BREADBOWL-EMBED: ENCODE ONCE, RETRIEVE AND RERANK

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern retrieval systems typically separate candidate generation from relevance refinement: bi-encoders provide efficient retrieval, while cross-encoders recover richer query–document interactions at substantially higher inference cost. We ask whether these two stages can instead operate on the same representation. We introduce *BreadBowl-Embed*, a retrieval model built around a *Slot Encoder*, which represents each query and document as a compact set of *routing–value slots*. The routing vectors provide an indexable representation for first-stage retrieval, while the same routing vectors subsequently define attention over a candidate's stored value vectors to produce query-dependent readouts for reranking. Retrieval and reranking therefore share not only an encoder, but the representation that determines both which documents are retrieved and how their contents are compared. Documents are *encoded once* and reused across both stages, so reranking requires neither a separate cross-encoder nor an additional backbone pass over candidate text. Across 15 BEIR tasks, value-based reranking raises macro nDCG@10 from 48.47 to 51.54 (scores scaled by 100) over routing alone on the same retrieved candidates. These results suggest that indexed retrieval and query-dependent relevance refinement can be supported by a single precomputed representation.

1 INTRODUCTION

Modern retrieval pipelines typically operate in two stages: candidate retrieval and reranking. Bi-encoders independently encode queries and documents for efficient candidate search (Huang et al., 2013; Reimers & Gurevych, 2019; Karpukhin et al., 2020), while cross-encoders jointly process shortlisted query–document pairs for finer relevance judgments (Devlin et al., 2019; Nogueira & Cho, 2019). This division makes large-scale search practical, but refinement requires another pass over candidate text. How much of this refinement can the retrieval representation itself support?

Late-interaction models move query-dependent comparison into the space of independently computed representations. ColBERT uses cached token vectors for indexed retrieval and candidate scoring (Khattab & Zaharia, 2020), while Multi-Vector Attention models compare query values with attention-weighted readouts of document values (Zhou & Devlin, 2021). Building on this line of work, we study how to construct a fixed-budget representation for both indexed discovery and attention-based refinement. The challenge is to retain information useful for detailed comparison while making the documents favored by that comparison discoverable through search.

We introduce **BreadBowl-Embed**, a model that uses the same representation for retrieval and reranking (Figure 1). Its shared *Slot Encoder* maps each query and document independently to a fixed number of *slots*. Each slot contains a *routing vector* for candidate search and a *value vector* for relevance refinement. Slots are constructed using *Referenced Cross Attention (RCA)*: standard cross-attention (Vaswani et al., 2017) in which a shared set of learned reference vectors attends to the input's contextual token representations.

The central property is *polymorphic routing*: the same routing vectors that make a document searchable also determine how its stored values are read. Query–document routing similarities define attention over those values, so the document representation remains fixed while its readout depends

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

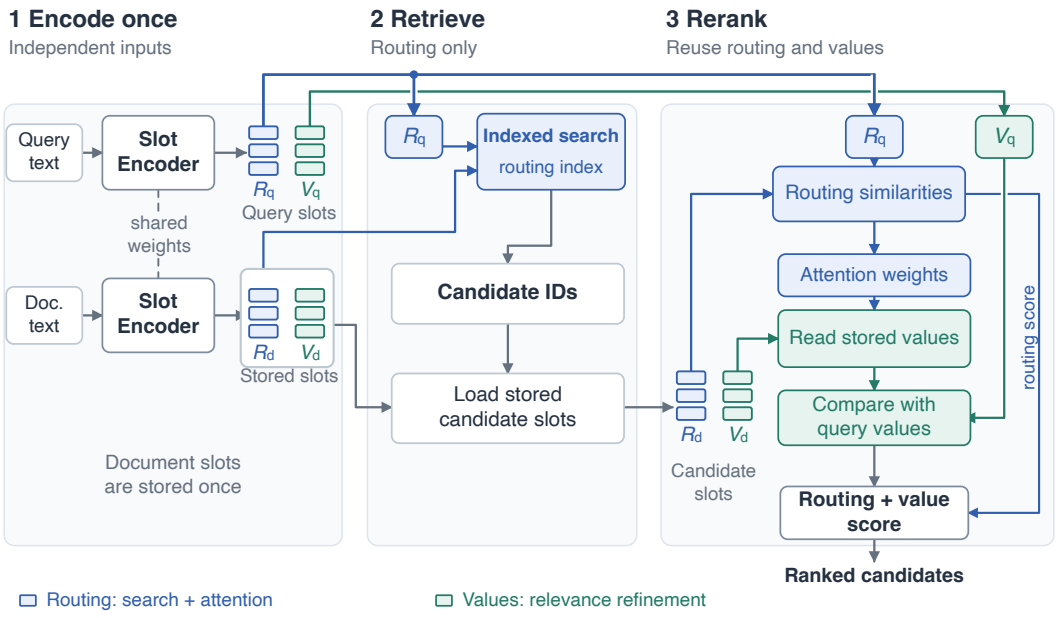

Figure 1: **One representation for retrieval and reranking.** A shared Slot Encoder independently maps queries and documents to paired routing–value slots. Routing vectors support indexed candidate discovery and then define attention over the candidates' stored values. Comparing these readouts with query values supplements the routing score without re-encoding candidate text. The three illustrated slots are schematic.

on the query. Documents are encoded offline and each query is encoded once; retrieval and reranking then operate on these representations without a separate cross-encoder or another backbone pass over candidate text.

Sharing the representation also couples the two learning objectives: adapting the slots for reranking can alter the routing vectors used for candidate retrieval. We therefore combine reranking distillation with *retrieval replay*, periodically revisiting the retrieval objective during reranking training. Teacher supervision trains the value pathway to refine relevance scores, while a contrastive retrieval objective supervises the routing pathway. Both objectives train the same RCA slot extractor. This training strategy aims to preserve candidate retrieval as the shared representation learns to support query-dependent relevance refinement, without introducing an additional inference-time module.

We contribute a unified slot-based architecture centered on polymorphic routing vectors, which support indexed retrieval and define attention over stored values for reranking. Experiments on 15 BEIR tasks demonstrate useful refinement of the model's own retrieved candidates. We detail the training and evaluation methodology and will open-source our code, model checkpoints, and experimental pipelines.

2 Related Work

Bi-encoders and cross-encoders. A single-vector bi-encoder maps each query and document independently to one vector in a shared space (Huang et al., 2013; Reimers & Gurevych, 2019; Karpukhin et al., 2020). Dot-product or cosine similarity supports vector-index retrieval and inexpensive candidate rescoring, but comparison is restricted to one vector pair. Cross-encoders instead process query and document tokens jointly (Nogueira & Cho, 2019), allowing interaction throughout the backbone. They require a backbone pass for each pair and are typically used to rerank retrieved candidates. UR^{2}N (Bhat et al., 2025) and E^{2}Rank (Liu et al., 2026) share a backbone across retrieval and reranking, but still process candidate text again during reranking. Table 1 compares representative encoding and scoring architectures.

Table 1: Representation and scoring designs for retrieval and reranking. Costs describe candidate scoring, not complete search latency.

Architecture	Retriever	Reranker	Extra backbone pass	Separate value readout	Stored doc. vectors	Scoring cost
Bi-encoder	✓	✓*	None	—	1	$O(h)$
Cross-encoder	—	✓	Per pair	—	—	$B(n_q + n_d)$
Poly-encoder	—	✓	None	—	1	$O(m_q h)$
ColBERT	✓	✓	None	—	n_d	$O(n_q n_d h)$
ConstBERT	✓	✓	None	—	m_d	$O(n_q m_d h)$
MVA (pooled)	—	✓	None	✓	$2m_d$	$O(m_q m_d h)$
BreadBowl-Embed	✓	✓	None	✓	$2m_d$	$O(m_q m_d h)$

Retriever denotes vector-index candidate generation; a dash indicates that this is not part of the cited pipeline. Reranker denotes scoring a supplied candidate set; *the bi-encoder reuses its pooled-vector score. Value readout denotes attention over cached document values distinct from the routing keys. Here n_q, n_d are token counts, m_q, m_d are pooled-vector counts, h is a common vector width for this accounting, and $B(\ell)$ is a backbone pass on ℓ tokens. Counts precede compression and omit index duplication; costs omit independent encoding, search, and data transfer.

Late interaction. Late-interaction models defer query–document interaction until after independent encoding, comparing multiple token or pooled vectors rather than a single vector pair. Poly-encoders (Humeau et al., 2020) let a cached candidate vector attend over learned summaries of the query context. This supports inexpensive reranking but does not directly support retrieval. ColBERT (Khattab & Zaharia, 2020) retains contextual token vectors and sums each query token's maximum similarity to a document token (MaxSim), supporting both indexed retrieval and cached reranking. Storage and comparison costs grow with token counts; ColBERTv2 reduces storage through residual compression (Santhanam et al., 2022). Multi-Vector Attention (MVA) (Zhou & Devlin, 2021) separates keys from values: key similarities determine attention over stored document values, whose weighted combinations are compared with query values. MVA is the closest antecedent to our value scorer, but its evaluated pipeline only reranks externally retrieved candidates. In BreadBowl-Embed, the same routing vectors support indexed retrieval and determine attention over stored document values during reranking.

Retrieval and attention. Attention keys can also serve as retrieval vectors. ReAtt indexes document attention keys for retrieval, then applies further Transformer layers to jointly process each query–document pair for question answering (Jiang et al., 2022). Memorizing Transformers use retrieved keys to attend over cached values during language modeling (Wu et al., 2022).

Learned pooling and slot construction. Learned pooling aggregates token states into a smaller set of vectors using trainable weights or queries. Lin et al. (2017) pool recurrent states into multiple attention-weighted vectors; Poly-encoders pool Transformer states with learned codes. Set Transformer uses learned seed queries for attention pooling (Lee et al., 2019), and Perceiver uses cross-attention to read inputs into a fixed set of latent vectors (Jaegle et al., 2021). MVA also studies learned pooling tokens. ConstBERT (MacAvaney et al., 2025) projects document token states to a fixed number of vectors while retaining MaxSim for retrieval and reranking. BreadBowl-Embed's RCA pools final backbone states with a shared set of learned reference queries using standard cross-attention (Vaswani et al., 2017). The pooled outputs are projected into a fixed number of routing–value slots shared by retrieval and reranking.

3 METHOD

BreadBowl-Embed encodes queries and documents independently into paired routing–value slots. Routing vectors first identify candidate documents, then determine attention over their stored values to refine relevance. The same representation supports both operations, as illustrated in Figure 1.

3.1 A SHARED SLOT ENCODER

The Slot Encoder (Figure 2) maps an input x to m pairs $(\mathbf{r}_i^x, \mathbf{v}_i^x)$, collected into routing vectors $\mathbf{R}_x \in \mathbb{R}^{m \times d_r}$ and value vectors $\mathbf{V}_x \in \mathbb{R}^{m \times d_v}$. A slot is an input-dependent pooled representation, rather than an individual token. Queries and documents share all encoder parameters.

A backbone first produces contextual token states $\mathbf{H}_x = f_\theta(x) \in \mathbb{R}^{n_x \times d_b}$. RCA pools these states using a learned reference bank $\mathbf{E} \in \mathbb{R}^{m \times d_b}$:

$$\mathbf{Z}_x = \mathrm{RCA}(\mathbf{E}, \mathbf{H}_x) = \mathrm{LN}(\mathrm{MultiHead}(\mathbf{E}, \mathbf{H}_x, \mathbf{H}_x)) \in \mathbb{R}^{m \times d_b}. \tag{1}$$

Here MultiHead is standard multi-head cross-attention (Vaswani et al., 2017), including its output projection and padding mask, and LN denotes layer normalization (Ba et al., 2016). Projected references supply attention queries; projected token states supply keys and values. Each reference pools information across the input into one slot state. The references are shared parameters, while their attention weights and resulting slot states depend on the input.

Two heads map the same slot states to routing and value vectors:

$$\mathbf{R}_x = \mathbf{Z}_x \mathbf{W}_r + \mathbf{b}_r, \qquad \mathbf{V}_x = \mathbf{Z}_x \mathbf{W}_v + \mathbf{b}_v + h_\psi(\mathbf{Z}_x). \tag{2}$$

The routing head is linear. The value head adds a residual two-layer multilayer perceptron (MLP) h_ψ, applied to each slot state, to increase value capacity without adding slots or another extractor. Both heads run during independent encoding, and their outputs are stored together. Appendix A.1 illustrates RCA (Figure 3) and specifies the projections, residual branch, and model dimensions.

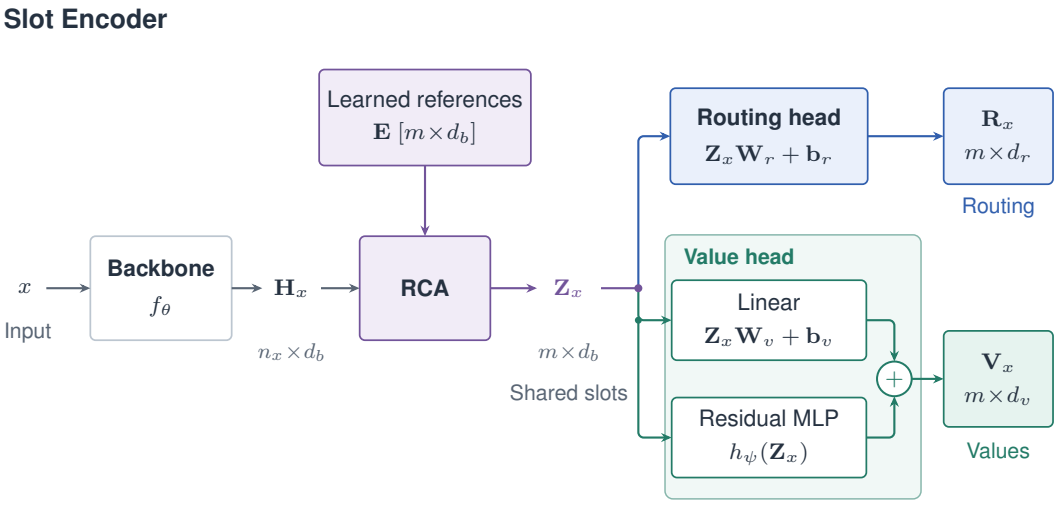

Figure 2: **The shared Slot Encoder in the final model.** Learned references pool backbone token states into shared slot states. Both output heads read these states; the value head adds a residual MLP to its linear projection. Each query or document is encoded independently, and document routing and value vectors are cached together.

3.2 RETRIEVAL THROUGH ROUTING VECTORS

Let a hat denote rowwise ℓ_2 normalization. For a query q and document d, the matrix $\mathbf{C}(q,d) = \widehat{\mathbf{R}}_q \widehat{\mathbf{R}}_d^\top$ contains all query-slot/document-slot routing similarities. We aggregate them with a smooth maximum over document slots, averaged over query slots:

$$s_r(q,d) = \frac{1}{m} \sum_{i=1}^{m} \tau_r \log \left[\frac{1}{m} \sum_{j=1}^{m} \exp\left(\frac{C_{ij}(q,d)}{\tau_r} \right) \right]. \tag{3}$$

This normalized log-mean-exp score depends only on routing vectors and approaches mean MaxSim as $\tau_r \to 0$.

Indexed routing-slot search produces a bounded document shortlist, which is rescored with s_r before value reranking (Algorithm 1). Finite search budgets approximate corpus-wide ranking by s_r; Appendix A.2 specifies hit aggregation and budgets.

3.3 Reranking through stored values

For each candidate, the same routing similarities define attention over its stored values:

$$\mathbf{A}(q,d) = \mathrm{softmax}_{\text{document slots}}\left(\frac{\mathbf{C}(q,d)}{\tau_v}\right), \qquad \mathbf{U}(q,d) = \mathbf{A}(q,d)\mathbf{V}_d. \tag{4}$$

Row $\mathbf{u}_i(q,d)$ is the document readout for query slot i: a weighted combination of the candidate's value vectors. Any query slot can attend to any document slot. The stored document vectors remain fixed, while the readouts depend on the query. Unlike RCA, which pools tokens within one input, this attention operates between two independently encoded slot sets.

We compare each readout with the value of its query slot and combine this agreement with the routing score:

$$\begin{aligned} s_v(q,d) &= \frac{1}{m}\sum_{i=1}^{m} \langle \widehat{\mathbf{v}}_i^q, \widehat{\mathbf{u}}_i(q,d)\rangle, \\ s(q,d) &= (1-\beta)s_r(q,d) + \beta s_v(q,d). \end{aligned} \tag{5}$$

Document values are aggregated before normalization; query values and the resulting readouts are normalized for cosine comparison. This is *polymorphic routing*: the vectors used for candidate discovery also serve as attention queries and keys, without additional projections at comparison time.

Algorithm 1 summarizes inference. Reranking costs $O(m^2(d_r + d_v))$ per pair, independent of candidate text length, without another backbone pass. Appendix A.2 details the additional encoding, index-search, and storage costs.

Algorithm 1 Encode once, retrieve and rerank

Input: Trained Slot Encoder, corpus $\mathcal{D}$, query q
Budgets: L hits per query slot; B shortlisted documents; K reranked documents
Offline: encode and index documents
For each $d \in \mathcal{D}$, compute and cache $(\mathbf{R}_d, \mathbf{V}_d) \leftarrow \mathrm{SlotEncoder}(d)$
Build index $\mathcal{I}$ over all normalized routing rows $\widehat{\mathbf{R}}_d$, retaining document IDs
Online: once per query
$(\mathbf{R}_q, \mathbf{V}_q) \leftarrow \mathrm{SlotEncoder}(q)$
Search $\mathcal{I}$ with each $\widehat{\mathbf{r}}_i^q$ to obtain its top-L slot hits $\mathcal{H}_i(q)$
Group hits by document; retain up to B by coarse score G as $\mathcal{C}_B(q)$ (Eq. 13)
Compute $s_r(q,d)$ for all $d \in \mathcal{C}_B(q)$ using their complete routing sets (Eq. 3)
$\mathcal{C}_K(q) \leftarrow$ up to K documents in $\mathcal{C}_B(q)$ with highest $s_r(q,d)$
for each $d \in \mathcal{C}_K(q)$ **do**
    Compute attention $\mathbf{A}(q,d)$ from routing vectors $\mathbf{R}_q$ and $\mathbf{R}_d$ (Eq. 4)
    Read stored document values: $\mathbf{U}(q,d) \leftarrow \mathbf{A}(q,d)\mathbf{V}_d$
    Compute $s_v(q,d)$ and combined score $s(q,d)$ (Eq. 5)
end for
return $\mathcal{C}_K(q)$ sorted by decreasing $s(q,d)$

3.4 Training

Training has two stages: weak contrastive training of the Slot Encoder, followed by distillation with retrieval replay. The corresponding data preparation is described in Appendix B.1.

Weak supervision. We first train the Slot Encoder end-to-end on weak query–document pairs, using linear routing and value heads. The residual value MLP is absent at this stage. A contrastive loss $\mathcal{L}_{\text{route}}$ supervises routing scores over in-batch documents, while $\mathcal{L}_{\text{comb}}$ supervises combined

scores over paired positives and routing-selected hard negatives. A routing-slot diversity penalty discourages redundant slots. At weak-training update u, we minimize

$$\mathcal{L}_{\text{weak},u} = \mathcal{L}_{\text{route},u} + \mu_u \mathcal{L}_{\text{comb},u} + \eta \mathcal{L}_{\text{div},u}, \tag{6}$$

where μ_u controls the combined-score objective and η weights the diversity penalty.

Distillation with retrieval replay. For adaptation, the value head includes the residual MLP from Equation 2. We freeze the backbone and routing head and train the remaining Slot Encoder parameters. We normalize the cross-encoder teacher's scores and the student's combined scores into distributions p_T and p_S over the same candidate documents, following knowledge distillation (Hinton et al., 2015). We minimize their listwise Kullback–Leibler (KL) divergence:

$$\mathcal{L}_{\text{dist}} = \mathbb{E}_q[D_{\text{KL}}(p_T(\cdot \mid q) \,\|\, p_S(\cdot \mid q))]\,. \tag{7}$$

Distillation gradients pass through query and document values; the routing score and routing-based attention weights are treated as constants.

Adapting shared slot states can change routing vectors even with a frozen routing head. We therefore use periodic retrieval replay: a contrastive routing loss on independently sampled queries with labeled positive documents, optimized alongside distillation. At joint update t, the objective is

$$\mathcal{L}_t = \mathcal{L}_{\text{dist},t} + \lambda_t \mathcal{L}_{\text{ret},t}, \tag{8}$$

where λ_t is nonzero only on scheduled replay updates. Replay directly supervises candidate discovery while distillation trains relevance refinement. Appendix A.3 specifies the losses, gradient paths, sampling, and replay schedule. After training, document encodings and the index are built from the final checkpoint.

4 Experiments

We test whether stored values improve ranking beyond the routing score used to retrieve candidates, and how retrieval replay affects candidate discovery and final ranking.

4.1 Training and evaluation setup

We use Qwen3.5-0.8B-Base (Qwen Team, 2026) with 16 slots per input, 256-dimensional routing and value vectors, and query/document limits of 128/384 tokens. Weak contrastive training draws on LightOn's curated pre-training collection (Sourty et al., 2026) with a 30-million-pair preparation target. A separate pool of 500,000 queries from nine sources supplies teacher-scored candidates for distillation and annotated positives for retrieval replay (Appendix B.1). Weak training took approximately 8 hours on eight H200 GPUs; the reported KL+replay run required approximately 10 hours of optimization on eight RTX 5090 GPUs to reach the selected checkpoint.

We evaluate on 15 BEIR tasks using their full corpora (Thakur et al., 2021). Model selection and training ablations use 5,000 held-out development queries across nine prepared corpora, distinct from the official benchmark collections. We report ranking quality (nDCG@10) and candidate coverage (recall@100), averaging tasks or development sources equally and scaling scores by 100.

For BEIR, we fix one development-selected checkpoint and its retrieval settings. We compare routing alone (QK) with routing plus value scoring (QKV) on the same 100 routing-selected candidates, without inserting relevant documents. This isolates the contribution of values at inference. Appendix B gives search settings, evaluation protocols, and checkpoint selection.

4.2 Refinement from retrieved representations

Table 2 shows that adding the value score improves BEIR macro nDCG@10 from 48.47 to 51.54, with gains on 10 of 15 tasks. Gains are largest on NQ, TREC-COVID, and FEVER; five tasks decline, most notably ArguAna. The same routing vectors retrieve candidates and define attention over their cached values, without re-encoding text.

These results show that the retrieval representation retains information useful for query-dependent refinement beyond its routing score. Appendix C provides an illustrative inspection of Slot Encoder token pooling.

Table 2: BEIR results for the final checkpoint. QK and QKV rank the same retrieved candidates; their recall@100 is therefore identical. All metrics are multiplied by 100; Δ is QKV minus QK, computed before rounding. Bold indicates the better nDCG@10 in each row.

	nDCG@10			
Task	QK	QKV	Δ	Recall@100
ArguAna	**57.69**	52.49	−5.20	97.44
CQADupStack	**41.89**	41.82	−0.08	76.13
Climate-FEVER	22.14	**29.05**	+6.91	55.18
DBPedia-Entity	38.43	**41.18**	+2.75	52.38
FEVER	77.90	**86.58**	+8.68	95.81
FiQA	43.21	**43.61**	+0.40	75.38
HotpotQA	60.90	**64.74**	+3.83	75.04
MS MARCO	31.91	**34.27**	+2.37	82.01
NFCorpus	38.52	**38.87**	+0.35	30.22
NQ	43.26	**55.16**	+11.90	91.32
Quora	**87.17**	84.96	−2.20	98.91
SCIDOCS	**21.79**	21.45	−0.34	42.95
SciFact	**73.78**	72.74	−1.05	96.50
TREC-COVID	66.45	**78.35**	+11.90	10.79
Touche2020	22.00	**27.77**	+5.77	38.66
Macro mean	48.47	**51.54**	+3.07	67.91

For context, published BEIR means remain higher for Qwen3-Embedding-0.6B (55.52) and Jina v5 text-small (56.67) (Akram et al., 2026). Appendix B.5 reports external-model comparisons and their protocol differences.

4.3 Effect of retrieval replay

We compare KL-only distillation with KL plus replay using the same initialization, training queries, and 4,000 distillation updates. Replay adds training compute. Each checkpoint re-encodes the development corpora and retrieves its own candidates, measuring changes in discovery as well as ranking.

Replay improves both routing and final nDCG@10, with final quality rising from 62.57 to 63.42 (Table 3). Recall@100 also increases, although coarse-shortlist recall falls slightly. Retrieval supervision therefore improves the combined system without preserving all aspects of candidate coverage. Appendix B.6 reports confidence intervals and their scope.

Table 3: Replay ablation with fresh retrieval on the 5,000-query development set. Shortlist recall measures coverage before routing selects the final 100 candidates. Metrics are source-macro averages multiplied by 100.

	Recall		nDCG@10	
Training	Shortlist	@100	QK	QKV
KL only	88.50	86.63	58.64	62.57
KL + replay	88.38	87.19	60.40	63.42

5 CONCLUSION

BreadBowl-Embed demonstrates that a searchable document representation can retain the information needed for query-dependent relevance refinement. Its Slot Encoder produces paired routing–value slots, with polymorphic routing connecting indexed candidate discovery to attention over stored values. Across BEIR, adding value readouts improves average ranking over routing alone on identical retrieved candidates. This provides direct evidence that refinement can use information already present in the retrieval representation, without another backbone pass over candidate text. Retrieval replay further improves routing and final ranking during adaptation. The resulting architecture makes retrieval and reranking complementary uses of one reusable representation: the document is encoded once, and the query determines how its stored information is read.

AI USE STATEMENT

We used generative AI tools to support the research process, discover and retrieve related literature, draft portions of the manuscript, and revise the writing for clarity and readability. The authors take responsibility for the accuracy, originality, and integrity of the final work, including all text, citations, methods, code, and reported results produced with AI assistance.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A specify the model architecture, retrieval procedure, and training objectives. Appendix B describes data preparation, training settings, checkpoint selection, and evaluation protocols, including the scope and limitations of the reported experiments.

REFERENCES

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation, 2026. URL https://arxiv.org/abs/2602.15547v1. Version 1.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

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. MS MARCO: A human generated MAchine Reading COmprehension dataset, 2016. URL https://arxiv.org/abs/1611.09268.

Riyaz Ahmad Bhat, Jaydeep Sen, Rudra Murthy, and Vignesh P. UR2N: Unified retriever and ReraNker. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 595–602. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-industry.51/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library, 2024. URL https://arxiv.org/abs/2401.08281.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the*

22nd ACM International Conference on Information & Knowledge Management, pp. 2333–2338. Association for Computing Machinery, 2013. doi: 10.1145/2505515.2505665. URL https://doi.org/10.1145/2505515.2505665.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkxgnnNFvH.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Zhengbao Jiang, Luyu Gao, Jun Araki, Haibo Ding, Zhiruo Wang, Jamie Callan, and Graham Neubig. Retrieval as attention: End-to-end learning of retrieval and reading within a single Transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2336–2349. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.149. URL https://aclanthology.org/2022.emnlp-main.149/.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, 2020. doi: 10.1145/3397271.3401075. URL https://arxiv.org/abs/2004.12832.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1703.03130.

Qi Liu, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, and Jiaxin Mao. E2Rank: Unifying text embedding and listwise reranking for effective and efficient search, 2026. URL https://arxiv.org/abs/2510.22733v3. Version 3; accepted to EMNLP 2026.

Sean MacAvaney, Antonio Mallia, and Nicola Tonellotto. Efficient constant-space multi-vector retrieval. In *Advances in Information Retrieval*, pp. 237–245, 2025. doi: 10.1007/978-3-031-88714-7_22. URL https://arxiv.org/abs/2504.01818. ECIR 2025.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT, 2019. URL https://arxiv.org/abs/1901.04085.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. SciRepEval: A multi-format benchmark for scientific document representations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5548–5566. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.338. URL https://aclanthology.org/2023.emnlp-main.338/.

Raphael Sourty, Antoine Chaffin, Orion Weller, Paulo Roberto Moura Junior, and Amélie Chatelain. DenseOn with the LateOn: Open state-of-the-art single and multi-vector models. Hugging Face blog, 2026. URL https://huggingface.co/blog/lightonai/denseon-lateon.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://arxiv.org/abs/2104.08663.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Christophe Van Gysel and Maarten de Rijke. Pytrec_eval: An extremely fast Python interface to trec_eval. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2018. URL https://github.com/cvangysel/pytrec_eval.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing Transformers. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2203.08913.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1393–1412. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-industry.103. URL https://aclanthology.org/2024.emnlp-industry.103/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Giulio Zhou and Jacob Devlin. Multi-vector attention models for deep re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5452–5456. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.443. URL https://aclanthology.org/2021.emnlp-main.443/.

A ENCODER, RETRIEVAL, AND TRAINING DETAILS

A.1 RCA AND THE VALUE HEAD

The encoder equations describe inference; training additionally applies dropout at the locations specified below. RCA applies one multi-head cross-attention pooling operation to the final backbone states (Figure 3).

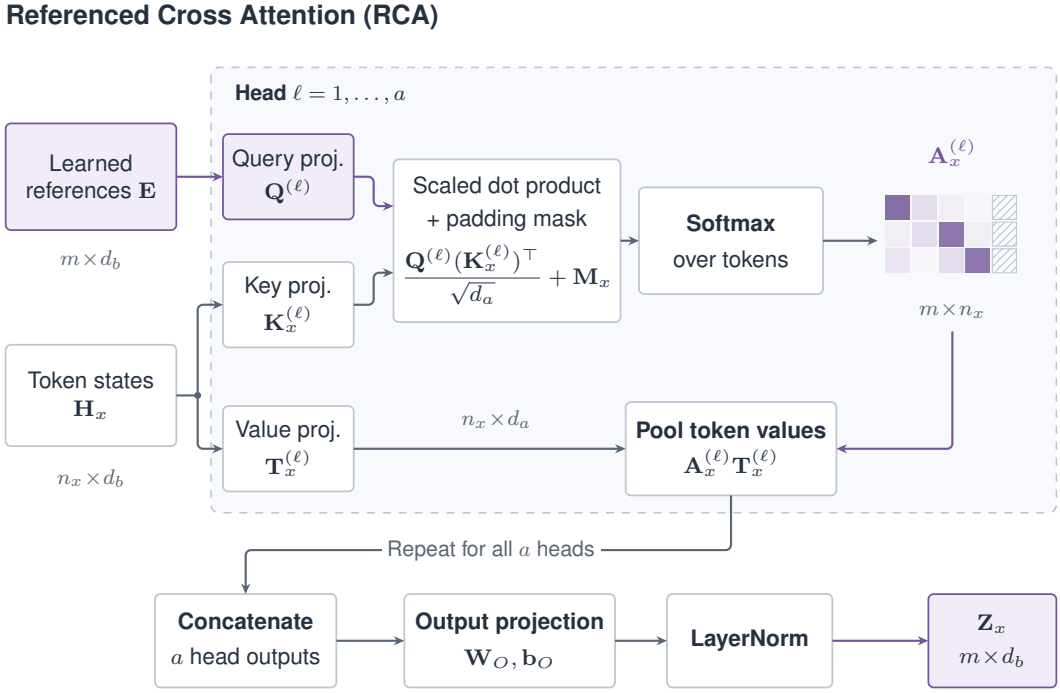

Figure 3: **Referenced Cross Attention (RCA).** Each head projects learned references into queries and token states into keys and token values. Masked attention pools token values; concatenation, output projection, and layer normalization produce the slot states. The schematic attention map has reference rows and token columns; hatching marks padding. Token values $\mathbf{T}_x^{(\ell)}$ differ from the final value slots $\mathbf{V}_x$. Training additionally applies attention dropout before pooling.

For head $\ell \in \{1, \ldots, a\}$, affine projections produce reference queries, token keys, and token values of width d_a:

$$\begin{aligned} \mathbf{Q}^{(\ell)} &= \mathbf{E}\mathbf{W}_Q^{(\ell)} + \mathbf{b}_Q^{(\ell)}, \\ \mathbf{K}_x^{(\ell)} &= \mathbf{H}_x\mathbf{W}_K^{(\ell)} + \mathbf{b}_K^{(\ell)}, \qquad \mathbf{T}_x^{(\ell)} = \mathbf{H}_x\mathbf{W}_T^{(\ell)} + \mathbf{b}_T^{(\ell)}. \end{aligned} \tag{9}$$

The projection matrices have shape $d_b \times d_a$; biases are broadcast across rows. The token-value projection $\mathbf{W}_T^{(\ell)}$ is distinct from the final value head $\mathbf{W}_v$. Each head computes

$$\mathbf{A}_x^{(\ell)} = \mathrm{softmax}_{\mathrm{tokens}}\left(\frac{\mathbf{Q}^{(\ell)}(\mathbf{K}_x^{(\ell)})^\top}{\sqrt{d_a}} + \mathbf{M}_x\right), \tag{10}$$

where $\mathbf{M}_x$ is zero for valid tokens and masks padding before softmax. The concatenated head outputs are mapped back to width d_b by $\mathbf{W}_O \in \mathbb{R}^{(ad_a)\times d_b}$:

$$\mathbf{Z}_x = \mathrm{LN}\Big(\mathrm{Concat}_{\ell=1}^{a}\left[\mathbf{A}_x^{(\ell)}\mathbf{T}_x^{(\ell)}\right]\mathbf{W}_O + \mathbf{b}_O\Big). \tag{11}$$

The extractor has no reference skip connection, feedforward block, or iterative slot refinement. During training, dropout with probability 0.1 is applied to attention weights before the token-value aggregation.

The residual branch of the value head in Equation 2 is

$$h_\psi(\mathbf{Z}_x) = \mathrm{SiLU}(\mathbf{Z}_x\mathbf{W}_\downarrow + \mathbf{b}_\downarrow)\mathbf{W}_\uparrow + \mathbf{b}_\uparrow. \tag{12}$$

Here $\mathrm{SiLU}(z) = z\sigma(z)$, where σ is the logistic sigmoid; training applies dropout with probability 0.1 after SiLU. The branch reads the shared slot states $\mathbf{Z}_x$ and is added to the linear value output. Its up-projection and bias are zero-initialized when the branch is introduced, preserving the initial linear value function.

Table 4: Encoder and scoring configuration. All slot vectors are produced during independent query or document encoding.

Component	Setting
Backbone	Qwen3.5-0.8B-Base (Qwen Team, 2026)
Backbone / slot-state width d_b	1024
Slots per input m	16
RCA heads a / per-head width d_a	4 / 128
Routing / value widths d_r, d_v	256 / 256
Residual value MLP hidden width	512
Query / document token limits	128 / 384
Routing temperature τ_r	0.02
Value-attention temperature τ_v	0.10
Value mixing coefficient β	0.25

Table 4 lists the evaluated configuration. The encoder returns unnormalized head outputs. Routing vectors are ℓ_2-normalized for indexing and scoring. Value attention aggregates raw document values; their relative norms can therefore affect the readout direction. Only query values and readouts are normalized for the final cosine comparison.

A.2 Candidate selection and computational cost

Each normalized document routing vector is indexed with its document identifier. For query slot i, let $\mathcal{H}_i(q)$ contain its top-L document-slot hits, and let $\mathcal{U}(q)$ be their union of document identifiers. The initial score averages the strongest returned hit for each query slot, assigning -1 when a document has no hit for that slot:

$$\begin{aligned} g_i(q,d) &= \max(\{-1\} \cup \{C_{ij}(q,d) : (d,j) \in \mathcal{H}_i(q)\}), \\ G(q,d) &= \frac{1}{m}\sum_{i=1}^{m} g_i(q,d). \end{aligned} \tag{13}$$

We retain up to B documents by G, compute s_r using all of their routing slots, and retain up to K candidates for value reranking:

$$\mathcal{C}_B(q) = \operatorname*{Top}_{B,\, d\in\mathcal{U}(q)} G(q,d), \qquad \mathcal{C}_K(q) = \operatorname*{Top}_{K,\, d\in\mathcal{C}_B(q)} s_r(q,d). \tag{14}$$

The evaluated pipeline uses $L = 128$, $B = 1024$, and $K = 100$ with exact flat inner-product search in Faiss (Douze et al., 2024), then ranks $\mathcal{C}_K(q)$ by $s(q,d)$. Exactness concerns slot lookup and scoring within the shortlist; finite candidate budgets do not guarantee exhaustive document-level ranking under s_r or s.

Routing and value attention use the same routing vectors, with different temperatures τ_r and τ_v. The similarity matrix may be recomputed between stages; caching it is not required. The final score is directional from query to document despite sharing encoder parameters.

Each document representation stores $m(d_r + d_v)$ scalars, or 8192 with the configuration in Table 4, excluding index copies and metadata. Computing the routing similarities costs $O(m^2 d_r)$, and value aggregation and comparison cost $O(m^2 d_v)$ per candidate. These costs exclude offline document encoding, online query encoding, index search, and data transfer. In particular, exact flat slot search over N documents costs $O(Nm^2 d_r)$ per query; the bounded shortlist limits subsequent document scoring and value attention.

A.3 TRAINING OBJECTIVES AND GRADIENT PATHS

Weak supervision. The Slot Encoder uses linear heads during weak training; the residual value MLP is added for distillation. Let $\mathcal{B}_w = ((q_i, d_i))_{i=1}^b$ be the global batch of weak pairs. For each query, $J_i \subseteq \{1, \dots, b\}$ indexes candidate document occurrences and includes its paired positive i. Writing $\mathcal{J} = (J_i)_{i=1}^b$, define

$$\mathcal{L}_{\mathrm{CE}}(f, \mathcal{J}) = -\frac{1}{b} \sum_{i=1}^{b} \log \frac{\exp(f(q_i, d_i)/T_w)}{\sum_{j \in J_i} \exp(f(q_i, d_j)/T_w)}, \qquad T_w = 0.05. \tag{15}$$

The paired occurrence is the sole positive target; repeated documents retain separate indices. Before selecting hard negatives, we apply a false-negative mask to the full global pool. For each query, it removes the highest-scoring negatives satisfying $s_r(q_i, d_j) > s_r(q_i, d_i) + 6.25 \times 10^{-5}$, up to $\min(\lceil 0.25(b-1) \rceil, \max(0, b - 1 - 16))$ occurrences. The mask is active from zero-based weak-training update $u = 5$.

For each query, $J_{r,i}$ contains all unmasked indices, and $J_{c,i}$ contains i plus up to 256 remaining indices with the highest routing scores. Writing $\mathcal{J}_r = (J_{r,i})_{i=1}^b$ and $\mathcal{J}_c = (J_{c,i})_{i=1}^b$, the two contrastive terms are $\mathcal{L}_{\mathrm{route},u} = \mathcal{L}_{\mathrm{CE}}(s_r, \mathcal{J}_r)$ and $\mathcal{L}_{\mathrm{comb},u} = \mathcal{L}_{\mathrm{CE}}(f_u, \mathcal{J}_c)$, where

$$f_u(q, d) = (1 - \beta_u)\, \mathrm{sg}(s_r(q, d)) + \beta_u s_v(q, d). \tag{16}$$

Here sg preserves the forward value and blocks gradients. Only the additive routing term is detached; routing-based attention inside s_v remains differentiable. The configured schedule is $\beta_u = 0.25 a_u$ and $\mu_u = a_u$, with $a_u = \min(1, \max(0, (u - 45)/90))$ and u the global weak-training update counter. Full weights apply from $u = 135$.

For an input x, the diversity penalty is

$$\mathcal{D}_{\mathrm{slot}}(x) = -\frac{1}{m(m-1)} \sum_{i \neq j} \log \Big[\max\Big(1 - \big\langle \widehat{\mathbf{r}}_i^x, \widehat{\mathbf{r}}_j^x \big\rangle^2, \epsilon\Big) \Big], \qquad \epsilon = 10^{-6}. \tag{17}$$

We average this penalty over query and document inputs to obtain $\mathcal{L}_{\mathrm{div}}$ and use $\eta = 0.1$ in Equation 6. For this regularizer, normalization uses $\widehat{\mathbf{r}} = \mathbf{r} / \max(\|\mathbf{r}\|_2, 10^{-6})$.

Listwise distillation. Throughout adaptation, the backbone and routing head are frozen; the shared slot construction and value-head parameters remain trainable. The residual MLP is specified in Appendix A.1. The cross-encoder teacher is used only during training.

Let $\mathcal{C}(q)$ be the sampled candidate set for query q and $z_T(q, d)$ the teacher's relevance logit margin (yes minus no). Define $\widetilde{s}_v(q, d)$ as the value score in Equation 5 with $\mathrm{sg}(\mathbf{C}(q, d))$ in Equation 4. The student logit is

$$\ell(q, d) = \mathrm{sg}(s_r(q, d)) + \gamma \widetilde{s}_v(q, d), \qquad \gamma = \frac{\beta}{1 - \beta} = \frac{1}{3}. \tag{18}$$

Teacher and student distributions are normalized over the same candidate set:

$$p_T(d \mid q) = \frac{e^{z_T(q,d)/T_T}}{\sum_{d' \in \mathcal{C}(q)} e^{z_T(q,d')/T_T}}, \qquad p_S(d \mid q) = \frac{e^{\ell(q,d)/T_S}}{\sum_{d' \in \mathcal{C}(q)} e^{\ell(q,d')/T_S}}. \tag{19}$$

For a query batch $\mathcal{B}_{\mathrm{dist}}$, we minimize

$$\mathcal{L}_{\mathrm{dist}} = \frac{1}{|\mathcal{B}_{\mathrm{dist}}|} \sum_{q \in \mathcal{B}_{\mathrm{dist}}} \sum_{d \in \mathcal{C}(q)} p_T(d \mid q) \log \frac{p_T(d \mid q)}{p_S(d \mid q)}. \tag{20}$$

We use $T_T = 2$ and $T_S = 0.10$, without a temperature-squared multiplier. The forward value of ℓ equals $s/0.75$, equivalent to applying a temperature of 0.075 directly to s. Detachment blocks distillation gradients through the routing score and attention weights; gradients still reach the trainable Slot Encoder parameters through query and document values.

Distillation uses fixed candidate lists. Each update samples a target of 32 candidates per query, retaining all labeled positives present in the list. Subject to the remaining budget, we select up to 16 negatives in stored routing rank order, up to 8 more by largest absolute teacher–routing rank disagreement, and fill the rest uniformly from remaining negatives. Positives may exceed the nominal budget. All student scores used in the loss are recomputed with the current model.

Retrieval replay. Replay independently samples query–positive pairs from the adaptation data. Let $\mathcal{B}_{\text{ret}}$ be the batch of query occurrences and $\mathcal{D}_B$ the pool of unique document identifiers in its paired documents, qualified by their data source to distinguish identical IDs across datasets, and let $\mathcal{P}_B(q) \subseteq \mathcal{D}_B$ contain all known positives for q present in that batch. Positives come from relevance labels. The retrieval objective is

$$\mathcal{L}_{\text{ret}} = -\frac{1}{|\mathcal{B}_{\text{ret}}|} \sum_{q \in \mathcal{B}_{\text{ret}}} \log \frac{\sum_{d \in \mathcal{P}_B(q)} \exp(s_r(q,d)/T_r)}{\sum_{d \in \mathcal{D}_B} \exp(s_r(q,d)/T_r)}, \qquad T_r = 0.05. \tag{21}$$

Duplicate document identifiers count once; additional known positives enter the numerator rather than serving as negatives. Replay can use labeled positives absent from the saved reranking candidate lists.

At one-based joint optimization update t, the weight in Equation 8 is

$$\lambda_t = \mathbf{1}[(t-1) \bmod 8 = 0]. \tag{22}$$

Replay occurs every eight updates starting at update 1, with coefficient 1. The distillation and nominal replay batches contain 256 queries and $16{,}384$ query–positive pairs, respectively, and each loss is averaged independently. On replay updates, both gradients are accumulated before one optimizer step. Retrieval gradients pass through the frozen routing head into shared Slot Encoder parameters; distillation reaches them through values. This provides direct retrieval supervision during adaptation, without constraining routing vectors to remain identical across updates.

B TRAINING DATA AND EVALUATION DETAILS

B.1 TRAINING DATA PREPARATION

The two stages use different forms of supervision: weak query–document pairs train the shared representation, followed by teacher-scored candidate lists and annotated positives for distillation and retrieval replay.

Weak query–document pairs. The weak-stage recipe samples from LightOn's curated pre-training collection (Sourty et al., 2026).[1] This collection reconstructs the English mGTE data recipe (Zhang et al., 2024) and includes title/body, title/abstract, question/answer, citation, and other text-pair sources. Our recipe targets 30 million accepted pairs across training and validation. Initial quotas across 34 source subsets are proportional to the square root of their advertised sizes, capped at source availability and 20% of the total. Unfilled quotas carry forward to subsequent sources, so the final mixture can differ from this initial allocation. Each source is streamed with a 100,000-row shuffle buffer; the base random seed is 1931.

Preparation collapses whitespace, caps each text at 4,000 characters, and requires at least 8 query and 40 document characters. We reject normalized identical query/document texts, deduplicate exact normalized pairs within and across sources, and exclude exact normalized matches to the configured evaluation-proxy query and document files. This is a limited exclusion check, not a complete benchmark-overlap audit. No additional threshold is applied to LightOn's supplied similarity scores. Each accepted pair is assigned to validation with probability 0.002. Thus, the 30M figure denotes the preparation target before this split, not the number of training-only pairs or optimizer presentations. We will release the weak-stage sampling script on GitHub.

Distillation and retrieval replay. The adaptation pool contains 500,000 training queries from nine sources: MS MARCO (175,000; Bajaj et al., 2016), Natural Questions (NQ; 65,000; Kwiatkowski et al., 2019), TriviaQA (40,000; Joshi et al., 2017), SQuAD v2 (20,000; Rajpurkar et al., 2018), HotpotQA (40,000; Yang et al., 2018), and FEVER (60,000; Thorne et al., 2018) from LightOn's separate fine-tuning collection (Sourty et al., 2026);[2] StackExchange duplicate question titles (50,000);[3] SciRepEval search (25,000; Singh et al., 2023); and PubMedQA question–abstract pairs (25,000; Jin et al., 2019). Repeated annotations are merged into query groups with their positive document sets. We reserve an additional 5,000 development queries before candidate mining, exclude normalized query duplicates across splits, and retain queries regardless of whether routing retrieves a relevant document. Training and development queries can share documents. The prepared corpora combine annotated positives with source negatives or sampled background documents; all known development positives are present. Each source is searched independently.

A fixed checkpoint mines 100 candidates per query using IVF-Flat slot search (up to 4,096 lists and 64 probes), a coarse shortlist, and exact routing rescoring. Qwen3-Reranker-0.6B (revision `e61197ed`, 512-token limit) scores these candidates for distillation (Zhang et al., 2025); labeled positives are not inserted into the retrieved lists. Replay independently samples annotated query–positive anchors from these same 500,000 training queries, including positives outside the saved candidate lists. Both objectives therefore use the adaptation pool: teacher scores supervise relative relevance, while positive annotations provide continued retrieval supervision.

B.2 DEVELOPMENT DATA AND COMPARISON PROTOCOL

The 5,000-query split from Appendix B.1 is used for development evaluation across nine prepared corpora totaling 3.41 million documents. Unlike candidate mining, fresh development and BEIR evaluation use exact flat slot search. Algorithm 1 uses $L = 128$ hits per query slot, a $B = 1024$-document shortlist, and $K = 100$ routing-selected candidates. QK uses s_r; QKV uses $0.75s_r + 0.25s_v$ on the same candidates, without inserting relevant documents. These settings are fixed across sources and BEIR tasks. Query/document token limits are 128/384.

[1] LightOn curated pre-training dataset.

[2] LightOn fine-tuning dataset.

[3] Title–title pairs from the sentence-transformers embedding-training-data collection, revision `03987af`.

Table 5 reports the actual corpus sizes and source-level ranking results. The development set includes the 1,152-query training monitor and is used for model selection. It is not an independent test set. FEVER and NQ account for most of the aggregate value gain; values improve six of nine sources. Jina exceeds the combined model on eight sources, with FEVER the exception.

Table 5: Prepared development sources and nDCG@10 multiplied by 100. QK and QKV use the final BreadBowl-Embed checkpoint. The first six sources are prepared LightOn subsets, not the official benchmark splits.

Source	Queries	Documents	QK	QKV	Qwen	Jina
FEVER	600	259,716	73.84	88.35	78.81	84.09
HotpotQA	400	387,215	65.60	68.34	68.26	74.08
MS MARCO	1,750	999,584	49.97	49.04	52.96	56.54
NQ	650	472,965	35.46	43.87	38.47	46.87
SQuAD v2	200	18,978	69.91	70.23	70.34	72.89
TriviaQA	400	754,021	50.92	49.99	50.08	52.29
PubMedQA	250	61,208	92.68	92.32	93.70	95.45
SciRepEval search	250	299,469	68.22	71.19	71.37	72.77
StackExchange	500	157,759	36.99	37.42	36.27	40.25

All three systems use BF16 encoder inference and float32 exact flat inner-product indexes over normalized vectors. Qwen and Jina use last-token pooling and their full embedding width. Qwen receives a query instruction and raw document text; Jina uses its `Query:` and `Document:` prefixes. The 128/384-token limits include these prefixes; different tokenizers need not preserve identical text at truncation. We pin Qwen and Jina to revisions `97b0c614` and `6856e76b`, respectively. Query, corpus, and label identities are verified across evaluations. Unlike the baselines' exhaustive cosine ranking, BreadBowl-Embed uses exact slot lookup followed by bounded document selection.

B.3 Adaptation and checkpoint selection

The KL-only and replay runs share an initialization, training queries, candidate lists, random seed, and optimizer settings (Table 6). Within the adaptation stage, both runs continue from the same earlier distillation checkpoint, also used to mine the fixed candidate lists. This initializer has 1,600 KL updates with 64 queries per update on 20,000 queries (10,000 LightOn and 10,000 PubMedQA-derived); its checkpoint SHA-256 prefix is `4b80708a670c2e15`. Learned weights, including the residual value MLP, are retained; optimizer and scheduler states start fresh for the 500,000-query continuation.

Exact accepted weak train/validation counts and the weak dataset revision were not recovered; 30M remains a preparation target. The schedules in Table 6 retain their configured horizons rather than being shortened to the selected checkpoints. Appendix A.3 specifies the losses and replay gradient paths.

Training resources. Weak training used eight H200 GPUs for approximately 8 hours (64 H200 GPU-hours). The selected 4,000-step checkpoint of the KL+replay run required approximately 10 hours on eight RTX 5090 GPUs (80 RTX 5090 GPU-hours), summing recorded training-update durations. This timing excludes evaluation and checkpoint saving. These figures describe the reported runs; compute for producing their initial checkpoints, data preparation, teacher scoring, and experimental development is not quantified here.

Checkpoint selection. For model selection, we rank the same saved 100 candidates per development query at every checkpoint, using the lists and teacher scores prepared as described in Appendix B.1. Table 7 reports the complete replay checkpoint sweep. We choose step 4,000 by combined-score macro nDCG@10 before running the final BEIR suite. Scores plateau over the last three checkpoints; steps 4,000 and 4,500 differ by less than 0.002 nDCG points. On these fixed lists, the teacher reaches 69.80 versus the student's 63.40, so the value benefit does not imply teacher parity. All main-paper development comparisons instead rerun corpus retrieval with each checkpoint. Fixed-list evaluation cannot measure changes in candidate recall, and its teacher scores are not reused for fresh candidates. The KL-only run has a larger QKV–QK gap but lower final

Table 6: Training recipe. The adaptation column describes the 500,000-query continuation within the second training stage; replay is omitted in the KL-only arm. Exposure counts are nominal presentations, not unique examples. All runs use BF16 computation, FP32 trainable parameters and optimizer moments, and AdamW defaults $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$.

Setting	Weak training	Adaptation continuation
Initialization	Qwen3.5-0.8B-Base, revision `a9a407bc`	Shared distillation checkpoint described above
Trainable modules	Backbone, slot extractor, linear routing/value heads	Slot extractor and value modules; backbone/routing head frozen
Peak learning rate	10^{-5} for all modules	10^{-5} slot extractor; 10^{-4} value modules
AdamW weight decay / gradient-norm clip	0.01 / 1	0 / 1
Learning-rate schedule	37 warmup updates, then linear decay; horizon 1,710	293 warmup updates, then cosine decay; horizon 5,860
Global batch	16,384 pairs: $2{,}048 \times 8$ GPUs; accumulation 1	256 queries: 4×8 GPUs $\times 8$ accumulation
Retrieval replay	—	16,384 pairs every 8 updates, starting at 1; weight 1
Encoder microbatch	64 inputs with gradient caching	Replay: 384 inputs; distillation: 4 query groups per GPU
Training seed	1677	20260924
Updates used	1,000	4,000 continuation updates
Nominal exposure	16.384M pair presentations	1.024M KL-query presentations; replay adds 8.192M pair presentations

quality than replay (Table 3); the value increment must be considered alongside the routing score it refines.

Table 7: Checkpoint selection on fixed development candidates. Scores are source-macro nDCG@10 multiplied by 100. The saved candidates and teacher scores are identical at every step.

Update	QK	QKV	Teacher
500	59.921	62.033	69.795
1,000	59.992	62.741	69.795
1,500	60.110	62.907	69.795
2,000	60.235	63.074	69.795
2,500	60.334	63.197	69.795
3,000	60.391	63.366	69.795
3,500	60.416	63.394	69.795
4,000	60.359	63.399	69.795
4,500	60.346	63.398	69.795

B.4 BEIR PROTOCOL AND SCOPE

We use official full corpora and relevance judgments with test splits except MS MARCO development. Metrics are computed with `pytrec_eval` (Van Gysel & de Rijke, 2018), including graded relevance for nDCG. Documents combine title and body as `title: text` when a title is present. Identical query/document IDs are excluded before shortlist selection. No task-specific mixture weights or retrieval budgets are tuned on the reported results. We retain all official queries and judgments, including five ArguAna queries whose sole relevant document is absent from the official corpus; these queries receive zero scores.

CQADupStack's 12 forum means are averaged into one task before computing the 15-task macro average. Excluding MS MARCO gives 49.65 QK and 52.77 QKV nDCG@10, with recall@100 of 66.91. Recall measures the fraction of judged relevant documents retrieved, rather than whether at least one is found; its scale consequently differs across tasks with many versus few positives.

Some BEIR domains occur in the adaptation data, and earlier SciFact and NFCorpus evaluations informed development. The final suite evaluates a fixed checkpoint, but is not uniformly zero-shot or wholly untouched. Earlier weak-training overlap has not been fully audited. The QK–QKV comparison removes the value score at inference from one trained model; it does not compare separately trained routing-only and combined architectures.

B.5 Comparisons with released embedding models

Published BEIR results. Table 8 provides context for absolute effectiveness. It includes all final released models in Table A17 of Akram et al. (2026), using the more precise means in their Table 4. The 15 named tasks match our suite, but the reports do not control input formatting, truncation, checkpoint revisions, or retrieval settings. Their paper combines reported leaderboard results and its own evaluations. These numbers therefore provide a literature comparison, not matched reruns or an isolation of architectural differences.

Table 8: BEIR context: nDCG@10 multiplied by 100. Published means and parameter counts are from Akram et al. (2026), version 1, Table 4; Table A17 gives the task breakdown. The lower block is evaluated in this work. Protocols differ across reports. *BreadBowl-Embed's parameter entry denotes nominal backbone size, excluding its pooling and output heads.

Model	Parameters	BEIR mean
Published results		
Qwen3-Embedding-4B	4B	61.58
Jina embeddings v4	3.8B	53.97
Qwen3-Embedding-0.6B	596M	55.52
Jina embeddings v3	572M	53.17
Snowflake Arctic Embed L v2	568M	55.22
Multilingual E5 Large Instruct	560M	52.74
Jina embeddings v5 text-small	677M	56.67
KaLM multilingual mini instruct v2.5	494M	55.00
Voyage 4 nano	480M	49.93
EmbeddingGemma 300M	308M	53.69
Jina embeddings v5 text-nano	239M	56.06
Evaluated in this work		
BreadBowl-Embed, QK	0.8B*	48.47
BreadBowl-Embed, QKV	0.8B*	51.54

Reruns on prepared development corpora. We evaluate Qwen3-Embedding-0.6B (Zhang et al., 2025) and Jina v5 text-small retrieval (Akram et al., 2026) on the same development queries, corpora, labels, and token limits, using their native input formatting. Each baseline performs exact cosine retrieval with its full 1,024-dimensional embeddings; each system retrieves its own candidates. Table 9 places BreadBowl-Embed between these two models. Adding values raises its nDCG@10 above Qwen's, while Jina remains stronger in both ranking and recall. Training data, model size, and training compute are not controlled across models.

Table 9: Reruns of released embedding models and BreadBowl-Embed on the prepared development corpora. All metrics are source-macro averages multiplied by 100. These are end-to-end retrieval comparisons, not shared-candidate reranking comparisons or official BEIR scores.

Model / score	nDCG@10	Recall@100
Qwen3-Embedding-0.6B	62.25	87.42
Jina v5 text-small retrieval	66.14	90.73
BreadBowl-Embed, QK	60.40	87.19
BreadBowl-Embed, QKV	63.42	87.19

B.6 UNCERTAINTY AND COMPUTATIONAL SCOPE

For fresh development results, we use 5,000 paired query-bootstrap resamples with seed 42, resampling within each source while keeping its query count fixed, and report 95% percentile intervals. The QKV–QK gain is 3.02 points with interval $[2.49, 3.54]$; the replay–KL final-score gain is 0.84 with interval $[0.45, 1.24]$. These intervals condition on the nine sources and selected checkpoints, do not capture training-seed variation, and do not adjust for development-based model selection. BEIR results and recall comparisons are reported as point estimates.

The evaluations establish effectiveness under the stated candidate budgets. They do not measure latency or throughput relative to a retriever followed by a cross-encoder. Avoiding candidate text re-encoding follows from the architecture, while practical efficiency also depends on multi-vector indexing, representation storage, and data transfer (Appendix A.2).

C QUALITATIVE INSPECTION OF SLOT POOLING

We inspect the selected KL+replay checkpoint on one pair written before examining its attention weights. The query is "In which city is the Eiffel Tower, and when was it completed?" The passage describes its location in Paris, construction from 1887 to 1889, the World's Fair, Gustave Eiffel's company, and distractors involving the Louvre and the Colosseum in Rome. The formatted inputs contain 22 query and 66 document tokens, with no truncation. They are encoded independently in evaluation mode.

Figures 4 and 5 show the arithmetic mean of the four heads' token-pooling probabilities for all 16 slots. These weights precede the routing and value projections; they are distinct from query–document value attention. All non-padding tokens, including the query instruction, are retained. Indices are zero-based.

Query slot 13 assigns 47.4% to `completed`; document slot 13 assigns 30.5% to the final digit of 1889, 20.0% to `Fair`, and 11.4% to the final digit of 1887. Document slot 14 assigns 21.7% to `Rome`. Matching is not restricted to equal slot indices: query slot 12 has its highest routing cosine with document slot 11 (0.933).

The maps describe pooling of contextual states, not causal word importance; a final digit can carry information about earlier digits. Head averages also do not represent the learned output projection. This single example illustrates variation across slots, without establishing stable semantic roles, an effect of replay, or a causal explanation of reranking gains.

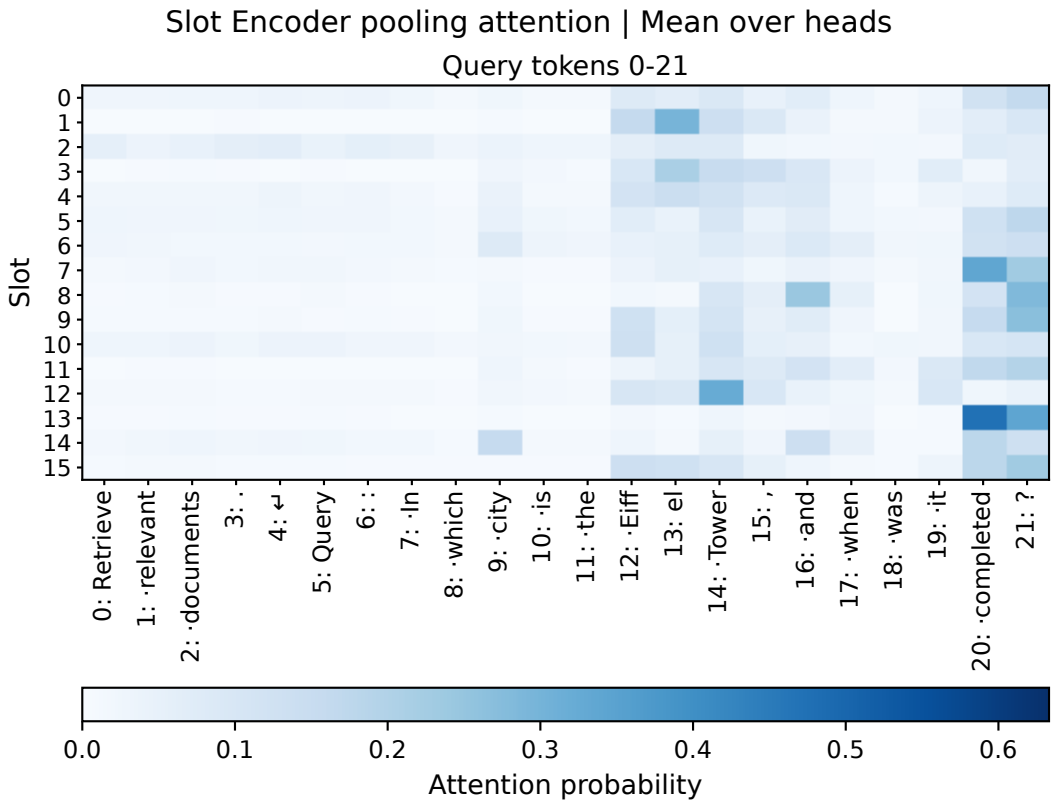

Figure 4: Slot Encoder pooling over query tokens, averaged across four heads. Rows sum to one. Labels retain token positions; a middle dot denotes a space and an arrow a newline. The instruction prefix is included. Both figures use the same probability scale.

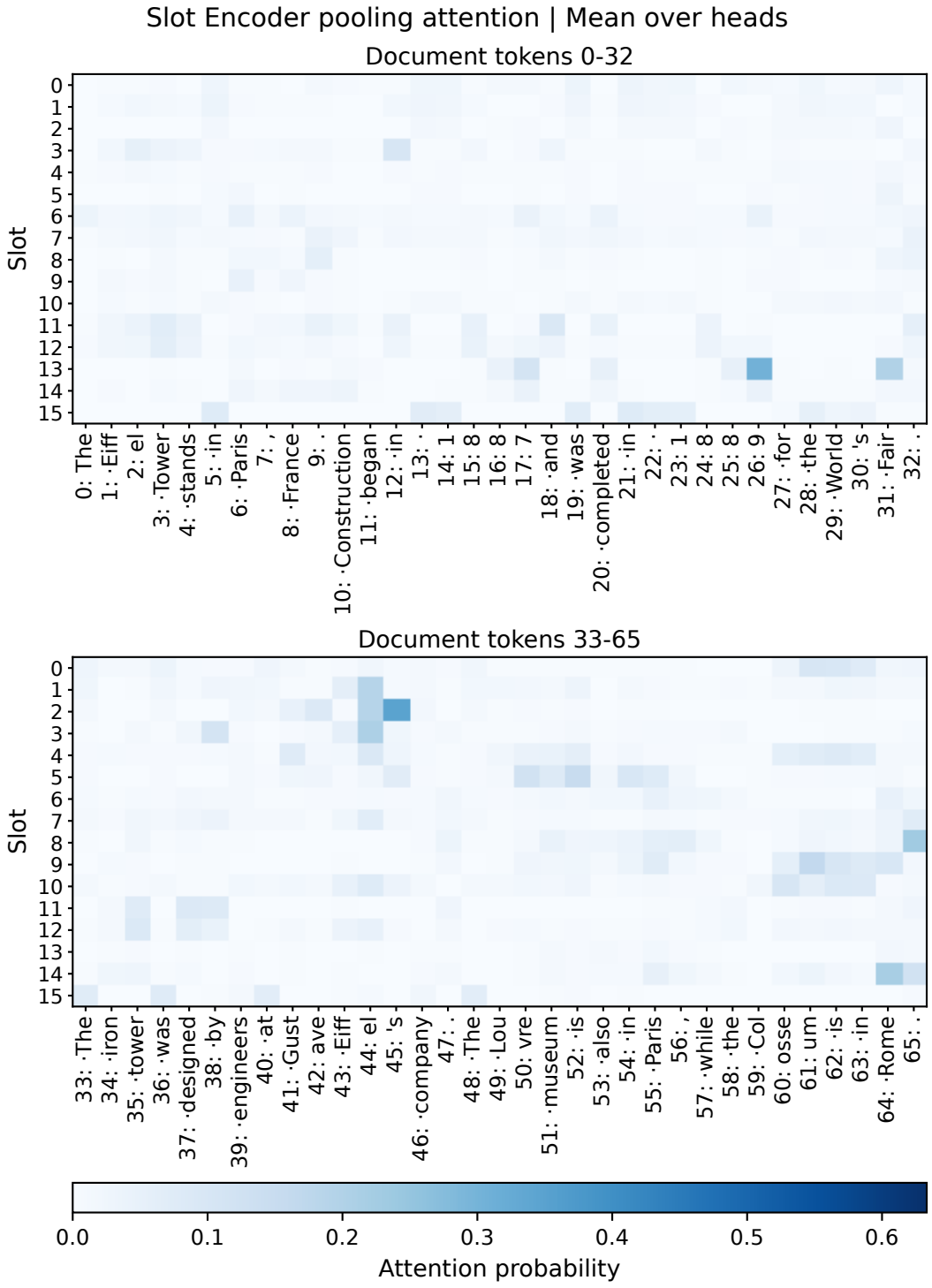

Figure 5: Slot Encoder pooling over document tokens, averaged across four heads and split into contiguous ranges. Each row sums to one across both panels. Token 26 is the final digit of 1889. No tokens are omitted or probabilities renormalized.