PROBABILITY-WEIGHTED POOLING FOR UNIFIED CONTEXT COMPRESSION AND RERANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) systems commonly use separate models for context compression and document reranking, requiring redundant encoding of the same query-document pairs. Unifying these stages allows a shared encoder to rank documents and prune the contents in a single pass. However, unified ranking requires choices in how to pool token representations and supervise the resulting ranking score. In this work, we introduce probability-weighted pooling, which weights document-token representations by the relevance probabilities predicted by the compression head. We theoretically prove that, as the predicted relevance weights in probability-weighted pooling become more accurate, its ranking score converges to that obtained using ground-truth evidence weights. We conduct comprehensive experiments across six benchmarks, comparing three pooling mechanisms and two ranking targets. Although the average improvement is modest, the gains are more pronounced on PubMed, where queries often involve multiple relevant documents, consistent with the statistically significant advantage observed in our aggregate comparisons. Document-level supervision yields a larger improvement over span-or-penalty supervision. These results highlight the greater impact of ranking supervision relative to pooling choice.

1 INTRODUCTION

Retrieval-augmented generation (RAG), where a language model answers a query using documents fetched from an external corpus, has been applied to a diverse range of real-world applications such as open-domain question answering, enterprise search and biomedical literature search. A deployed RAG system is a pipeline of several stages. A first-stage retriever returns a pool of candidate documents, a cross-encoder reranker scores each candidate to select the best few, and a context compressor prunes the selected documents down to the sentences that support an answer, before anything is passed to the generator.

One of the main challenges in deploying such a pipeline is that the reranker and the compressor are separate neural models, and each runs its own forward pass over every query-document pair. For a pool of 50 candidates, this means 50 reranker passes followed by up to 50 compressor passes, where both models encode exactly the same text. This is wasteful, because the two stages read the same input and answer closely related questions about it. A reranker estimates how relevant a document is, while a compressor estimates which parts of it are relevant. Recently, Provence (Chirkova et al., 2025) showed that a single encoder can be trained to do both, producing sentence-level pruning decisions and a document-level relevance score from one forward pass. We refer to this setting as *unified-rerank*: the model prunes each document and also produces the score that selects and reorders the top-k documents reaching the generator, so no external reranker is needed at all.

Sharing an encoder in this way is only useful if the shared model ranks as well as the dedicated reranker it replaces, and this is where the design of the ranking head becomes important. The two tasks read the same hidden states but require different things from them. Compression is a token-level decision and produces a relevance probability at every token position. Ranking, in contrast, requires a single score for the whole document, so the sequence of token representations must be reduced to one vector before the score can be computed. This reduction is a design choice that has no analogue in a standalone reranker, and to our knowledge it has not been studied. Provence obtains its ranking representation from the encoder's `[CLS]` (classification) token. This special

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.

token is placed at the start of the query–document sequence; its contextual hidden state aggregates information from the input and is used for sequence-level prediction (Devlin et al., 2019). This is a reasonable default, but it does not explicitly use the token-level relevance probabilities already computed by the compression head.

Our key idea is to exploit the compression head's own per-token probabilities as the pooling weights for the ranking score. In a unified model these probabilities are available for free, because the model computes them anyway in order to decide which sentences to keep. We use these probabilities to weight the document-token representations passed to the ranking classifier, coupling the ranking representation to the compression predictions. We call this probability-weighted pooling. A second design choice concerns what the ranking head is trained to predict, where the two natural targets are the relevance of the whole document and the relevance of the supporting span alone. We compare these design choices empirically and analyze the consistency of document-token pooling with oracle evidence weights.

For comparison, uniform mean pooling gives every document token the same weight, whereas oracle pooling assigns weight only to tokens marked as evidence. We use the latter as an analytical reference rather than as a deployable ranking method.

Our main contributions are:

- We introduce probability-weighted pooling for unified context compression and document reranking. It uses the compression head's token-relevance probabilities to construct the document representation, and we compare it with mean and `[CLS]` pooling as well as document-level and span-or-penalty ranking targets (Section 4.1; Section 4.2).
- We establish a conditional oracle-pooling consistency bound for fixed token representations and a fixed Lipschitz ranking readout (Section 5). The result relates the student's score to the score obtained from evidence weights; it is an internal consistency statement rather than a guarantee of teacher-score recovery or downstream accuracy.
- We evaluate the ranking-head choices on six question-answering datasets. The ranking target has the larger effect: document-level supervision consistently outperforms span-or-penalty supervision, while probability-weighted pooling gives a smaller improvement over mean pooling under the primary LLM-judged accuracy metric and does not significantly separate from `[CLS]` pooling. Each variant uses one trained checkpoint, so these results describe the evaluated checkpoints rather than the architectures in general.

2 Related Work

Research on efficient retrieval-augmented generation considers both which documents to select and which parts of them to retain. These decisions link the relevance of individual text spans to the usefulness of a whole document. We review this work in three groups: context compression and reranking, evidence aggregation, and the role of context in ranking supervision.

Context compression and reranking. Context compression methods aim to shorten the input to a language model while preserving information needed for the task. Token selection is one approach: LLMLingua-2 uses a bidirectional encoder to decide which tokens to keep, taking their surrounding text into account (Pan et al., 2024). A shared model can also use the same encoding to select documents and prune their contents. Provence follows this approach by jointly predicting document relevance and token-level pruning decisions (Chirkova et al., 2025). Building on this architecture, we study how the ranking head can use the compression head's evidence predictions when forming a document representation, and how that head should be supervised.

Evidence aggregation. Earlier retrieval models also connect evidence in smaller text units with document relevance. Sentence-level scoring and joint document–snippet ranking use the relevance of parts of a document to inform the overall ranking (Akkalyoncu Yilmaz et al., 2019; Pappas & Androutsopoulos, 2021). For documents that exceed an encoder's input limit, passage aggregation combines information from separately encoded pieces; PARADE studies how those representations should be brought together (Li et al., 2024). A related idea appears in attention-based multiple-instance learning, where a prediction for a collection of instances is made from a weighted combination of their representations (Ilse et al., 2018). Our method uses this general form of pooling.

Its weights are the token-relevance probabilities produced by a compression head trained to identify evidence, so the same predictions guide both pruning and the document representation used for ranking.

Context and ranking supervision. Extracting relevant text does not necessarily preserve all the context needed to interpret it. A sentence may contain a pronoun whose referent occurs earlier in the document, or rely on a definition given elsewhere. Work on decontextualization studies how such sentences can be rewritten to stand on their own (Choi et al., 2021). The same concern arises when a teacher model supplies relevance scores for training a ranking head: the teacher receives different information when it scores a supporting span and when it scores the full document. We examine this distinction by comparing full-document supervision with span-or-penalty supervision, which uses teacher scores for supporting spans when available and a fixed low target otherwise.

3 Problem Formulation

In unified context compression and reranking, the same encoder must decide which document tokens contain evidence and also assign a single relevance score to the document. The problem is therefore how to turn token-level evidence predictions into a document representation without introducing a second encoder or an independent ranking mechanism.

Let q be a query and d a retrieved document. A shared encoder produces contextual states $h_1, \dots, h_L \in \mathbb{R}^m$ for the query–document sequence. Let $m_i \in \{0, 1\}$ indicate whether position i is a document token, and define $\mathcal{D} = \{i : m_i = 1\}$ and $N = |\mathcal{D}|$. The compression head produces a logit c_i and an evidence probability

$$p_i = \sigma(c_i), \qquad i \in \mathcal{D}.$$

The ranking head maps the document states to one representation and then to a scalar score. For token-based pooling, let $a = (a_i)_{i \in \mathcal{D}}$ be a probability distribution over document tokens and define

$$\bar{h}_a = \sum_{i \in \mathcal{D}} a_i h_i, \qquad s_a = g(\bar{h}_a),$$

where g is the ranking readout. We study uniform mean pooling,

$$a_i^{\text{mean}} = \frac{1}{N},$$

and probability-weighted pooling,

$$a_i^{\text{prob}} = \frac{p_i}{P}, \qquad P = \sum_{j \in \mathcal{D}} p_j,$$

when $P > 0$; the implementation replaces P by $\max\{P, \epsilon\}$ with a small $\epsilon > 0$ for numerical stability. The third pooling mechanism uses the encoder's `[CLS]` representation, $d_{\text{CLS}} = \text{Pool}(h_{\text{CLS}})$, and scores it with the same ranking head.

The ranking target is either the teacher score for the full document, $t_{\text{doc}}(q, d)$, or the span-or-penalty target

$$t_{\text{sop}}(q, d) = \begin{cases} t_{\text{span}}(q, d), & \text{if a supplied supporting span exists,} \\ \text{P}_1(t_{\text{span}}) - \kappa \ \text{sd}(t_{\text{span}}), & \text{otherwise,} \end{cases} \qquad \kappa = 0.5,$$

where P_1 and sd are computed over the training examples with supplied supporting spans. Thus, the experiment varies the pooling rule and the ranking target while keeping the shared encoder, compression task, retrieval pipeline, and evaluation protocol fixed. The methodology section gives the implementation details, and the theory section analyzes the token-based pooling rules relative to evidence-weighted oracle pooling.

4 Methodology

A unified-rerank model jointly performs context compression and document reranking from a shared encoder pass over each query–document pair. The compression head predicts token-level relevance

Variant	Pooling operator	Ranking target
CLS	`[CLS]` state	Full-document teacher score
Probability-weighted	Compression probabilities	Full-document teacher score
Mean	Uniform document-token mean	Full-document teacher score
Span-or-penalty	Compression probabilities	Span score or derived no-span penalty

Table 1: Ranking-head configurations. The span-or-penalty checkpoint isolates the ranking-target change by retaining probability-weighted pooling.

probabilities for context pruning, while the ranking head produces a document-level score for selecting and reordering retrieved documents. We focus on how to aggregate token representations and how to supervise the resulting score. We study these choices in Section 4.1 and Section 4.2, respectively, followed by multi-block score aggregation (Section 4.3) and training data (Section 4.4).

4.1 Ranking-head pooling mechanisms

Let $h_1, \ldots, h_L$ denote the final-layer hidden states produced by the encoder for an input sequence of length L, comprising the query, retrieved document, and special tokens. Here, `[CLS]` and `[SEP]` are special tokens for sequence representation and segment separation, respectively. We define a document mask $m_i \in \{0, 1\}$, where $m_i = 1$ if and only if position i corresponds to a document token, excluding the query, special tokens, and padding. The compression head produces a logit c_i at each position, with relevance probability $p_i = \sigma(c_i)$. The ranking head constructs a representation d by pooling the encoder hidden states and maps d to a document-level ranking score. We consider three pooling mechanisms: CLS pooling, probability-weighted pooling, and mean pooling.

CLS pooling. We use the encoder's pooling layer to obtain a sequence-level representation from the hidden state h_{CLS} of the leading `[CLS]` token (Devlin et al., 2019). For the DeBERTa encoder used in our experiments, the pooling layer applies a learned linear transformation followed by GELU: $d_{\text{CLS}} = \text{GELU}(W_p h_{\text{CLS}} + b_p)$. This formulation matches the teacher cross-encoder's pooling mechanism, allowing its ranking classifier to initialize the corresponding student parameters. While h_{CLS} contextualizes the entire input sequence, CLS pooling does not explicitly aggregate document-token representations or incorporate the relevance probabilities predicted by the compression head.

Probability-weighted pooling. Rather than relying on a single token representation, we aggregate all document-token hidden states using the relevance probabilities predicted by the compression head as weights: $d_{\text{prob}} = \frac{\sum_i m_i p_i h_i}{\max\{\sum_i m_i p_i, \epsilon\}}$, where m_i restricts pooling to document tokens, p_i determines the contribution of each token, and $\epsilon = 10^{-9}$ ensures numerical stability. When $\sum_i m_i p_i > \epsilon$, d_{prob} is a convex combination of the document-token hidden states weighted by their predicted relevance. Hence, the ranking representation explicitly incorporates the compression head's predictions.

Mean pooling. As the third variant, we use mean pooling, which assigns equal weight to every document token, as in Reimers & Gurevych (2019), and does not use the compression head's relevance predictions: $d_{\text{mean}} = \frac{\sum_i m_i h_i}{\max \sum_i m_i, \epsilon}$. For inputs containing at least one document token, the denominator equals the number of document tokens. This variant provides a controlled ablation of probability-weighted pooling, differing only in the token weighting used for aggregation.

In all three variants the ranking score is $s = W_r \,\text{dropout}(d_*) + b_r$. At inference, dropout is disabled, so the readout of the pooled vector is affine. We train compression against binary sentence-level oracle labels, broadcasting each sentence's label to its tokens. For probability-weighted and mean pooling, the single-logit compression head uses unweighted binary cross-entropy averaged over all positions in each padded input sequence, including query, special, and padding positions with label zero; the document mask affects pooling but does not mask this loss. Training combines compression with an MSE ranking loss against the teacher's relevance score. Appendix A distinguishes this sequence-wide loss from the document-token loss used in the analysis.

4.2 RANKING TARGETS

Independently of the pooling mechanism, the ranking head can be trained against different notions of the target score it should regress towards. The default target, which we call DOC, is simply the teacher cross-encoder's score for the query against the entire retrieved document. An alternative, span-or-penalty, instead distills the cross-encoder's score for just the oracle-selected supporting span when one exists, and falls back to a fixed low penalty score for documents that have no supporting span at all.

We adopt DOC as our primary target for an empirical reason. On two independent training corpora, NQ and MS MARCO (Bajaj et al., 2016), we find that an isolated extracted span is scored systematically lower, and with substantially higher variance, by the same teacher cross-encoder than the same query paired with the full source document. This holds even when the span genuinely contains the supporting evidence. We attribute it to decontextualization effects, such as broken coreference once a span is separated from its surrounding sentences, rather than to any property of the extracted content itself. Using such a score directly as a ranking target means training against a noisier and systematically depressed signal. The span-or-penalty target was designed as a compromise that keeps the informative cases while substituting a stable penalty for the uninformative ones, but as Section 6.3 shows, it remains the weaker of the two targets in most of our downstream evaluations.

4.3 RANKING-SCORE AGGREGATION ACROSS MULTI-BLOCK DOCUMENTS

Every transformer encoder has a fixed maximum input length. When a document's sentences do not fit inside this budget, it is split into multiple contiguous *blocks*. As many whole sentences as fit are packed into each block, sentences are never split part-way through, and each block is scored by an independent forward pass. This splitting is transparent to the compression head. Every block's sentences are pruned independently and the surviving spans from all blocks are concatenated back together into a single pruned document, so compression quality is unaffected by how many blocks a document required.

The ranking score, however, is not treated the same way. In both our own implementation and the the published implementation we build on, only the first block's ranking score is retained for a multi-block document, and every subsequent block's score is computed and then silently discarded. In effect, the decision of whether an entire document is selected for generation through top-k reordering can depend only on evidence in the document's first block, regardless of how many blocks the document actually spans.

Among the datasets we evaluate downstream in this paper (Section 6.1), documents are short enough relative to the backbone's context window that this limitation rarely triggers. Fewer than 1% of documents in any of our evaluation corpora require more than one block. We therefore do not observe a downstream effect of this limitation ourselves. The limitation is nonetheless a property of the architecture rather than of any particular dataset, and we expect it to matter on corpora with substantially longer documents. To address it, we implement a `ranking_aggregation` parameter with three settings. FIRST retains the original behavior and remains the default for backward compatibility, MAX takes the maximum score across all of a document's blocks, and MEAN takes the average across all blocks. We leave a downstream evaluation of this fix on longer-document corpora to future work.

4.4 TRAINING DATA

We train all ranking-head variants described above on MS MARCO (Bajaj et al., 2016) document passages, retrieved with SPLADE-v3 (Lassance et al., 2024) and reranked by the same cross-encoder used as the distillation teacher throughout. We opt for MS MARCO-only training rather than NQ-only training or a mixture of the two, for two reasons. First, the published checkpoint's own documentation is inconsistent about its exact training-data composition, since its model card and code release disagree on whether it was trained on MS MARCO alone or on a mixture with NQ. Second, we find empirically that MS MARCO-only training reproduces the published checkpoint's reported downstream numbers, and exceeds them on several evaluation sets, without requiring us to guess at an undocumented data mixture.

5 THEORETICAL ANALYSIS: ORACLE-POOLING CONSISTENCY

In this section, we define our problem statement and analyze how the compression probabilities used for pooling control the ranking score's deviation from a reference obtained using the supplied evidence mask. The result establishes conditional consistency with this internal reference: for fixed token representations and ranking readout, vanishing document-token prediction loss implies convergence to the oracle-pooled score. Due to the page limit, we defer full definitions, supporting lemmas, and proofs to Appendix A.

5.1 SETUP AND REFERENCE SCORE

We fix one query–document block, its contextual token representations, and the deterministic ranking readout. Let $\mathcal{D} := \{i : m_i = 1\}$ be its document-token positions, $N := |\mathcal{D}| \geq 1$, and $h_i \in \mathbb{R}^d$ their representations; query, special, and padding tokens are excluded. For any nonnegative weights a_i summing to one, the pooled representation is $\bar{h}_a := \sum_{i\in\mathcal{D}} a_i h_i$. All subsequent token sums and extrema range over $\mathcal{D}$.

Let $y_i \in \{0, 1\}$ be the supplied evidence labels, assuming $r := \sum_{i\in\mathcal{D}} y_i \geq 1$. For each $i \in \mathcal{D}$, let $c_i \in \mathbb{R}$ be the compressions logit and $p_i := \sigma(c_i)$ be the predicted evidence probabilities, with $P := \sum_{i\in\mathcal{D}} p_i > 0$. Here, σ is the sigmoid function. Define

$$\tilde{p}_i := \frac{p_i}{P}, \quad \tilde{u}_i := \frac{1}{N}, \quad \tilde{y}_i := \frac{y_i}{r}, \quad \hat{s}_a := g(\bar{h}_a), \quad s_{\text{oracle}} := g(\bar{h}_{\tilde{y}}) \qquad \text{for all } i \in \mathcal{D}.$$

Here, $\tilde{p}$, $\tilde{u}$, and $\tilde{y}$ are the probability-weighted, mean, and oracle pooling distributions. The oracle assigns equal weight to labeled evidence tokens and zero elsewhere. Both scores use the same contextual states and readout; the oracle score is an analytical reference, not an external teacher score or a proved optimal relevance score.

We measure weight mismatch by total variation and representation spread by the token diameter:

$$\mathrm{TV}(a, b) := \frac{1}{2}\sum_{i\in\mathcal{D}} |a_i - b_i|, \qquad \mathrm{Diam}(\mathcal{D}) := \max_{i,j\in\mathcal{D}} \|h_i - h_j\|_2.$$

Total variation is the amount of pooling weight that must be reassigned, while $\mathrm{Diam}(\mathcal{D})$ bounds the distance between the token vectors involved. We assume an L_g-Lipschitz readout, $|g(x) - g(x')| \leq L_g\|x - x'\|_2$. At inference, our affine classifier $g(x) = w^\top x + \beta$ satisfies this condition with $L_g = \|w\|_2$.

5.2 SCORE BOUNDS AND CONSISTENCY

Theorem 1 (Oracle-pooling score consistency)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. For any document-token pooling distribution a, we have*

$$|\hat{s}_a - s_{\text{oracle}}| \leq 2L_g \,\mathrm{Diam}(\mathcal{D})\,\mathrm{TV}(a, \tilde{y}).$$

For binary evidence labels, define the per-example document-token loss, using natural logarithms,

$$\mathcal{L}_{\mathcal{D}} = -\sum_i \big[y_i \log p_i + (1 - y_i)\log(1 - p_i)\big].$$

This sum differs from the sequence- and batch-averaged training loss; their relationship is given in Remark 1.

Corollary 1 (Probability-weighted score bound)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. Probability-weighted pooling $\tilde{p}$ satisfies*

$$|\hat{s}_{\tilde{p}} - s_{\text{oracle}}| \leq 2L_g \,\mathrm{Diam}(\mathcal{D})\mathcal{L}_{\mathcal{D}}.$$

Consequently, $\mathcal{L}_{\mathcal{D}} \to 0$ implies $|\hat{s}_{\tilde{p}} - s_{\text{oracle}}| \to 0$.

Corollary 2 (Mean-pooling score bound)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. Uniform pooling $\tilde{u}$ satisfies*

$$|\hat{s}_{\tilde{u}} - s_{\text{oracle}}| \leq 2L_g \,\mathrm{Diam}(\mathcal{D})\left(1 - \frac{r}{N}\right).$$

5.3 Interpretation and scope

Interpretation of the pooling-specific bounds. Both corollaries provide upper bounds, not exact score errors. Corollary 2 therefore does not imply a nonzero lower bound on the error of mean pooling, and Corollary 1 does not by itself establish that probability-weighted pooling has smaller actual error. For example, under an affine readout, if all token projections $w^\top h_i$ are equal, every document-token pooling distribution produces the same score, irrespective of its total variation from the oracle weights.

Consistency and the denominator clamp. Consider a fixed evidence-positive block, with its token representations and readout held fixed. By Theorem 1, $|\hat{s}_{\tilde{p}} - s_{\mathrm{oracle}}| \to 0$. This is the claimed conditional consistency property. To relate this statement to the implemented representation d_{prob}, recall from Section 4.1 that $d_{\mathrm{prob}} = \left(\frac{P}{\max\{P,\epsilon\}}\right)\bar{h}_{\tilde{p}}$. In the proof of Corollary 1, we have $|P - r| \le \mathcal{L}_{\mathcal{D}}$. Hence, $\mathcal{L}_{\mathcal{D}} \to 0$ implies $P \to r \ge 1$, so the denominator clamp is eventually inactive. When $P \ge \epsilon$, we have $g(d_{\mathrm{prob}}) = g(\bar{h}_{\tilde{p}})$ and this reduces to the normalized-pooling bound in Corollary 1.

6 Experiments

6.1 Experimental setup

We instantiate the unified-rerank setting of Section 4 on Provence (Chirkova et al., 2025), the published system that first trained a single encoder for sentence-level context pruning and document-level reranking together. Its backbone is DeBERTa-v3 (He et al., 2021), which fixes the encoder for every variant we train: a hard limit of 512 input tokens per forward pass, leaving roughly 490 content tokens per block once the query and the special tokens are accounted for. That limit is what makes the multi-block aggregation question of Section 4.3 concrete. It gives us a trained starting point, a released checkpoint to compare against, and a published set of baseline operating points measured under the same retrieval and generation pipeline. Its ranking head uses CLS pooling with a document-relevance target, which is one of the four points in the design space of Section 4; the other three are ours, trained identically and differing only in the pooling operator or the regression target.

We evaluate all four ranking-head variants in *unified-rerank* mode. The four are CLS pooling, probability-weighted pooling, mean pooling and the span-or-penalty target, all trained on MS MARCO-only data (Section 4.4). In unified-rerank mode, SPLADE-v3 retrieves 50 candidates and the ranking head itself selects and reorders the documents that reach a Llama-2-7B-chat generator. Downstream answer accuracy is judged by a SOLAR-10.7B LLM judge against reference answers. The span-or-penalty variant uses a penalty coefficient of $\kappa = 0.5$, defined in Section 4.2. A full-context baseline with no pruning is included on every dataset, and all results use the default FIRST ranking-score aggregation (Section 4.3).

A context compressor is not a single operating point but a family of them, exposed through the knobs a practitioner can turn. For our variants there are two: the sentence-pruning threshold, which we sweep over $\{0.1, 0.5\}$, and the number of documents the ranking head retains, which we sweep over $\{1, 2, 3, 4, 5\}$. This gives each variant ten operating points per dataset, or 240 variant–condition evaluations across the four variants and six datasets. We are not interested in the retention count as a finding in its own right, since a smaller prompt at lower accuracy is the expected outcome of retaining fewer documents. It matters here only because it traces out the frontier each variant can reach, and comparing frontiers rather than single points is what Section 6.3 does. Tables reporting a single operating point use the retention count of five that the original Provence evaluation uses.

We report context cost in two ways. The *compression rate* is the percentage reduction in context length relative to the unpruned document set. It is the axis the original Provence evaluation reports, so it is the one on which published baselines can be placed alongside our variants. It is measured before documents are dropped, however, so it registers only sentence-level pruning. Where we need the true cost of a configuration we instead use the *mean prompt size*, the average number of Llama-2 tokens in the prompt actually sent to the generator, tokenized from each run's logged instruction.

Comparison	Mean (pts)	95% CI	p	Datasets won
Prob.-weighted − Span-or-Penalty	$+3.71$	$[+1.51, +5.91]$	$\mathbf{0.004}$	6/6
CLS − Span-or-Penalty	$+3.21$	$[+1.33, +5.08]$	$\mathbf{0.003}$	6/6
Mean − Span-or-Penalty	$+3.45$	$[+1.10, +5.81]$	$\mathbf{0.009}$	6/6
Prob.-weighted − Mean	$+0.26$	$[+0.03, +0.49]$	$\mathbf{0.029}$	5/6
Prob.-weighted − CLS	$+0.50$	$[-0.27, +1.28]$	0.174	4/6
Mean − CLS	$+0.25$	$[-0.49, +0.98]$	0.469	3/6

Table 2: Pairwise comparisons of the ranking heads on judge accuracy, over all 60 (dataset, threshold, retention) conditions. A positive mean favours the first head. p is two-sided and corrects for the measured between-dataset correlation; 95% CI is the matching interval. The three span-or-penalty comparisons are large and robust. Probability-weighted pooling beats mean pooling by a small but consistent margin. It does not separate from CLS pooling, whose relative standing varies strongly by dataset. Appendix F gives the method.

Dataset	CLS	Prob.-weighted	Mean	Span-or-penalty
NQ	71.4 / 85.1	**72.1** / 81.9	71.9 / 83.1	71.0 / 83.6
HotpotQA	55.1 / 93.1	**56.4** / 90.7	56.0 / 91.6	53.6 / 93.4
PopQA	**57.5** / 83.8	56.9 / 81.7	57.3 / 82.5	52.4 / 83.4
TyDiQA	72.5 / 83.1	72.7 / 79.6	**73.2** / 80.7	72.2 / 81.8
PubMed	81.7 / 84.9	**82.1** / 83.7	81.0 / 83.9	81.4 / 84.6
SyllabusQA	55.3 / 94.4	**56.9** / 94.1	55.8 / 93.6	50.6 / 94.6
Average	65.6 / 87.4	**66.2** / 85.3	65.9 / 85.9	63.5 / 86.9

Table 3: Accuracy / compression (percent) at retention count $k = 5$ and pruning threshold 0.5, using the SOLAR-10.7B judge. Bold marks the best accuracy among our four heads within each row.

We evaluate six datasets with 50 SPLADE-v3 candidates per query; their compact corpus summary appears beside the main accuracy–compression figure.

We reproduce the published pipeline with the specified Wikipedia and SyllabusQA corpora and report the remaining PubMed corpus mismatch as a caveat. The four ranking-head variants are compared with the published Provence checkpoint and the published context-pruning baselines; detailed corpus, baseline, and reproduction information appears in Appendix C and Appendix D.

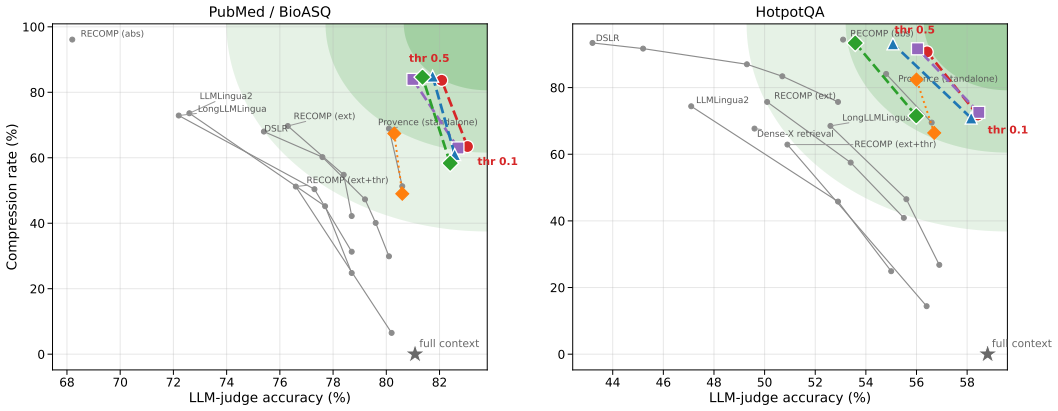
6.2 Main results: which ranking head is better?

We begin with the comparison the paper is for: given the same encoder, the same training data and the same pipeline, does the choice of pooling operator or regression target change how well the unified model ranks? Each variant is evaluated at every combination of dataset, pruning threshold and retention count, and the comparison is paired within each of those conditions, so differences in dataset difficulty cancel.

Table 2 tests every pair of heads across the full grid of 6 datasets $\times$ 2 thresholds $\times$ 5 retention counts, giving 60 paired observations per comparison. The cells of that grid are not independent, since ten of them share a dataset, so we estimate the between-dataset correlation rather than assume it and inflate the standard error accordingly. Appendix F states the procedure; Appendix G reports the correlation, the effective sample size and a sensitivity analysis for each comparison.

Two results survive this treatment. The span-or-penalty target is worse than all three document-target heads by 3.2 to 3.7 accuracy points on every dataset, and the margin holds even if the grid is collapsed to one observation per dataset. Probability-weighted pooling beats mean pooling by $+0.26$ points, winning five of the six datasets.

Two do not. Probability-weighted pooling does not separate from CLS pooling despite a larger mean gap of $+0.50$ points, because that gap swings from -0.64 on PopQA to $+1.93$ on SyllabusQA; consistency rather than magnitude is what the test rewards. Mean pooling and CLS pooling are indistinguishable.



(a) PubMed/BioASQ (b) HotpotQA

Figure 1: Accuracy–compression trade-offs at $k = 5$ for the two pruning thresholds. Higher values on both axes are better. Published baselines are from Chirkova et al. (2025); corpus differences qualify the PubMed comparison (Section 6.1). The all-dataset figure and teacher-score analysis are in the appendix.

The span-or-penalty target is consistently weaker than the three document-target heads, whereas probability-weighted pooling gives a smaller but statistically significant improvement over mean pooling. Probability-weighted pooling does not significantly separate from CLS pooling. These conclusions are limited to one checkpoint per variant and to the LLM-judged accuracy metric; the dependence correction and metric analyses are reported in Appendix F and Appendix J.

6.3 Comparison against published context-pruning methods

The unified variants are competitive with the published context-pruning methods. At the retention count used for the published comparison, the document-target heads generally match or exceed the published checkpoint while retaining substantially less context; the complete per-dataset comparison and its corpus caveats are in Appendix C and Appendix D. HotpotQA is the main raw-accuracy exception, but its small full-context advantage disappears when prompt cost is included.

Hypervolume analysis. Across the full accuracy–prompt-size frontier, probability-weighted pooling has the highest mean hypervolume and ranks first on four of the six datasets, but its margin over the other document-target heads is not statistically significant. The full table, normalization details, sensitivity analysis, Pareto fronts, and teacher-score analysis are in Appendix H and Appendix I.

7 Conclusion

We studied how a unified context compressor and document reranker should form and supervise document representations. Probability-weighted pooling reuses the compression head's token-level evidence probabilities, and our analysis gives a conditional consistency bound relating those weights to an oracle evidence-weighted representation. Across six benchmarks, document-level ranking supervision has the clearest effect: it consistently outperforms span-or-penalty supervision. Probability-weighted pooling provides a smaller, statistically supported improvement over mean pooling, with a more visible benefit on PubMed, while remaining statistically comparable to CLS pooling. These results support using shared evidence predictions for ranking, but the modest pooling differences and single-checkpoint design leave repeated-seed and longer-document evaluations as important directions for future work.

AI USE STATEMENT

We used multiple AI tools to support manuscript preparation, retrieval of relevant literature and technical sources, and development of the theoretical analysis. Writing assistance included drafting and revising portions of the manuscript, including the abstract and theory section, and improving the organization, clarity, and precision of the exposition.

For the theoretical analysis, AI assistance included refining mathematical claims, identifying missing assumptions and counterexamples, developing proof arguments, and drafting proofs. We also used AI assistance to inspect the training and inference code, compare the pooling and loss implementations with the mathematical definitions, and develop numerical checks of the stated inequalities. These checks support the analysis but do not replace the proofs or extend their stated assumptions.

We reused the training examples and supplied evidence annotations from the prior Provence work (Chirkova et al., 2025); we did not use generative AI to create additional training examples or evidence annotations for this study. The teacher cross-encoder provides the ranking supervision described in Section 4.2. The experimental pipeline also uses language models for answer generation and accuracy evaluation, as described in Section 6.1.

We take responsibility for the final content of this work, including the text, mathematical claims, citations, code, and interpretation of results developed with AI assistance. AI assistance does not provide independent validation of the theoretical or empirical claims.

REPRODUCIBILITY STATEMENT

The model variants, training targets, and evaluation settings are described in Section 4 and Section 6.1. The appendix provides complete proofs (Appendix A), per-dataset results (Appendix C), and the statistical procedure (Appendix F). The anonymized supplementary source package contains the relevant training and inference code, configuration files, and analysis scripts. It does not include model checkpoints, downloaded datasets, or experiment logs; reproducing the numerical results requires obtaining those resources separately.

REFERENCES

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *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. 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1352. URL https://aclanthology.org/D19-1352/.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 38082–38106, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5e956fef0946dc1e39760f94b78045fe-Paper-Conference.pdf.

Eunsol Choi, Jennimaria Palomaki, Matthew Lamm, Tom Kwiatkowski, Dipanjan Das, and Michael Collins. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics*, 9:447–461, 2021. doi: 10.1162/tacl_a_00377. URL https://aclanthology.org/2021.tacl-1.27/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019. URL https://arxiv.org/abs/1810.04805.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2127–2136. PMLR, 2018. URL https://proceedings.mlr.press/v80/ilse18a.html.

Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. Splade-v3: New baselines for splade, 2024. URL https://arxiv.org/abs/2403.06789.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. PARADE: Passage representation aggregation for document reranking. *ACM Transactions on Information Systems*, 42(2), 2024. doi: 10.1145/3600088. URL https://doi.org/10.1145/3600088.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Dimitris Pappas and Ion Androutsopoulos. A neural model for joint document and snippet ranking in question answering for large document collections. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3896–3907, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.301. URL https://aclanthology.org/2021.acl-long.301/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019. URL https://arxiv.org/abs/1908.10084.

A Proofs and additional analysis of oracle-pooling consistency

We provide the detailed definitions, supporting lemmas, and proofs for Section 5. We also relate the bounds to the implemented loss and denominator clamp, derive the affine-readout and pairwise-ordering refinements, and discuss the scope of the reference score.

A.1 Preliminaries and definitions

We analyze one query–document-block pair at a time. As described in Section 4.3, a block is the portion of a document processed in a single encoder pass; if the document fits within the encoder's input window, the block is the entire document. All quantities below are associated with this fixed pair. We condition on the contextual token representations produced by the encoder and hold the ranking readout fixed, thereby isolating the effect of changing only the pooling weights.

Let L be the length of the input sequence. For each $i \in [L]$, let h_i be the backbone's final-layer hidden state and $m_i \in \{0, 1\}$ be the indicator that the i-th token is a document token. We define $\mathcal{D}$ to be the set of document-token indices and N to be its size, i.e.

$$\mathcal{D} := \{i : m_i = 1\}, \qquad N := |\mathcal{D}| \geq 1.$$

Let $\Delta(\mathcal{D})$ be the set of all probability distribution on $\mathcal{D}$, i.e.

$$\Delta(\mathcal{D}) := \{a \in \mathbb{R}^{\mathcal{D}} : \sum_{i \in \mathcal{D}} a_i = 1 \text{ and } a_i \geq 0 \text{ for all } i \in \mathcal{D}\}.$$

We call a probability distribution $a \in \Delta(\mathcal{D})$ *a pooling distribution*. For any pooling distribution $a \in \Delta(\mathcal{D})$, let $\bar{h}_a$ be the weighted average of the corresponding hidden states, i.e.

$$\bar{h}_a := \sum_{i \in \mathcal{D}} a_i h_i.$$

Unless stated otherwise, all subsequent sums, maxima, and minima over token positions are taken over $\mathcal{D}$.

For any $i \in \mathcal{D}$, let $y_i \in \{0, 1\}$ be the supplied binary evidence label. We define S to be the set of labeled evidence-token indices and r to be its size, i.e.

$$S := \{i \in \mathcal{D} : y_i = 1\}, \qquad r := |S|.$$

We assume $r \geq 1$.

For any $i \in \mathcal{D}$, let $c_i \in \mathbb{R}$ be the compression logit and $p_i = \sigma(c_i) \in (0, 1)$ the corresponding predicted evidence probability, where σ is the sigmoid function. We define P to be the total predicted evidence mass within the document-tokens, i.e.

$$P := \sum_{i \in \mathcal{D}} p_i$$

Also, we define $\tilde{p}$, $\tilde{u}$ and $\tilde{y}$ to be the probability-weighted, uniform and oracle pooling distribution respectively, i.e.

$$\tilde{p}_i := \frac{p_i}{P}, \qquad \tilde{u}_i := \frac{1}{N}, \qquad \tilde{y}_i := \frac{y_i}{r} \qquad \text{for all } i \in \mathcal{D}.$$

For any $i \in \mathcal{D}$, let ℓ_i be the binary cross-entropy with respect to y_i, p_i, i.e.

$$\ell_i := -y_i \log p_i - (1 - y_i) \log(1 - p_i) \qquad \text{where } \log \text{ denotes the natural logarithm.}$$

Also, let $\mathcal{L}_{\mathcal{D}}$ be the per-example binary cross-entropy sum over document tokens, i.e.

$$\mathcal{L}_{\mathcal{D}} := \sum_{i \in \mathcal{D}} \ell_i$$

For any two pooling distributions $a, b \in \Delta(\mathcal{D})$, we define $\mathrm{TV}(a, b)$ to be the total variation between a and b as usual.

$$\mathrm{TV}(a, b) := \frac{1}{2} \sum_{i \in \mathcal{D}} |a_i - b_i|.$$

We also define $\mathrm{Diam}(\mathcal{D})$ to be the diameter of the hidden states among the document-tokens, i.e.

$$\mathrm{Diam}(\mathcal{D}) := \max_{i,j \in \mathcal{D}} \|h_i - h_j\|_2.$$

Let g be the model's deterministic ranking readout at inference time. For any pooling distribution $a \in \Delta(\mathcal{D})$, we define $\hat{s}_a$ to be the model score obtained using the pooling distribution a, i.e.

$$\hat{s}_a := g(\bar{h}_a).$$

In particular, if the pooling distribution a is $\tilde{y}$, we write

$$s_{\mathrm{oracle}} := \hat{s}_{\tilde{y}} = g(\bar{h}_{\tilde{y}}).$$

A function g is Lipschitz if there exists a positive number L_g such that $|g(x') - g(x)| \leq L \cdot \|x - x'\|_2$ for any x, x'. We call g is L_g-Lipschitz to specify the positive Lipschitz constant L_g. In particular, our affine classifier has the format of $g(x) = w^\top x + \beta$ and it is an L_g-Lipschitz function with $L_g = \|w\|_2$. We call such a function *an affine readout.*

Remark 1. Relation to the implemented training loss.

- The theoretical quantity $B_{\mathcal{D}}$ is a per-example sum of binary cross-entropy over document tokens and does not equal the scalar compression loss used during training. For the probability-weighted and mean-pooling variants, the implementation applies unweighted binary cross-entropy to all L positions in each padded input sequence, including query,

special, and padding positions, whose labels are zero. Extending ℓ_i to all input positions, the per-example sequence loss is

$$\bar{\mathcal{L}}_{\text{seq}} = \frac{1}{L}\sum_{i=1}^{L} \ell_i.$$

Since each loss term is nonnegative and $\mathcal{D} \subseteq \{1, \ldots, L\}$,

$$\mathcal{L}_{\mathcal{D}} \leq \sum_{i=1}^{L} \ell_i = L\bar{\mathcal{L}}_{\text{seq}}.$$

Thus, the theoretical bound can be related to the implemented loss through $\mathcal{L}_{\mathcal{D}} \leq L\bar{\mathcal{L}}_{\text{seq}}$. This relation holds for a single example and its corresponding logits. The reported training loss, however, is further averaged across examples and devices and therefore cannot be identified with any individual example's $B_{\mathcal{D}}$. Moreover, a stochastic training-time loss does not directly characterize the deterministic inference-time predictions analyzed in this section.

Remark 2 (Scope and implementation correspondence)**.** Three qualifications delimit the analysis:

- **Well-definedness and scope.** In exact arithmetic, finite compression logits imply $p_i \in (0, 1)$; together with $N \geq 1$, this ensures $P > 0$ and makes $\tilde{p}$ well-defined. In finite-precision computation, sigmoid underflow can instead produce $P = 0$. The denominator clamp still makes d_{prob} well defined in that case, but $\tilde{p}$ is undefined, so the normalized-pooling analysis below applies only when the computed value of P is positive. The oracle weights additionally require $r \geq 1$, so the analysis is restricted to *evidence-positive* blocks containing at least one labeled evidence token. Here, *oracle* refers solely to the supplied evidence mask and does not imply that the mask is error-free or that the resulting representation or score is optimal for ranking.
- **Implementation correspondence.** Section 4.1 defines the implemented probability-weighted representation as

$$d_{\text{prob}} = \frac{\sum_{i\in\mathcal{D}} p_i h_i}{\max\{P, \epsilon\}} = \frac{P}{\max\{P, \epsilon\}}\bar{h}_{\tilde{p}}, \qquad \epsilon = 10^{-9}.$$

  Our analysis focuses on the normalized representation $\bar{h}_{\tilde{p}}$. When $P \geq \epsilon$, the implementation coincides with this representation: $d_{\text{prob}} = \bar{h}_{\tilde{p}}$. Indeed, the region we focus on is $P \geq \epsilon$ as explained in Section 5.3. The denominator clamp is for implementation stability.
- **CLS pooling.** The CLS-pooling baseline described in Section 4.1 represents the input using the contextual state of the leading `[CLS]` (*classification*) token, a special token prepended to the query–document sequence for sequence-level prediction. Because this token is not a document token, CLS pooling operates outside $\{h_i : i \in \mathcal{D}\}$. Its learned dense layer and GELU activation therefore cannot, in general, be expressed as pooling with a distribution in $\Delta(\mathcal{D})$. We retain CLS pooling only as an empirical baseline.

A.2 Proof of Theorem 1

In this subsection, we will give the detailed proof for our main theorem (Theorem 1). Theorem 1 considers an L_g-Lipschitz readout, meaning that $|g(x) - g(x')| \leq L_g\|x - x'\|_2$ for all pooled representations x and x' under consideration. The implementation-level clamp bound below additionally requires this inequality between d and $\bar{h}_{\tilde{p}}$. Section 4.1 writes the implemented ranking score as $s = W_r\,\text{dropout}(d) + b_r$. At inference, dropout is disabled. Writing the scalar-output row vector W_r as $w^\top$ and its bias b_r as β, the corresponding theoretical readout is $g(x) = w^\top x + \beta$. We preview that, for this specific form of g, we provide a tighter result in Theorem 2.

Lemma 1. *Given fixed token representations and an L_g-Lipschitz ranking readout. For any document-token pooling distributions a and b, we have*

$$|\hat{s}_a - \hat{s}_b| \leq 2L_g\,\text{Diam}(\mathcal{D})\,\text{TV}(a, b).$$

Proof. Recall that $\hat{s}_a = g(\bar{h}_a)$ and $\hat{s}_b = g(\bar{h}_b)$. We have

$$|\hat{s}_a - \hat{s}_b| = |g(\bar{h}_a) - g(\bar{h}_b)| \leq L_g \|\bar{h}_a - \bar{h}_b\|_2 \qquad \text{by the Lipschitz property of } g.$$

Namely, we need to prove

$$\|\bar{h}_a - \bar{h}_b\|_2 \leq 2\,\mathrm{Diam}(\mathcal{D})\,\mathrm{TV}(a, b).$$

Let i^*, j^* be the indices in $\mathcal{D}$ such that $\mathrm{Diam}(\mathcal{D}) = \|h_{i^*} - h_{j^*}\|_2$, i.e. $i^*, j^* = \arg\max_{i,j} \|h_i - h_j\|_2$. We have

$$\bar{h}_a - \bar{h}_b = \sum_{i \in \mathcal{D}} a_i h_i - \sum_{i \in \mathcal{D}} b_i h_i = \sum_{i \in \mathcal{D}} (a_i - b_i)(h_i - h_{j^*}).$$

The last equality is from the fact that both a and b are probability distributions and hence $\sum_{i \in \mathcal{D}} (a_i - b_i) = 0$. By triangle inequality, we have

$$\|\bar{h}_a - \bar{h}_b\|_2 \leq \sum_{i \in \mathcal{D}} |a_i - b_i| \|h_i - h_{j^*}\|_2 \leq \left(\sum_{i \in \mathcal{D}} |a_i - b_i| \right) \mathrm{Diam}(\mathcal{D}) = 2\,\mathrm{Diam}(\mathcal{D})\,\mathrm{TV}(a, b).$$

The penultimate inequality is due to $\|h_i - h_{j^*}\|_2 \leq \|h_{i^*} - h_{j^*}\|_2 = \mathrm{Diam}(\mathcal{D})$ for any $i \in \mathcal{D}$ and the last equality is from the definition of $\mathrm{TV}(a, b)$ that $2\,\mathrm{TV}(a, b) = \sum_{i \in \mathcal{D}} |a_i - b_i|$. Hence, the result follows.

□

Theorem 1 (Oracle-pooling score consistency)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. For any document-token pooling distribution a, we have*

$$|\hat{s}_a - s_{\mathrm{oracle}}| \leq 2 L_g\,\mathrm{Diam}(\mathcal{D})\,\mathrm{TV}(a, \tilde{y}).$$

Proof. We set $b = \tilde{y}$ in Lemma 1. The result immediately follows.

□

A.3 Proof of Corollary 1 and Corollary 2

In this subsection, we will provide the detailed proof of Corollary 1 and Corollary 2.

Corollary 1 (Probability-weighted score bound)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. Probability-weighted pooling $\tilde{p}$ satisfies*

$$|\hat{s}_{\tilde{p}} - s_{\mathrm{oracle}}| \leq 2 L_g\,\mathrm{Diam}(\mathcal{D}) \mathcal{L}_{\mathcal{D}}.$$

Consequently, $\mathcal{L}_{\mathcal{D}} \to 0$ implies $|\hat{s}_{\tilde{p}} - s_{\mathrm{oracle}}| \to 0$.

Proof. We set $a = \tilde{p}$ in Theorem 1. Hence, we need to show

$$\mathrm{TV}(\tilde{p}, \tilde{y}) \leq \mathcal{L}_{\mathcal{D}}.$$

Recall that $\tilde{p} = \frac{p}{P}$ and $\tilde{y} = \frac{y}{r}$. We have

$$2\,\mathrm{TV}(\tilde{p}, \tilde{y}) = \left\| \frac{p}{P} - \frac{y}{r} \right\|_1 \leq \left\| \frac{p}{P} - \frac{p}{r} \right\|_1 + \left\| \frac{p}{r} - \frac{y}{r} \right\|_1 \qquad \text{by triangle inequality.}$$

The first term $\| \frac{p}{P} - \frac{p}{r} \|_1 = |\frac{1}{P} - \frac{1}{r}| \|p\|_1 = \frac{|P - r|}{r}$ by the definition of $P = \|p\|_1$. Moreover, we further bound the term $|P - r|$ by

$$|P - r| = |\|p\|_1 - \|y\|_1| \leq \|p - y\|_1 \qquad \text{by triangle inequality.}$$

Hence, we have

$$\mathrm{TV}(\tilde{p}, \tilde{y}) \leq \frac{1}{2} \left(\frac{\|p - y\|_1}{r} + \frac{\|p - y\|_1}{r} \right) \leq \|p - y\|_1 \qquad \text{recall that we assume } r \geq 1.$$

Now, we need to prove $\|p - y\|_1 \leq \mathcal{L}_{\mathcal{D}}$. Recall that

$$\mathcal{L}_{\mathcal{D}} = \sum_{i \in \mathcal{D}} \ell_i = \sum_{i \in \mathcal{D}} (-y_i \log p_i - (1 - y_i) \log(1 - p_i)).$$

It is easy to check that, by considering $y_i = 0$ or $y_i = 1$, it can be rewritten as

$$\mathcal{L}_{\mathcal{D}} = -\sum_{i \in \mathcal{D}} \log(1 - |p_i - y_i|).$$

By the inequality $\log(x) \leq x - 1$ for all $x > 0$, we have

$$\mathcal{L}_{\mathcal{D}} = -\sum_{i \in \mathcal{D}} \log(1 - |p_i - y_i|) \geq -\sum_{i \in \mathcal{D}} ((1 - |p_i - y_i|) - 1) = \|p - y\|_1.$$

Hence, the result follows.

□

Corollary 2 (Mean-pooling score bound)**.** *Given fixed token representations and an L_g-Lipschitz ranking readout. Uniform pooling $\tilde{u}$ satisfies*

$$|\hat{s}_{\tilde{u}} - s_{\text{oracle}}| \leq 2L_g \operatorname{Diam}(\mathcal{D}) \left(1 - \frac{r}{N}\right).$$

Proof. We set $a = \tilde{u}$ in Theorem 1. Hence, we need to show

$$\operatorname{TV}(\tilde{u}, \tilde{y}) = 1 - \frac{r}{N}.$$

Recall that $|\mathcal{D}| = N$, $S = \{i : \mathcal{D} : y_i = 1\} \subseteq \mathcal{D}$ and $r = |S|$. Hence, $r \leq N$. Also,

$$\tilde{u}_i = \frac{1}{N} \quad \text{for } i \in \mathcal{D} \qquad \text{and} \qquad \tilde{y}_i = \begin{cases} 0 & \text{for } i \notin S \\ \frac{1}{r} & \text{for } i \in S. \end{cases}$$

We have

$$2 \operatorname{TV}(\tilde{u}, \tilde{y}) = \sum_{i \in S} |\frac{1}{N} - \frac{1}{r}| + \sum_{i \notin S} |\frac{1}{N} - 0| = r\left(\frac{1}{r} - \frac{1}{N}\right) + \frac{N - r}{N} = 2\left(1 - \frac{r}{N}\right).$$

Hence, the result follows.

□

A.4 Affine readout

The preceding bounds use the full diameter $\operatorname{Diam}(\mathcal{D})$ of the token representations. For the affine readout used by our model, only variation along the classifier direction w can affect the score, which permits a tighter coefficient.

Theorem 2 (Affine readout refinement)**.** *Given fixed token representations $\{h_i \mid i \in \mathcal{D}\}$ and an affine readout $g = w^\top x + \beta$. Define the range of token-level classifier projections by $\Omega_w = \max_{i \in \mathcal{D}} w^\top h_i - \min_{i \in \mathcal{D}} w^\top h_i$. For any document-token pooling distributions a and b, we have*

$$|\hat{s}_a - \hat{s}_b| \leq \Omega_w \operatorname{TV}(a, b).$$

Proof. Recall that $\hat{s}_a = g(\bar{h}_a) = w^\top \bar{h}_a + \beta$ and $\hat{s}_b = g(\bar{h}_b) = w^\top \bar{h}_b + \beta$. We have

$$|\hat{s}_a - \hat{s}_b| = |w^\top (\bar{h}_a - \bar{h}_b)| = |\sum_{i \in \mathcal{D}} (a_i - b_i) w^\top h_i|.$$

Let c be $(\max_{i \in \mathcal{D}} w^\top h_i + \min_{i \in \mathcal{D}} w^\top h_i)/2$. We have

$$\sum_{i \in \mathcal{D}} (a_i - b_i) w^\top h_i = \sum_{i \in \mathcal{D}} (a_i - b_i)(w^\top h_i - c)$$

since both a and b are probability distributions and hence $\sum_{i\in\mathcal{D}}(a_i - b_i) = 0$. By triangle inequality, we have

$$|\hat{s}_a - \hat{s}_b| = |\sum_{i\in\mathcal{D}}(a_i - b_i)(w^\top h_i - c)| \le \sum_{i\in\mathcal{D}}|a_i - b_i||w^\top h_i - c|$$

It is easy to check that, for any $i \in \mathcal{D}$, $|w^\top h_i - c| \le \Omega_w/2$. Hence, we have

$$|\hat{s}_a - \hat{s}_b| \le \sum_{i\in\mathcal{D}}|a_i - b_i||w^\top h_i - c| \le \sum_{i\in\mathcal{D}}|a_i - b_i|(\Omega_w/2) = \Omega_w \,\mathrm{TV}(a, b).$$

□

This refinement depends only on variation in the component of the token representations aligned with w. Any displacement orthogonal to w leaves the affine ranking score unchanged. Furthermore, we observe that

$$\Omega_w = \max_{i,j\in\mathcal{D}} w^\top(h_i - h_j) \le \max_{i,j\in\mathcal{D}} \|w\|_2\|h_i - h_j\|_2 = \|w\|_2 \,\mathrm{Diam}(\mathcal{D}) < 2\|w\|_2 \,\mathrm{Diam}(\mathcal{D})$$

which means that the bound in Theorem 2 is no worse than the bound in Theorem 1.

A.5 Interpretation and limitations

Evidence weighting and score discrepancy. The compression probabilities explicitly determine the ranking representation, and token BCE bounds their mismatch from the oracle weights. This is a consistency statement rather than a monotonicity result: a reduction in BCE need not reduce actual weight or score discrepancy at every step. For mean pooling, only the weight mismatch $1 - r/N$ is fixed; the hidden-state geometry and ranking readout can still change during training. A larger evidence fraction reduces this weight mismatch, but document length alone does not determine the evidence fraction or predict a downstream performance gap. When $r = N$, the uniform weights equal the oracle weights; the probability weights equal them only when the normalized predictions are uniform.

Training and supervision. The analysis compares operators on the same contextual states and readout, rather than comparing separately trained models. Extending the convergence statement to a sequence of jointly trained models requires control of the changing coefficient, for example a uniform bound on $L_g \,\mathrm{Diam}(\mathcal{D})$. For documents of varying length, vanishing mean document-token BCE alone is insufficient unless its product with N/r also vanishes, or another condition controls the stated bound. Furthermore, training loss does not by itself give a generalization guarantee. Probability weighting introduces evidence-aware structure, but the theorem does not establish that span-level ranking targets are redundant: changing the target can change the encoder, readout, and optimization outcome.

External teacher scores. The internal oracle score is not identified with either the teacher's document score or its score on an extracted span. For the DOC target s_T, the triangle inequality gives

$$|\hat{s}_a - s_T| \le 2L_g \,\mathrm{Diam}(\mathcal{D})\,\mathrm{TV}(a, \tilde{y}) + \varepsilon_{\mathrm{probe}}, \qquad \varepsilon_{\mathrm{probe}} = |s_{\mathrm{oracle}} - s_T|.$$

This decomposition makes the additional teacher discrepancy explicit. The compression bound does not control $\varepsilon_{\mathrm{probe}}$ or establish that it is small.

Scope of the reference representation. The requirement $r \ge 1$ excludes blocks with no supporting span, so the pairwise result does not cover positive-versus-negative ranking. For $r = 0$, all probabilities can approach zero while their normalized proportions remain arbitrary. Oracle pooling also retains the contextualization of the original block; it does not re-encode the extracted evidence. Consequently, the analysis does not guarantee correctness of the supplied mask, behavior under multi-block aggregation, or retrieval and downstream generation accuracy. CLS pooling remains outside its scope.

B GOLD DOCUMENTS REQUIRED PER QUERY

How much the ranking head can matter depends on how many documents a query actually needs. A query answerable from a single passage tolerates an imprecise head, because retaining five of fifty candidates will often include that passage anyway. A query needing several passages does not, since each one has to survive the retention cut. Table 4 counts distinct gold provenance entries per query from the oracle runs.

Passages are the units the retriever returns and the ranking head selects over. Documents are the underlying sources, and the two differ for the KILT corpora, where several passages can come from one Wikipedia page. For BioASQ one abstract is one passage, so the two coincide.

Dataset	Queries	Passages/query	≥ 2	> 5	Documents/query
NQ	2,675	1.22	17.1%	0.2%	1.00
HotpotQA	5,600	2.01	100.0%	0.0%	2.00
PubMed / BioASQ	4,719	9.03	79.3%	52.7%	9.03
PopQA, SyllabusQA, TyDiQA	no gold provenance available				

Table 4: Gold provenance entries per query. NQ is effectively single-document, every HotpotQA query needs exactly two, and for over half of PubMed's queries the relevant set is larger than the retention budget of five. Three of our six datasets ship no gold provenance in the evaluation framework we use, so they cannot be counted.

These counts are produced by a script that validates the shape of every document identifier against the shape the dataset is declared to use, and refuses to report a dataset whose identifiers do not match. That check is not redundant. The evaluation framework ships two files whose names differ only in case, and the lower-case one is mislabelled: it contains Wikipedia entity URLs rather than PubMed identifiers, and reading it yields exactly 1.00 gold documents per query for PubMed instead of 9.03. The same check also caught our own initial conflation of KILT passage identifiers with page identifiers, which is the distinction the two right-hand columns now separate.

C PER-DATASET ACCURACY AND COMPRESSION

Table 5 gives accuracy and compression for every dataset and ranking head at the retention count of five, alongside the published Provence checkpoint measured under the same pipeline. Section 6.3 summarises it; the per-dataset detail is here.

D COMPARISON AGAINST PUBLISHED CONTEXT-PRUNING METHODS

Section 6.3 compares the four ranking heads against each other on the prompt-size axis, which is the axis that reflects what the generator is actually charged for. The published baselines cannot be placed on that axis, because the numbers reported for them give accuracy and compression rate only, with no prompt-token counts. This appendix therefore repeats the hypervolume comparison on the compression axis, where all methods can be placed together.

Setup. Each operating point is normalized as

$$n_x = \frac{\text{accuracy}}{100}, \qquad n_y = \frac{\text{compression}}{100},$$

with the same fixed reference point $r = (0, 0)$ and the same sweep formula as Section 6.3. Here r means zero accuracy with no compression achieved. Every method in Table 6 retains five documents, ours and published alike, so the comparison is like for like. Baseline figures are the ones reported by Chirkova et al. (2025) rather than reproductions of our own. The full-context baseline achieves no compression by definition, so it sits on the reference axis with hypervolume zero and is reported as an accuracy reference in Table 5 instead of as a row here.

Dataset	Ranking head	Ours		Provence (published)	
		thr. 0.1	thr. 0.5	thr. 0.1	thr. 0.5
NQ (full ctx. 74.7)	CLS Pooling	**76.5** / 59.8	71.4 / 85.1	72.4 / 62.2	72.6 / 76.0
	Prob.-weighted Pooling	75.4 / 60.0	**72.1** / 81.9		
	Mean Pooling	74.7 / 61.3	71.9 / 83.1		
	Span-or-Penalty	71.0 / 57.0	71.0 / 83.6		
HotpotQA (58.8)	CLS Pooling	58.2 / 70.9	55.1 / 93.1	56.7 / 66.4	56.0 / 82.4
	Prob.-weighted Pooling	58.4 / 71.8	**56.4** / 90.7		
	Mean Pooling	**58.5** / 72.6	56.0 / 91.6		
	Span-or-Penalty	56.0 / 71.5	53.6 / 93.4		
PopQA (59.7)	CLS Pooling	**61.0** / 60.6	**57.5** / 83.8	59.3 / 68.6	59.5 / 75.8
	Prob.-weighted Pooling	60.0 / 62.0	56.9 / 81.7		
	Mean Pooling	60.4 / 63.2	57.3 / 82.5		
	Span-or-Penalty	52.9 / 58.1	52.4 / 83.4		
TyDiQA (76.1)	CLS Pooling	74.9 / 58.2	72.5 / 83.1	70.5 / 63.0	73.6 / 76.2
	Prob.-weighted Pooling	74.7 / 58.0	72.7 / 79.6		
	Mean Pooling	**75.1** / 59.0	**73.2** / 80.7		
	Span-or-Penalty	69.2 / 56.7	72.2 / 81.8		
PubMed (81.1)	CLS Pooling	82.6 / 61.0	81.7 / 84.9	80.6 / 49.0	80.3 / 67.4
	Prob.-weighted Pooling	**83.1** / 63.4	**82.1** / 83.7		
	Mean Pooling	82.7 / 63.0	81.0 / 83.9		
	Span-or-Penalty	82.4 / 58.3	81.4 / 84.6		
SyllabusQA (59.8)	CLS Pooling	61.8 / 62.8	55.3 / 94.4	49.8 / 60.6	51.0 / 76.5
	Prob.-weighted Pooling	61.1 / 72.6	**56.9** / 94.1		
	Mean Pooling	**62.5** / 71.4	55.8 / 93.6		
	Span-or-Penalty	56.1 / 72.4	50.6 / 94.6		
Average (68.4)	CLS Pooling	**69.2** / 62.2	65.6 / 87.4	64.9 / 61.6	65.5 / 75.7
	Prob.-weighted Pooling	68.8 / 64.6	**66.2** / 85.3		
	Mean Pooling	69.0 / 65.1	65.9 / 85.9		
	Span-or-Penalty	64.6 / 62.4	63.5 / 86.9		

Table 5: Accuracy and compression (%) at the retention count of five, SOLAR-10.7B judge, reported as accuracy / compression. **Bold** marks the best of our four variants per dataset and threshold. The right-hand block repeats the published Provence unified-rerank result for the same dataset, which is identical across our four rows and is therefore given once per dataset. The final block macro-averages each row over the six datasets, weighting every dataset equally.

Because compression is measured before any document is dropped, it is unchanged by the retention count. We report the single retention count of five used by the published evaluation; the full prompt-size analysis in Appendix H uses all five retention values.

The unified variants improve over the published baseline on average. Probability-weighted pooling exceeds the published Provence checkpoint by 0.084 mean hypervolume points on the accuracy-compression axis. The comparison is dataset-dependent: RECOMP (abs) is stronger on NQ and PopQA, while the document-target variants are generally stronger on the remaining datasets. The full table should therefore be read as a contextual pipeline comparison rather than as a universal dominance claim.

Two caveats on the margin. First, part of it reflects the corpus and chunking corrections of Section 6.1 rather than the ranking head itself, since those corrections raised our accuracy on NQ and PopQA. The comparison against published numbers is therefore between complete pipelines rather than between ranking heads in isolation. Second, RECOMP in its abstractive form is the one competitive baseline, reaching 0.550 on average and edging out our variants on NQ and PopQA. It does so by compressing at between 93% and 99%, which the hypervolume indicator rewards heavily in this coordinate system. Its aggressive operating points can score well in this metric even when the raw-accuracy trade-off is unattractive, and it collapses on SyllabusQA. This is a reminder that

Method	NQ	HotpotQA	PopQA	Syllabus	TyDiQA	PubMed	Mean	Pts
Ours (unified-rerank)								
CLS Pooling	0.638	0.535	0.503	0.563	0.617	0.699	**0.593**	2
Mean Pooling	0.615	0.531	0.492	0.570	0.602	0.691	0.583	2
Prob.-weighted Pooling	0.610	0.526	0.484	0.566	0.590	0.693	0.578	2
Span-or-Penalty	0.594	0.518	0.440	0.519	0.591	0.695	0.559	2
Published baselines								
RECOMP (abs)	0.632	0.501	0.505	0.377	0.632	0.655	0.550	1
Provence (standalone)	0.557	0.473	0.449	0.385	0.553	0.555	0.495	2
Provence (published)	0.552	0.466	0.451	0.390	0.561	0.543	0.494	2
DSLR	0.549	0.486	0.446	0.370	0.556	0.537	0.491	5
RECOMP (ext)	0.533	0.407	0.416	0.310	0.537	0.545	0.458	3
LLMLingua2	0.477	0.382	0.373	0.319	0.476	0.560	0.431	3
LongLLMLingua	0.466	0.378	0.353	0.332	0.446	0.556	0.422	3
Dense-X retrieval	0.433	0.336	0.356	–	0.475	–	0.400	1
RECOMP (ext+thr)	0.415	0.328	0.312	0.168	0.373	0.395	0.332	2

Table 6: Hypervolume in accuracy-compression space, reference point $(0, 0)$. Higher is better. The document-target variants generally improve over the published unified-rerank baseline on this axis, while RECOMP (abs) remains competitive on NQ and PopQA. The final column gives the number of operating points each method reports, which matters because a method with a denser frontier has more opportunity to accumulate hypervolume.

hypervolume in accuracy-compression space is compression-hungry, and that a method reporting a single very aggressive operating point can score well without being preferable at any accuracy a practitioner would accept.

The compression axis reverses the ordering among our four heads. On this axis CLS pooling comes out marginally ahead at 0.593, against 0.583 for mean pooling and 0.578 for probability-weighted pooling, which is the reverse of Table 9. The reversal is caused by the axis and not by the narrower operating range: holding the range fixed and changing only the axis reverses the ordering on its own. Compression counts the text the sentence pruner removed and is computed before any document is dropped, so it is blind to the ranking head's document selection. Since the ranking head is the component under study, we treat the prompt-size view of Section 6.3 as the one that speaks to it, and report this axis for comparability with published numbers.

E Accuracy against compression on every dataset

Figure 1 shows PubMed/BioASQ and HotpotQA. Figure 2 shows all six evaluation datasets on the same axes, against the context-pruning methods reported by Chirkova et al. (2025). The document-target heads generally sit above the published frontier, but the relative ordering depends on the dataset and baseline; span-or-penalty is usually weaker.

F Statistical method

Every comparison in Section 6.2 has one observation per (dataset, threshold, retention count) cell, so $6 \times 2 \times 5 = 60$ for accuracy. Hypervolume consumes the retention sweep inside the metric, leaving 12 observations, one per (dataset, threshold). All numbers are produced by a single script from the saved per-run results, with no GPU and no hidden steps.

The dependence problem. Grid cells are not independent, because ten of them share a dataset. The size of that dependence differs by comparison: taking a difference between two heads cancels dataset difficulty, which is the quantity that normally clusters, so what remains varies. We therefore measure it rather than assume it, and report three p-values so the choice is visible.

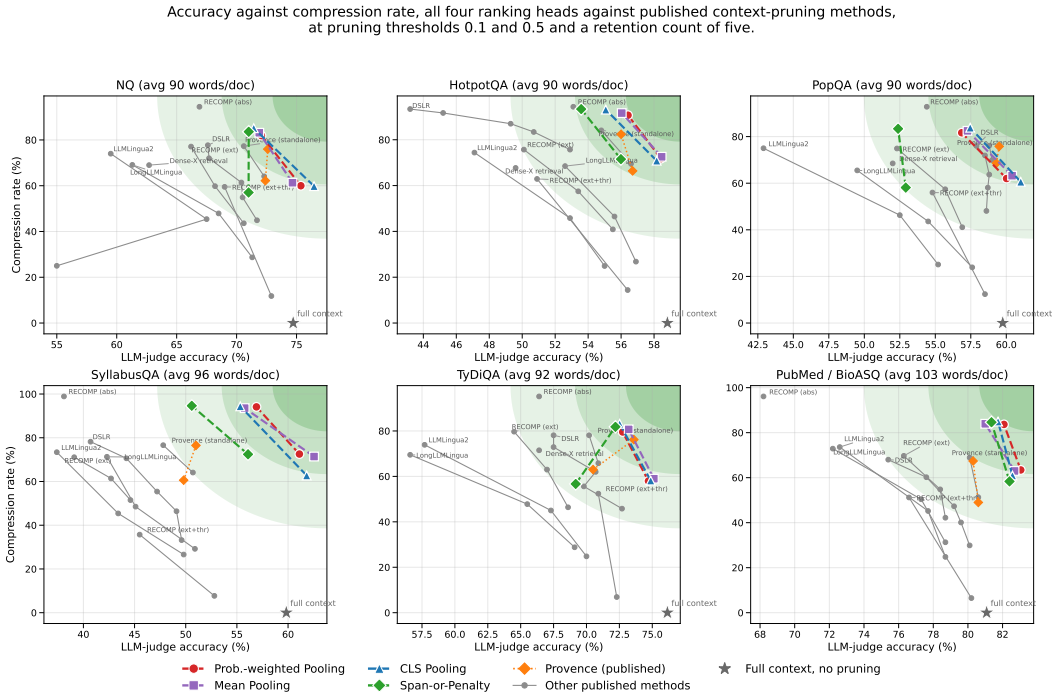


Figure 2: Accuracy against compression rate for all four ranking heads against published context-pruning methods, on all six evaluation datasets, at pruning thresholds 0.1 and 0.5 and a retention count of five. Our numbers come from the same per-run results as Table 5; the published points are the figures reported by Chirkova et al. (2025).

- p_{naive} treats every cell as independent. It is a sign-flip permutation over all cells, exact by enumeration when $n \leq 20$ and 200,000 Monte Carlo draws at a fixed seed otherwise. Reported only so the inflation is visible.
- p_{measured} is the value we quote. A one-way random-effects decomposition over datasets gives the intraclass correlation $\text{ICC} = (\text{MS}_\text{b} - \text{MS}_\text{w})/(\text{MS}_\text{b} + (m-1)\text{MS}_\text{w})$ for m cells per dataset. The design effect is $1 + (m-1)\text{ICC}$; the standard error is multiplied by its square root and the degrees of freedom reduced to $n/\text{deff} - 1$, then a two-sided t test. When $\text{ICC} = 0$ this reduces to the uncorrected test.
- $p_{\text{worstcase}}$ assumes the dependence is maximal. Cells are averaged within each dataset and the six dataset means are permuted exactly. This is equivalent to $\text{ICC} = 1$ and is floored at $2/2^6 = 0.031$, so with six datasets no result can report a smaller value.

Confidence intervals are 95%, two-sided, design-effect corrected, using the small-sample t critical value at the effective degrees of freedom. They therefore always agree with p_{measured} about whether zero is included.

Two things should be read alongside any of these. The ICC is itself estimated from six datasets, so its interval is wide and Appendix G reports a sensitivity analysis rather than a single value. And none of these tests addresses training-seed variance: every cell uses one trained checkpoint per head, so the replication unit is the evaluation condition and not the training run.

G Dependence structure and sensitivity

Table 7 reports, for each comparison, the share of variance lying between datasets, the intraclass correlation with its exact F-based interval, the design effect and the effective sample size.

Comparison	Var. between	ICC	ICC 95% CI	Deff	Effective n
Prob.-weighted − Mean	13.0%	0.038	$[-0.069, +0.488]$	1.34	44.8
Prob.-weighted − CLS	50.5%	0.537	$[+0.356, +0.923]$	5.83	10.3
Prob.-weighted − Span-or-Penalty	52.7%	0.554	$[+0.278, +0.899]$	5.98	10.0
Mean − CLS	53.8%	0.540	$[+0.290, +0.903]$	5.86	10.2
Mean − Span-or-Penalty	59.6%	0.602	$[+0.306, +0.909]$	6.41	9.4
CLS − Span-or-Penalty	45.0%	0.460	$[+0.153, +0.841]$	5.14	11.7

Table 7: Dependence structure of each comparison, accuracy axis, $n = 60$. A large ICC means the cells within a dataset are correlated and the naive test overstates the evidence. The probability-weighted against mean pooling comparison is the only one where the F test cannot reject the absence of clustering, which is why its cells can be pooled.

Comparison	0.00	0.05	0.10	0.15	0.20	0.30	0.40
Prob.-weighted − Span-or-Penalty	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	.0002	.0009
CLS − Span-or-Penalty	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	.0005	.0018
Mean − Span-or-Penalty	$< .001$	$< .001$	$< .001$	$< .001$	.0001	.0005	.0019
Prob.-weighted − Mean	.0114	.0360	.0676	.1011	.1339	.1943	.2466
Prob.-weighted − CLS	.0007	.0052	.0148	.0288	.0458	.0837	.1228
Mean − CLS	.0727	.1369	.1948	.2447	.2875	.3570	.4108

Table 8: p_{measured} recomputed under assumed values of the intraclass correlation, accuracy axis. The probability-weighted against mean pooling result holds to roughly $\text{ICC} = 0.07$ and is lost beyond it; the three span-or-penalty comparisons are unaffected across the whole range.

H Hypervolume on both cost axes

Table 9 reports the prompt-size axis, which is the one that reflects what the generator is charged for. Table 10 gives the significance of every pairwise hypervolume difference on that axis. The three

Ranking head	NQ	HotpotQA	PopQA	Syllabus	TyDiQA	PubMed	Mean
Prob.-weighted Pooling	0.6784	**0.5335**	0.5488	**0.5465**	**0.6887**	**0.7460**	**0.6236**
Mean Pooling	0.6746	0.5308	0.5525	0.5425	0.6841	0.7430	0.6212
CLS Pooling	**0.6804**	0.5256	**0.5557**	0.5315	0.6803	0.7443	0.6196
Span-or-Penalty	0.6501	0.5025	0.4889	0.4804	0.6585	0.7430	0.5873

Table 9: Hypervolume of each ranking head over its full operating range, in accuracy against prompt-size space, reference point $(0, 0)$. Higher is better. Each variant's frontier is taken over its ten operating points per dataset, spanning both pruning thresholds and all five retention counts $k \in \{1, \ldots, 5\}$. Per-dataset values are means over the two thresholds. Probability-weighted pooling attains the highest mean hypervolume and leads on four of the six datasets, but the margin over mean pooling is not statistically significant ($+0.0024$, 95% CI $[-0.0005, +0.0053]$, $p = 0.090$): collapsing each frontier to a single number leaves twelve paired observations rather than the sixty available on the accuracy axis. The three comparisons against span-or-penalty are significant on this axis ($p < 0.015$). Appendix F gives the method and Appendix H the full tables on both cost axes, together with the Pareto fronts these hypervolumes are computed from.

span-or-penalty comparisons are significant; none of the comparisons among the three document-target heads is, because each frontier collapses to a single number and only 12 paired observations remain.

Comparison	Mean	95% CI	p	Datasets won
Prob.-weighted − Span-or-Penalty	$+0.0364$	$[+0.0136, +0.0591]$	$\mathbf{0.009}$	6/6
CLS − Span-or-Penalty	$+0.0324$	$[+0.0097, +0.0550]$	$\mathbf{0.013}$	6/6
Mean − Span-or-Penalty	$+0.0340$	$[+0.0097, +0.0582]$	$\mathbf{0.015}$	5/6
Prob.-weighted − Mean	$+0.0024$	$[-0.0005, +0.0053]$	0.090	5/6
Prob.-weighted − CLS	$+0.0040$	$[-0.0037, +0.0117]$	0.249	4/6
Mean − CLS	$+0.0016$	$[-0.0042, +0.0075]$	0.537	3/6

Table 10: Pairwise hypervolume differences, accuracy against prompt-size space, $n = 12$.

The compression axis reverses the ordering among the pooling heads. On the accuracy-compression frontier of Table 6, mean pooling comes first at 0.5053 against 0.5035 for probability-weighted pooling and 0.4992 for CLS pooling, and the probability-weighted minus mean pooling difference is -0.0018, winning two of six datasets ($p = 0.538$). Neither direction is significant. The reversal is caused by the axis rather than by the operating range: compression counts the text the sentence pruner removed and is computed before any document is dropped, so it is blind to the ranking head's document selection, which is the component under study. We therefore treat the prompt-size view as the one that speaks to the ranking head, and report the compression axis for comparability with published methods.

I Teacher-score analysis

The same teacher cross-encoder scores an isolated, oracle-selected supporting span systematically lower and with substantially higher variance than the same query paired with the full source document, even though the span is by construction relevant. We observe this pattern on both NQ and MS MARCO oracle training data. It motivates using a document-level ranking target while leaving open the possibility that span-level supervision contains additional information.

J Metric dependence

The accuracy advantage of probability-weighted pooling over mean pooling appears under the SOLAR-10.7B judge and not under the lexical metrics recorded alongside it. At $k = 1$, pooled over all 12 conditions, the judge gives $+0.20$ points ($t = +2.20$) while F1 gives -0.06, ROUGE-L -0.05 and exact match -0.14. Exact match is close to zero for every system, because Llama-2-7b-chat produces verbose answers that rarely match a short gold string exactly, and F1 and ROUGE-L

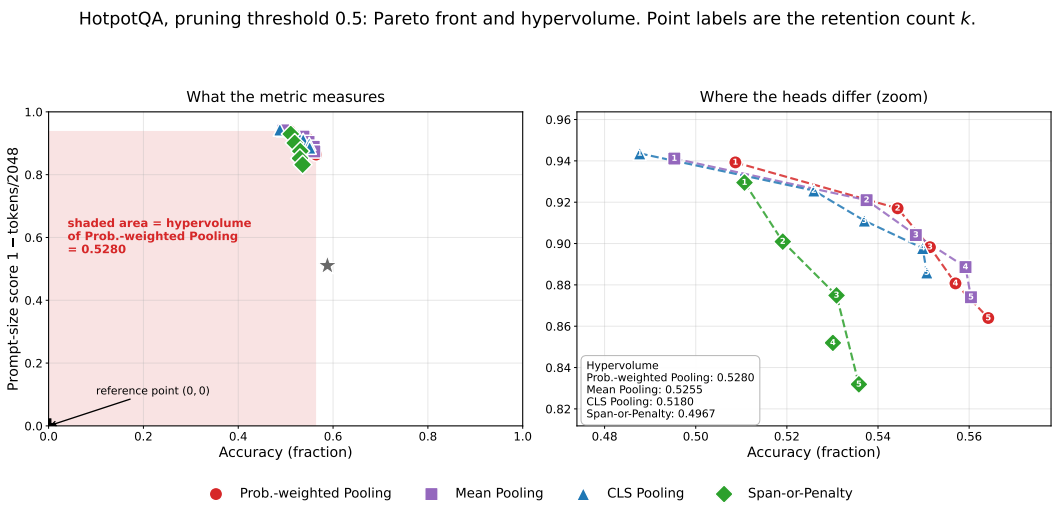


Figure 3: The hypervolume of Table 9, shown for one condition. *Left:* the objective space, with the region dominated by probability-weighted pooling shaded; its area against the fixed reference point $(0, 0)$ is the hypervolume, 0.5280 here. *Right:* the same data zoomed to the operating points, where the heads can actually be told apart. Point labels give the retention count k. Probability-weighted pooling attains the outermost front, mean pooling sits just inside it, and span-or-penalty falls away sharply as k grows. The four fronts are separated by well under one percent of the distance to the reference point, which is why the hypervolume differences among the three document-target heads are small and, as Appendix H reports, not statistically significant. Figure 4 gives every (dataset, threshold) panel.

partly measure response length rather than correctness. We regard them as weak instruments for this generator, but the dependence of the result on a single metric should be stated rather than omitted.

K Per-query analysis

The judge metric is binary per query, so only queries where the two heads disagree carry information about the comparison. At $k = 1$, the two pruning thresholds produce $54{,}940$ query–threshold observations from $27{,}938$ unique queries. Probability-weighted pooling is correct where mean pooling is wrong on $1{,}350$ observations and the reverse on $1{,}238$, leaving $2{,}588$ discordant pairs, or 4.7% of the query–threshold observations. An exact McNemar test on these pooled counts gives $p = 0.029$; this is a descriptive, size-weighted summary because the repeated observations are not independent. PopQA alone supplies $1{,}340$ of the discordant pairs. Of the twelve individual conditions only HotpotQA at threshold 0.5 is significant on its own (172 against 101, $p < 10^{-4}$, surviving Bonferroni correction at twelve tests). Judge sentinel values occur in 1.4% of rows and are excluded from both arms; we do not assume that this exclusion is automatically conservative.

L Limitations

Scope of the theory. The oracle-pooling result concerns fixed contextual states and the student's own readout on evidence-positive blocks. It does not cover evidence-absent blocks, CLS pooling, or multi-block score aggregation, and it does not guarantee teacher-score recovery or downstream accuracy. A smaller upper bound also does not establish a smaller actual score error. Section 5.3 gives the assumptions and the conditions needed to extend the consistency statement across training.

One trained checkpoint per head. The replication unit throughout is the evaluation condition, not the training run, so for a claim about the architecture rather than about these particular checkpoints the effective sample size is one. The natural remedy is to retrain probability-weighted and mean

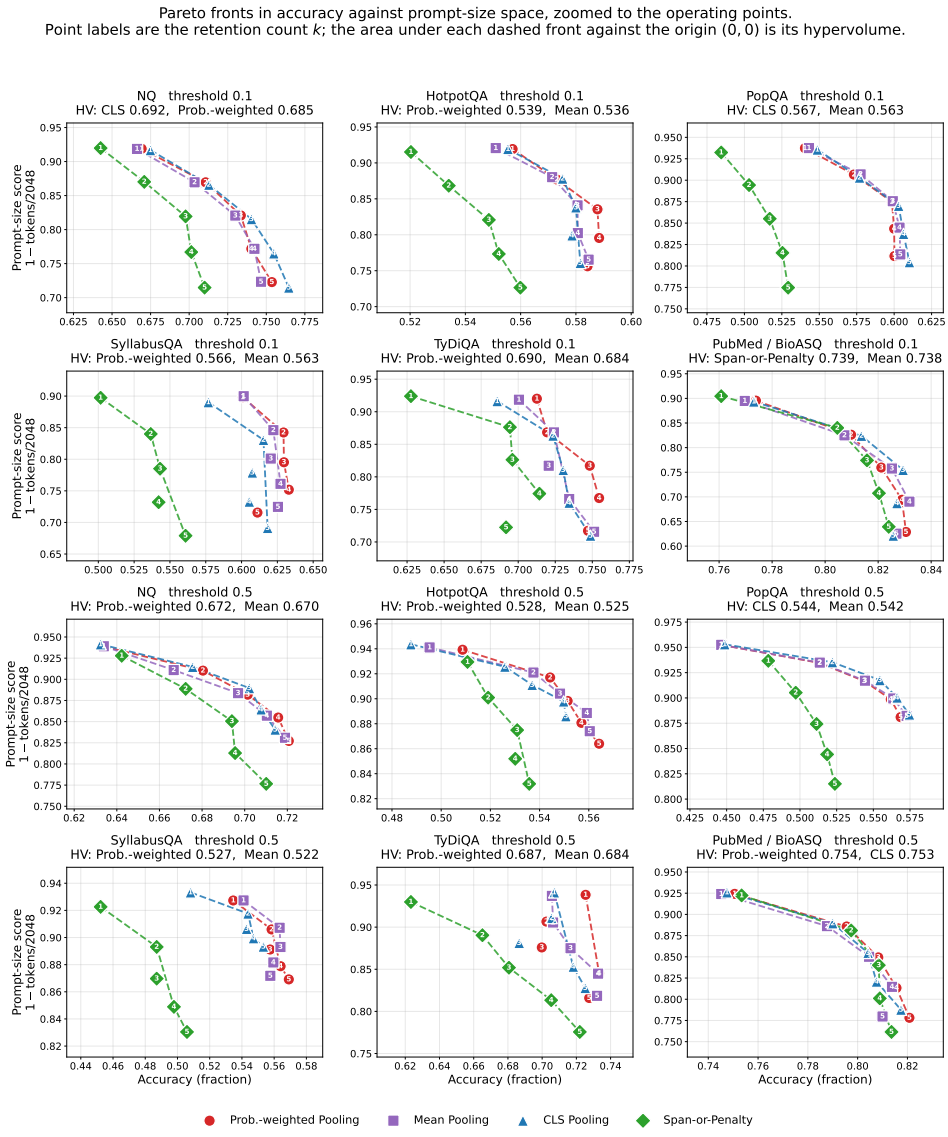


Figure 4: Pareto fronts for every (dataset, threshold) condition, zoomed to the operating points. Point labels are the retention count k; the two highest hypervolumes are given above each panel. Span-or-penalty is visibly separated on every dataset. The three document-target heads overlap closely, and which of them sits outermost varies by condition, which is what Table 10 quantifies.

pooling under several seeds and repeat the evaluation. We note that varying the generator or judge seed would not help: both decode greedily with the temperature fixed, so a seed has no effect on their output.

The effect is judge-specific. See Appendix J.

Significance is fragile. The intraclass correlation for the probability-weighted against mean pooling comparison is 0.038 with a 95% interval reaching $+0.49$, and the result holds only below roughly $\mathrm{ICC} = 0.07$ (Table 8). The effect is also small in absolute terms: Cohen's d_z is about 0.35 and the win rate is 33 of 60 conditions.

Hypervolume does not separate the pooling heads. See Appendix H.

One retention count comes from a different experiment batch. The runs at $k \in \{1, 2, 3, 4\}$ were produced together; those at $k = 5$ predate them and reuse the earlier baseline runs. The low value at $k = 5$ may therefore be partly a batch effect rather than an effect of retention, and no conclusion here rests on it.

Six datasets cap what any test can establish. The worst-case permutation floor is $2/2^6 = 0.031$, so no result on this suite can report a smaller value under the most conservative treatment of the dependence. Adding a large-query evaluation set would raise that ceiling.

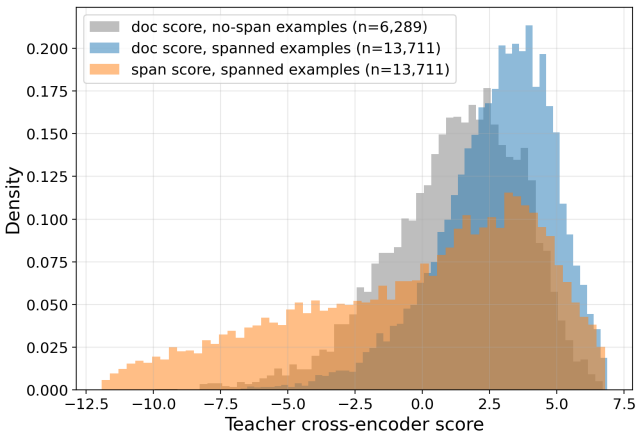


Figure 5: Teacher cross-encoder scores for full documents and isolated supporting spans on NQ training examples. Supporting spans receive lower and more variable scores.