for RRA-GPT. The same backbones were also used for MonoBERT and RankGPT in baseline experiments. Additionally, we used an in-house T5 backbone for MonoT5. For hyperparameters, the optimizer used for all models is AdamW (Loshchilov and Hutter, 2017), with a learning rate of 1e-5. We trained each model three times and reported the averaged performance. Training data was randomly split into a 9:1 ratio for training and validation. BERT was validated every 300 steps, GPT every 1,000 steps. Early stopping occurred if performance didn't improve for five consecutive validations. The generated ranking labels were used in training as follows:

$$\begin{aligned} 2 - i * 0.1 \text{ for } d_i \in D_{\text{ranked}} \\ 0.2 - (j+1) * 0.01 \text{ for } d_j \in D_{\text{excluded}} \\ 0 \text{ for } d_{\text{neg}}; \text{ where } d_{\text{neg}} \text{ are negatives with a size of 3} \end{aligned} \tag{10}$$

To clarify, j is randomly assigned, since there is no order in D_{excluded}. For MonoBERT and MonoT5 training, which utilize a classification task framework, we label D_{ranked} as relevant documents and $D_{excluded}$ as irrelevant ones. In addition, for TCL training of RRA-BERT, we set k, the number of selected document tokens per query token, to 3 for token selection (Section 2.2.1), and α to 0.3 as the hyperparameter for combining TCL loss (Section 2.2.3).

A.3 Serving details

First of all, despite using a small-sized GPT, its speed and throughput were not as good as BERT and T5, making it difficult to use it for search engine, which demand high Queries Per Second (QPS). With RRA-BERT, we measured performance degradation and QPS based on floating point format. Compared to float32 used during training, using bfloat16, performance was 99.8%, and with fp8, it was 99%, both offering an 18% QPS advantage. Consequently, we selected bfloat16 for serving and successfully deployed to the service.

A.4 Real-world qualitative examples

We provide a few examples of long-tail queries ranked by our final model, RRA-BERT. The original query and document were in Korean and have been translated into English; the document refers to the [title] and snippet. Table 6 illustrates two queries where a single character change completely alters the meaning: "8개월 강아지가 잠만자요" *(8-month-old dog **only** sleeps)*" vs "8개월 강아지가 잠안자요" *(8-month-old dog **doesn't** sleep)*. This example shows how RRA-BERT successfully distinguishes between the two semantically different queries despite their minimal character variation. Additionally, Table 7, 8 and 9 present ranking results for single long-tail queries, demonstrating how our model ranks relevant documents at the top by capturing important term signals while still considering the overall semantic context.

	Query	
	8개월 강아지가 잠만자요 *(8-month-old dog only sleeps)*	**8개월 강아지가 잠안자요** *(8-month-old dog doesn't sleep)*
Rank	**Document**	
1	[갑자기 잠만 자는 강아지?] 8개월 비숑인데요... 성격 활발하고 집에서도 뛰어다니는데 며칠 전부터 계속 잠만 자네요... *[Suddenly sleeping all the time?] I have an 8-month-old Bichon... It's very active but has been sleeping constantly for a few days...*	[우리 강아지가 밤에 안자요] 강아지 8개월인데, 저녁에 계속 놀아달라고 하고 너무 활발해요... *[Our dog doesn't sleep at night] Our 8-month-old dog keeps playing all evening and is super energetic...*
2	[강아지가 밥도 안 먹고 잠만 자요] 강아지가 밥도 안 먹고 잠을 많이 자는데, 어디 아픈 걸까요? 치와와이고 8개월 되었어요... *[Dog isn't eating and only sleeping] My 8-month-old Chihuahua isn't eating and sleeps all the time. Could it be sick?*	[강아지가 밤에 잠을 안자요! 원인과 대처법] 강아지가 밤새도록 울면 보호자도 지치기 쉬운데... *[The dog isn't sleeping at night! Causes and solutions] If a dog cries all night, it can exhaust its owner...*
3	[8개월 강아지 잠이 원래 이렇게 많나요?] 하루 20시간씩 자요... 스트레스 때문일까요? 병원 다녀왔는데... *[Is it normal for an 8-month-old dog to sleep this much?] It sleeps 20 hours a day... Could it be stress? We've visited the vet...*	[새끼 강아지가 잠을 안자요] 새끼 강아지가 밤에 자지 않아서 보호자의 수면을 방해해요... *[Puppy not sleeping] My puppy doesn't sleep at night and disturbs the owner's sleep...*
4	[강아지가 잠만 자는 이유가 뭘까요?] 강아지가 하루 종일 잠만 자는데, 어디 아픈 걸까요? 밥도 잘 안 먹고... *[Why is my dog only sleeping?] My dog is sleeping all day long. Could it be sick? It doesn't eat well either...*	[아기 강아지 수면시간] 강아지가 잠을 너무 많이 자거나 너무 적게 자서 보호자가 걱정할 수 있어요... *[Puppy sleep time] A puppy sleeping too much or too little can make the owner worry...*

Table 6: Document ranking results for two queries with a one-character difference that completely changes their meaning, using our RRA-BERT model.

Query = 80대 요관암 말기 암 항암치료 *(Chemotherapy treatment for terminal stage ureteral cancer in people in their 80s)*	
Rank	**Document**
1	[아버지가 요관암 말기 판정 받으셨어요] 아버지가 올해 만 71세이신데 요관암 말기 판정 받으셨어요. 참전용사이셔서 집근처 중앙보훈병원에서 검사받고 진료 받으셨는데 뼈와 임파선까지 전이가 되어 수술이 불가하다고 하여 항암치료 상담하려고 3월 26일 혈액종양과 진료 예약해놓은 상태인데 차병원 혈종과가 잘 봐주신다고 하여 차병원 천재경 교수님 진료를 3월19일 예약해놓은 상황이에요. 근데 제가… *[My father was diagnosed with terminal ureter cancer] My father is 71 years old this year and was diagnosed with terminal ureter cancer. He is a war veteran and received tests and care at the Central Veterans Hospital near our house. However, since the cancer has spread to the bones and lymph nodes, surgery is not an option. We have scheduled a consultation for chemotherapy on March 26th with the hematology-oncology department, and we also made an appointment with Dr. Cheon Jae-kyung at Cha Hospital for March 19th for another consultation... But I...*
2	[아버지 요관암 말기] 저희 친정 아버지가 2주전 요관암 말기 진단을 받으셨어요. 지방 대학병에서는 항암도 어렵다하여 신촌세브란스 최영득교수한테 며칠전 진료를 받았어요. 진료들어가기전에 서브... 정말 너무 화가 치밀었지만 항암치료라도 해보자고 하니 을이된 입장으로 더 이상 말도 못하고 일단 돌아왔습니다. 이런 의사한테 아버지 치료를 맡겨야 하는건지 모르겠습니다. *[Father's terminal ureter cancer] My father was diagnosed with terminal ureter cancer two weeks ago. At a local university hospital, they said chemotherapy would be difficult, so we saw Dr. Choi Young-deuk at Sinchon Severance a few days ago. Before going into the consultation... I was really furious, but when they suggested trying chemotherapy, I felt like a subordinate and couldn't say anything more, so we just left. I don't know if I should trust this doctor with my father's treatment.*
3	[요관암 4기 말기 원인과 증상, 생존율 알아보기 (부작용/항암치료방법/병원)] 원인, 생존율, 요관암 4기 치료 방법, 요관암 말기병원에 대해 알아보는 시간을 가졌습니다. 전체 암 발생 중 1% 미만에 해당되는 만큼 요관암 말기에 발견되는 경우가 많습니다. 하지만 포기하지 않고 신체 상태와 요관암 항암치료를 꾸준히 이행하면서 요관암 말기 병원에서 후유증 완화를 위한 면역 관리에 꾸준히 노력하신다면 충분히 암을 이겨내실 수 있을 것입니다. *[Understanding Stage 4 Ureter Cancer: Causes, Symptoms, Survival Rates (Side Effects, Chemotherapy Methods, Hospitals)] We took some time to understand the causes, survival rates, stage 4 ureter cancer treatments, and hospitals for terminal ureter cancer. Since ureter cancer accounts for less than 1% of all cancer cases, it is often detected in its terminal stage. However, if you keep up with chemotherapy and consistent immune management for side-effect relief, there's a good chance you can overcome the cancer.*
4	[아빠의 투병 일지] 2022년 1월 아빠는 요관암 수술을 받았다. 80대인 아빠는 항암을 거부하셨고 3개월에 한번 검사를 받았다. 2023년 6월 정기검사에서 재발과 전이가 되었다. 아빠에게 재발과 전이 됐다는 소식을 전했다. 이제 철드니 부모님이 너무 아프시고 늙어 계신다. 오늘 아빠 항암 맞으러 가는 엘리베이터에서 함께 찍은 사진을 남겼다. 80대 부모님, 엄마, 아빠, 암투병, 항암치료, 힘내자. *[Dad's Cancer Battle Journal] In January 2022, my father had surgery for ureter cancer. At 80 years old, my father refused chemotherapy and underwent tests every three months. In June 2023, during a routine exam, we found out the cancer had recurred and spread. I broke the news to my father that it had returned and metastasized. Now that I'm finally growing up, my parents are so sick and frail. Today, we took a photo in the elevator while taking my dad to chemotherapy. My 80-year-old parents, mom, dad, cancer battle, chemotherapy, stay strong.*

Table 7: Document ranking results for the long-tail query **"80대 요관암 말기 암 항암치료"** *(Chemotherapy treatment for terminal stage ureteral cancer in people in their 80s)*. This query includes a highly specific intent, targeting a particular type of cancer, specific stage and a specific age group. The results must focus on this precise scenario, and not be confused with treatments for other cancers or stage or age groups. Our method ensures that documents related to the treatment process, recurrence, questions for elderly patients with ureteral cancer are ranked higher.

Query = 흑백요리사 11화 넷플릭스에 올라오는 요일과 시간 *(Episode 11 release day and time for Culinary Class Wars on Netflix)*	
Rank	**Document**
1	[넷플릭스 흑백요리사 11화 예고 ‖ 업로드 공개 시간] ▶흑백 식당 예약◀ 흑백요리사 예고 공개시간(업로드) 선공개 매주 화요일 오후 4시 1화, 2화,3화, 4화 2024년 9월 17일(화) 5화, 6화, 7화 2024년 9월 24일(화) 8화, 9화, 10화 2024년 10월 1일(화) 11화, 12화 2024년 10월 8일(화) '흑백요리사'는 넷플릭스 글로벌 TOP 10 TV(비영어) 부문에서 1위를 차지했습니다... *[Netflix Culinary Class Wars Episode 11 Preview ‖ Upload Release Time] ▶Culinary Class Wars Restaurant Reservation◀ Culinary Class Wars preview release time (upload) early release every Tuesday at 4 p.m. Episodes 1, 2, 3, 4 on September 17, 2024 (Tue), Episodes 5, 6, 7 on September 24, 2024 (Tue), Episodes 8, 9, 10 on October 1, 2024 (Tue), Episodes 11, 12 on October 8, 2024 (Tue). 'Culinary Class Wars' ranked 1st on Netflix's global TOP 10 TV (non-English) section...*
2	[넷플릭스 흑백요리사 11화 예고 ‖ 업로드 공개 시간 - KakaoNaver] 누구도 그들의 날카로운 심사를 피해갈 순 없다. <흑백요리사: 요리 계급 전쟁>, 지금 오직 넷플릭스에서. 일반적으로 넷플릭스 공식 SNS 채널이나 유튜브 채널에서 예고편을 공개하는 경우가 많으므로, 해당 플랫폼들을 확인해보시는 것이 좋을 것 같습니다. *[Netflix Culinary Class Wars Episode 11 Preview ‖ Upload Release Time - KakaoNaver] No one can escape their sharp judgment. <Black and White Chef: Culinary Class Wars>, now only on Netflix. Previews are often released on Netflix's official social media or YouTube channels, so it's a good idea to check those platforms.*
3	[넷플릭스 흑백요리사 공개시간 심사위원은 누구?] 공개시간? 나는 화요일날 땡치면 올라오는지 알았지? 흑백요리사 첫화 방영 이후 다음화는 언제 업데이트 되나 그게 제일 궁금 하더라구요 넷플릭스에서 보니 화요일마다 조금씩... 알아보니 흑백요리사의 정확한 업데이트 시간은 '화요일 오후4시' 라고 하네요 지금은 1 10화까지 공개되어있으며 마지막인 11화, 12화는 10월 8일 오후 4시에 공개가 될 것 같습니다 우승자는... *[Netflix Culinary Class Wars Release Time, Who Are the Judges?] Release time? I thought it would be uploaded exactly on Tuesday. After the airing of the first episode of Culinary Class Wars, I was most curious about when the next episodes would be updated. I found out that Culinary Class Wars is updated 'every Tuesday at 4 p.m.' Right now, episodes 1 to 10 are available, and the last episodes, 11 and 12, will be released on October 8 at 4 p.m. The winner is...*
4	[넷플릭스 흑백요리사 공개시간 몇부작 제작사 백종원 등 출연진 정보] 총 20명의 유명 요리사 '백수저' 셰프들과 80명의 흑수저 셰프가 출연해 요리 경연을 펼칩니다. 흑백요리사 공개시간 2024년 9월 17일 화요일 오후 4시에 넷플릭스(Netflix)를 통해서만 공개됩니다. 한 번에 전편이 모두 공개되는 넷플릭스 드라마와는 다르게, 9월 17일 화요일 오후 4시에 1회부터 4회까지 한번에 공개되고, 매 주 순차적으로 나머지 회차가 공개됩니다. 몇부작… *[Netflix Culinary Class Wars Release Time, Number of Episodes, Producers, and Cast Info] A total of 20 famous 'White Spoon' chefs and 80 'Black Spoon' chefs compete in cooking contests. Culinary Class Wars release time: 4 p.m. on Tuesday, September 17, 2024, exclusively on Netflix. Unlike some Netflix series that drop all episodes at once, Culinary Class Wars releases episodes in parts, with episodes 1 to 4 released on September 17, and the rest following weekly.*

Table 8: Document ranking results for the long-tail query **"흑백요리사 11화 넷플릭스에 올라오는 요일과 시간"** *(Episode 11 release day and time for Culinary Class Wars on Netflix)*. This query specifically targets the release schedule for episode 11 of a TV show. Our method accurately captures the relevant documents, focusing on precise information about release dates and times, and ranking those that explicitly mention episode details and related content higher, while not missing the overall semantic context.

Query = 40대 실비 암보험 리모델링하는 방법 *(How to Remodel Cancer and Medical Insurance for People in Their 40s)*	
Rank	**Document**
1	[40대 보험 실비 암보험 리모델링] .. 지금 가지고 있는 병력은 없고, 40대 초반 기준으로 리모델링 하여 실비 + 암보험 한달 보험료는 얼마정도가 적절할까요? 혼자서 보험 가입에... 현실적으로 저축액을 정하시고 남는 금액으로 보험을 유지하는게 가장 좋은 방법이에요. 이번에 보험료 줄이기 꼭 성공하세요. 일단 필수적인... *[Cancer and Medical Expense Insurance Remodeling for People in Their 40s] ... There are no pre-existing conditions, and based on early 40s, how much would be appropriate for a monthly premium for remodeled cancer + medical expense insurance? ... Realistically, the best way is to set aside a savings amount and maintain the insurance with the remaining amount. Make sure to succeed in reducing your premium this time...*
2	[40대 의료실비보험 리모델링 해지] 40대 초반 공무원 입니다. 의료실비보험 암보험 리모델링 가입때문에 알아보고 있습니다. 지인에게 가입해 9년 정도 유지했던 한화생명... 이런 종합보험은 지금 해지하는것이 피해를 최소화 할 수 있는 방법이며, 다른 상품이나 같은회사의 상품으로 제대로 재설계, 리모델링을 권장하며... *[Canceling and Remodeling Medical and Cancer Insurance in Their 40s] I'm in my early 40s and a civil servant. I'm looking into remodeling my medical and cancer insurance. I had been maintaining a policy with Hanwha Life for about 9 years... Canceling this type of comprehensive insurance now would minimize damages, and I recommend switching to a properly redesigned plan, even with the same company...*
3	[40대 여성 실비+암보험 어떻게 준비해야할까? :: 더 좋은생각] 실비+암 상담 및 가격/격적 무료조회 하기 ◆ 40대 여자 실비 + 암 보장은 월납입료가 얼마가 적당할까? 이는 정답이 없습니다. 그렇기 때문에 장기간 유지할 수 있는 정도로 부담없는 가격대로 준비하는 것이 유리할 수 있습니다. 장기적으로 가입을 유지하고 평생을 보장받는다면 비갱신형+순수보장형으로 설계를 하는 것이 비용을 줄일 수 있는 방법일 수 있습니다.... *[How Should Women in Their 40s Prepare for Medical + Cancer Insurance? :: Better Thoughts] Medical + cancer insurance consultation and free quotes available ◆ How much is appropriate for the monthly premium for a woman in her 40s? There is no correct answer. It's best to set a premium at a reasonable price you can maintain for a long period. For long-term coverage, choosing a non-renewable, pure insurance plan is a cost-effective way to prepare...*
4	[40대 암보험] 40대 초반이고 실비만 1개있고 암보험이 없어서 가입해야 할 것 같아서 고민중입니다. 보장금액이나 항목을 어떻게 하면 좋을까요? *[Cancer Insurance for People in Their 40s] I'm in my early 40s, and I only have one medical expense insurance plan, so I'm thinking of getting cancer insurance. What kind of coverage and terms would be best?*

Table 9: Document ranking results for the long-tail query **"40대 실비 암보험 리모델링하는 방법"** *(How to Remodel Cancer and Medical Insurance for People in Their 40s)*. Our method accurately captures both the specific age group and the detailed types of insurance, ranking documents with relevant advice on remodeling cancer and medical insurance higher.