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

NAVER

Nayoung Choi*, Youngjune Lee*, Gyu-Hwung Cho, Haeyu Jeong, Jungmin Kong, Saehun Kim, Keunchan Park, Sarah Cho, Inchang Jeong, Gyohee Nam, Sunghoon Han, Wonil Yan and Jaeho Choi

NAVER Corp., Emory University

Abstract

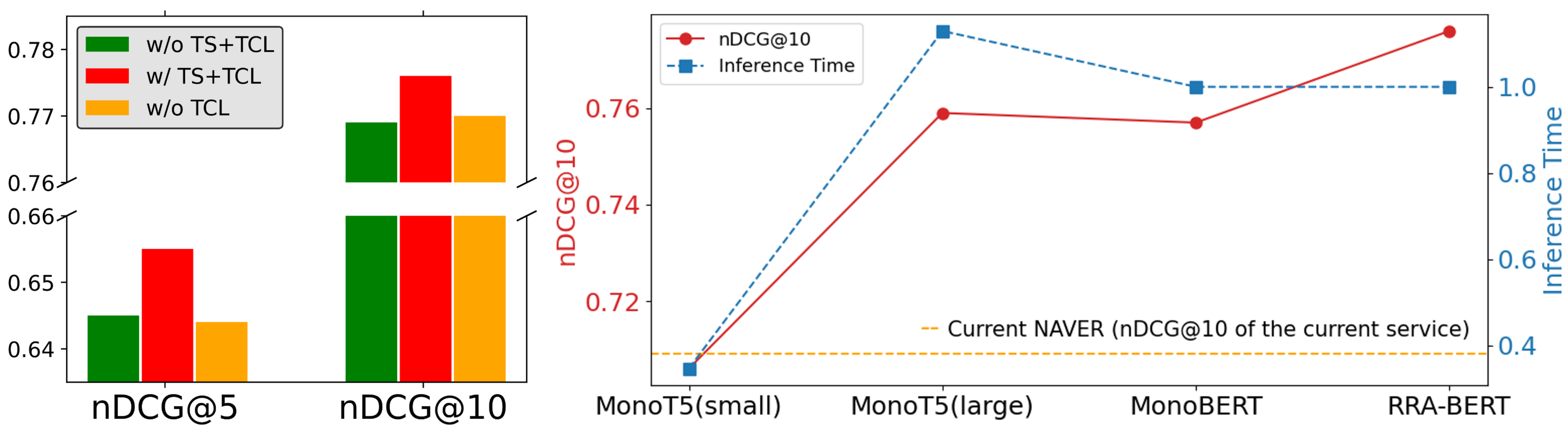
- LLMs excel at understanding complex contexts, which is especially valuable for handling long-tail queries that are long, intricate and typically lack sufficient user feedback.
- The challenge is that LLMs are too slow for ranking in real search engines, and some documents are missing.
- To address this, we proposed efficient label generation and training methods for SLM distillation, validating their effectiveness through A/B testing on NAVER.

Main Results

- Our method performs especially well on long-tail queries (NAVER), while also performing well with general queries. achieving better performance than larger models.

	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	<u>0.533</u>	<u>0.600</u>	**0.696**	**0.759**	<u>0.656</u>	**0.662**	**0.565**	<u>0.560</u>
MonoT5 (large)	<u>0.650</u>	<u>0.759</u>	0.520	0.589	0.668	0.739	0.633	0.652	<u>0.560</u>	**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	<u>0.686</u>	0.733	0.621	0.620	0.480	0.480
RRA-BERT (ours)	**0.655**	**0.776**	**0.543**	**0.607**	0.671	<u>0.743</u>	**0.667**	<u>0.658</u>	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

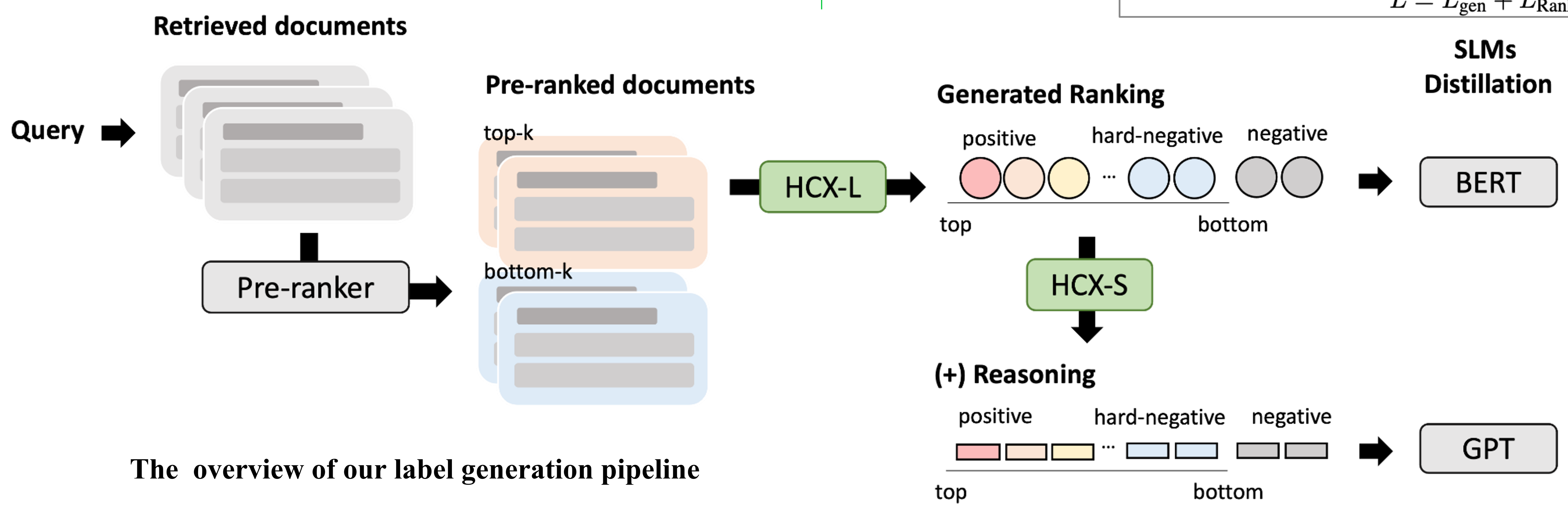
- The additional layer used during training can be omitted during inference without any performance degradation, making it an efficient training method for industry setting.
- In online A/B testing, our final model, RRA-BERT, improved CTR by 5.63%, top-1 document clicks by 5.9%, and dwell time by 7.97% compared to the current search results.



Effectiveness study of TCL using RRA-BERT (left) and the comparison of inference time ratio and nDCG@10 across four models (right).

Ranking label generation pipeline

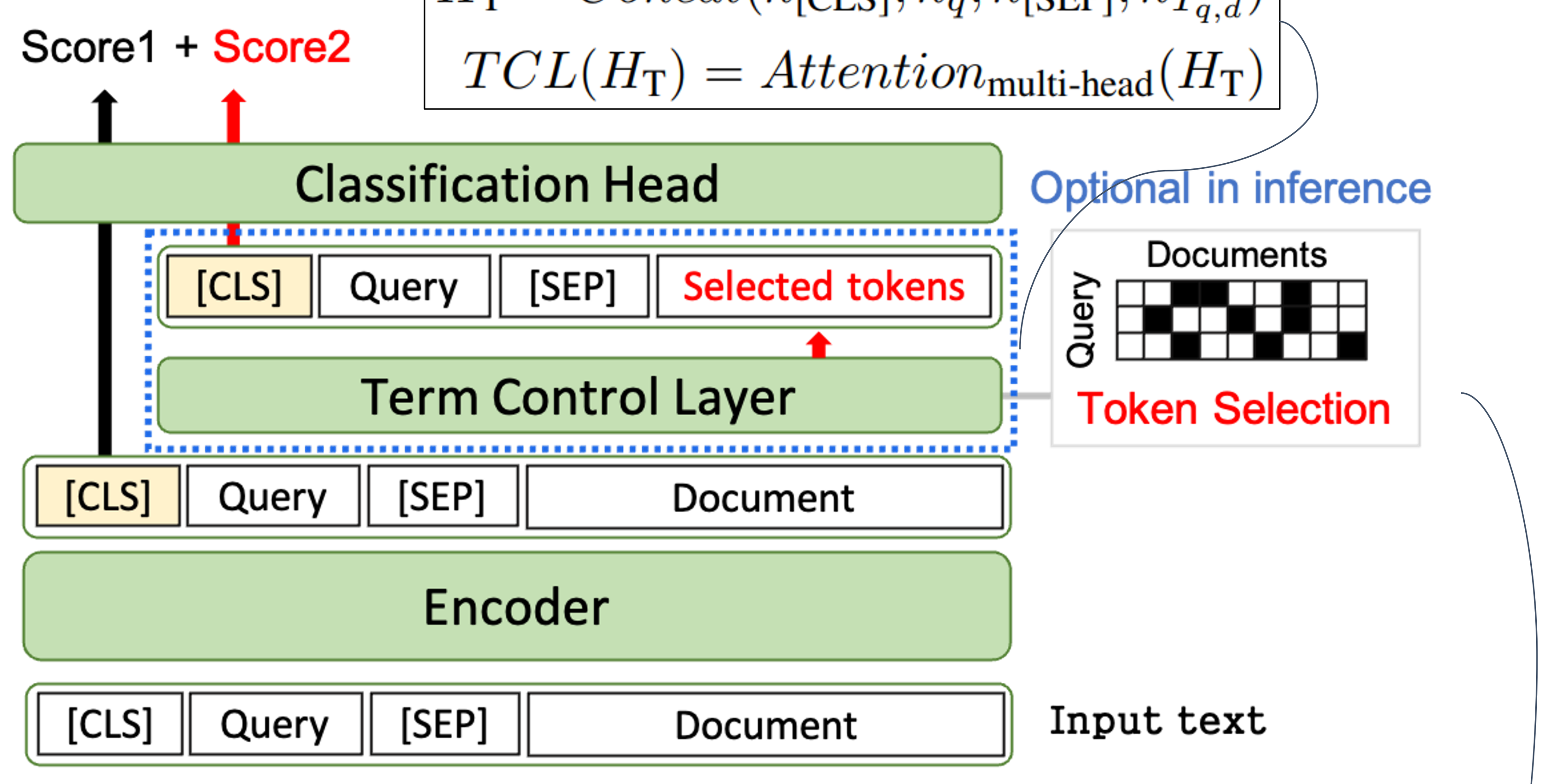
- The pre-ranker filters documents based on relative relevance, selecting documents that are more and less relevant.
- After conducting list-wise ranking with the LLM, the excluded (missing) documents are labeled as hard negatives.



The overview of our label generation pipeline

RRA-BERT

- RRA-BERT enhances BERT-based ranking by incorporating **Token Selection**, which identifies tokens in the document with meanings similar to those in the query.
- Using a **Term Control Layer**, signals from selected tokens are injected into the training process, allowing the model to focus on key terms while preserving the overall semantic context.



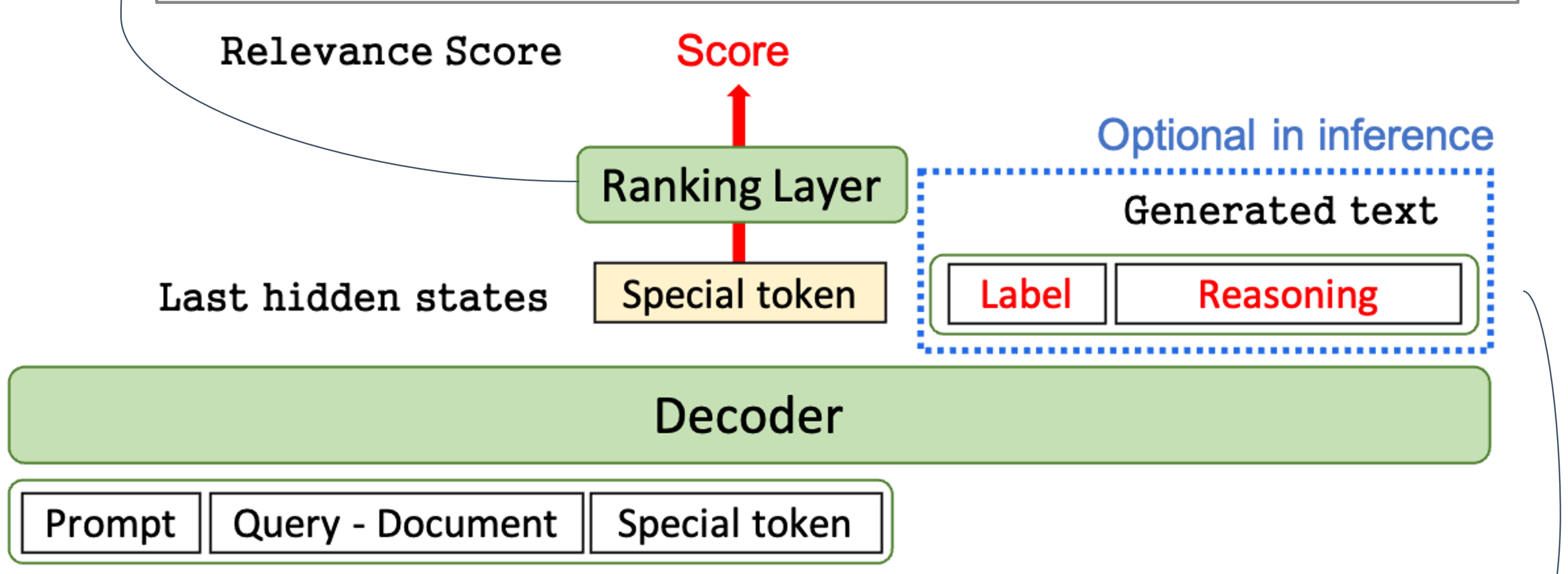
$$H_{\text{T}} = Concat(h_{[\text{CLS}]}, h_q, h_{[\text{SEP}]}, h_{T_{q,d}})$$
$$TCL(H_{\text{T}}) = Attention_{\text{multi-head}}(H_{\text{T}})$$

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

RRA-GPT

- RRA-GPT enhances GPT-based ranking by incorporating a dense **ranking layer** and leveraging **generative capabilities** for relevance classification and reasoning.

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



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