Optimizing Compound Retrieval Systems

Harrie Oosterhuis*
Radboud University
Nijmegen, NL
harrie.oosterhuis@ru.nl

Rolf Jagerman
Google DeepMind
New York, US
jagerman@google.com

Zhen Qin
Google DeepMind
New York, US
zhenqin@google.com

Xuanhui Wang
Google DeepMind
Mountain View, US
xuanhui@google.com

Abstract

Modern retrieval systems do not rely on a single ranking model to construct their rankings. Instead, they generally take a cascading approach where a sequence of ranking models are applied in multiple re-ranking stages. Thereby, they balance the quality of the top-K ranking with computational costs by limiting the number of documents each model re-ranks. However, the cascading approach is not the only way models can interact to form a retrieval system.

We propose the concept of *compound retrieval systems* as a broader class of retrieval systems that apply multiple prediction models. This encapsulates cascading models but also allows other types of interactions than top-K re-ranking. In particular, we enable interactions with large language models (LLMs) which can provide relative relevance comparisons. We focus on the optimization of compound retrieval system design which uniquely involves learning where to apply the component models and how to aggregate their predictions into a final ranking. This work shows how our compound approach can combine the classic BM25 retrieval model with state-of-the-art (pairwise) LLM relevance predictions, while optimizing a given ranking metric and efficiency target. Our experimental results show optimized compound retrieval systems provide better trade-offs between effectiveness and efficiency than cascading approaches, even when applied in a self-supervised manner.

With the introduction of compound retrieval systems, we hope to inspire the information retrieval field to more out-of-the-box thinking on how prediction models can interact to form rankings.

CCS Concepts

• **Information systems** → **Search engine architectures and scalability**; **Retrieval models and ranking**; **Learning to rank**.

Keywords

Learning to Rank, Large Language Models, Ranking Distillation

ACM Reference Format:
Harrie Oosterhuis, Rolf Jagerman, Zhen Qin, and Xuanhui Wang. 2025. Optimizing Compound Retrieval Systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730051

*Work done while Harrie Oosterhuis was working at Google DeepMind.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730051

1 Introduction

Modern retrieval systems consist of many components that are applied together to construct rankings of documents [8]. Thereby, they can combine different prediction models that provide different ranking utility at different costs, such that their service is both responsive and provides high quality search results [18, 38]. Accordingly, it appears very important for the information retrieval (IR) field to consider the following question:

How should retrieval systems utilize multiple prediction models to form rankings in the most effective and efficient manner?

Over the past decade, the IR field appears to have reached a consensus that cascading retrieval systems are the answer to this question [1, 2, 5, 6, 9, 11, 12, 18, 19, 29, 38, 45, 47]. In the cascading paradigm, models are applied in sequence to re-rank the top results of the previous model [38]. The order of the sequence follows model scalability, which generally is inversely correlated with ranking effectiveness [6]. In other words, a retrieval model M_0 with minimal costs per document is used to create a first ranking (e.g., BM25 [33] or dense retrieval [12, 44]), its top-K_0 results are subsequently re-ranked by the next model M_1, its top-K_1 results are then re-ranked by M_2, and so forth. The cascading paradigm applies costlier models to fewer documents than cheaper models, this saves costs whilst it also uses the costlier models to improve upon the rankings of cheaper models [9, 27]. As a result, a cascading retrieval model can have an efficiency-utility trade-off that none of its models can obtain individually [2, 6, 38, 45].

However, the recent introduction of large language models (LLMs) has revealed limitations of the cascade paradigm. LLMs are extremely flexible in their application, as they can handle a virtually endless variety of prompts [4, 36, 39, 46]. Previous work found that using LLMs for predicting relative relevance differences instead of absolute judgements results in substantially better rankings [20, 21, 31, 42]. For instance, *pairwise relevance prompting* (PRP) [31] has state-of-the-art zero-shot ranking by generating pairwise relevance labels that each indicate whether one document is more relevant than another. The downside of standard PRP is that the number of pairs -and thus the number of LLM calls- scales quadratically with the number of documents. This makes PRP extremely costly, even as the final stage in a cascade retrieval system.

Inspired by this shortcoming and because the cascading paradigm has rarely been questioned in the IR field, we reconsider the question stated in the first paragraph. Our first contribution is introducing the concept of a *compound retrieval system* that encompasses any retrieval system that applies multiple predictions models. By keeping our definition broad and avoiding assumptions, we aim to explore the largest design space possible. We argue that compound retrieval system design depends on two essential questions:

(i) *What predictions should the component models make?*
(ii) *How should predictions be combined to construct a ranking?*

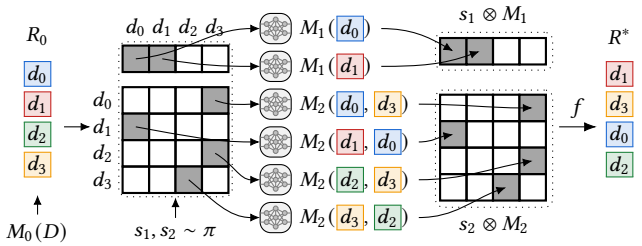

Figure 1: Overview of the compound retrieval system described in Section 4. A first-stage retrieval model M_0 retrieves documents to create a first ranking R_0; based on their position in R_0, the policy π decides which documents to apply the pointwise prediction model M_1 and pairwise prediction model M_2. Subsequently, the predictions of M_1 and M_2 are only gathered where activated and combined into a final ranking R^* using the score aggregation function f.

We note that the cascade paradigm has very specific answers: model predictions should be made on documents in the top-K of previous models; and predicted scores should be used to re-rank the same top-K [38]. Therefore, cascading retrieval systems are a subset of compound retrieval systems, which is thus a generalization.

We argue that the answers to these questions should depend on both the costliness and usefulness of the component model predictions, in addition to the efficiency constraints and ranking quality needs one has for their retrieval system. Accordingly, instead of providing prescriptive answers to these questions, our second contribution is a framework that optimizes the design of a compound retrieval system *automatically*. We break the system design down in a policy that decides what model predictions to gather, and a prediction aggregation function to construct a ranking out of the gathered predictions. Given a differentiable loss function, our framework can then optimize the system. For example, given a linear interpolation of a ranking utility and prediction costs, the framework will optimize the design for the corresponding effectiveness-efficiency trade-off. Importantly, it is capable of finding novel system designs that fall outside the cascading paradigm, e.g., by combining multiple predictions in a single re-ranking step, or by treating documents differently depending on their rank in the previous ranking.

In our experiments, we optimize a compound retrieval system that combines classic BM25 retrieval with pointwise and pairwise LLM predictions. We perform optimization both in a supervised setting where relevance labels are available, and in a self-supervised setting where the goal is to reconstruct the most-costly PRP ranking. Our efficiency target is to limit the number of LLM calls, we vary its importance to construct effectiveness-efficiency trade-offs of our model. Our results reveal that our optimized compound retrieval systems have better trade-off curves than cascading systems. In particular, we can closely match the nDCG of a top-1000 PRP re-ranker with only a fraction of its LLM calls, regardless of whether relevance labels are available for supervision. An analysis of our optimized designs reveal a large variety of novel strategies depending on the desired effectiveness-efficiency trade-off. For example, on the extremes, they default to BM25 to avoid costs or to PRP to maximize effectiveness; for trade-offs in the middle, they gather pointwise and pairwise predictions in targeted patterns based on the BM25 ranking. This bolsters our claim that compound retrieval system designs should adapt to its components and the desired system properties, without being constrained to the cascade paradigm.

To the best of our knowledge, this is the first work to consider alternatives to the existing cascade paradigm. Our work reveals a high potential for designs outside this paradigm, and for our optimization framework for finding such designs. We hope that it provides a starting point for a novel research direction that further explores compound retrieval system design and optimization.

2 Definition: Compound Retrieval Systems

Inspired by compound A.I. systems [43], we define compound retrieval systems as any retrieval system that constructs a ranking by using multiple prediction models. Conversely, a non-compound retrieval system uses a singular monolithic ranking model [38, 43]. Components are prediction models that give scores to individual or sets of documents, these can be machine learned models but do not need to be. We denote the set of components of a system as $\{M_0, M_1, \ldots\}$ and ϕ as the aggregation function that defines how the components interact with each other to construct rankings. Thus, we use $\phi(M_0, M_1, \ldots)$ to denote a compound retrieval system.

3 Background

3.1 Cascade ranking systems

There is a seeming consensus on the cascading paradigm for the trade-off between effectiveness and efficiency in the IR field. It underlies virtually every advancement in retrieval and ranking of the past decade, as (almost) every model is ultimately applied inside a cascade retrieval system [1, 2, 5, 6, 9, 11, 12, 18, 19, 29, 38, 45, 47].

Cascading ranking systems balance the quality of their rankings with their computation costs by applying several ranking models in sequence [38]. Generally, a first-stage retrieval model like BM25 [33] creates the first ranking of the collection, of which only a top-K_0 is used. Let M_0 be a first-stage ranking model and its top-K_0 ranking:

$$\begin{aligned} &R_0 = [d_1, d_2, d_3, \ldots d_{K_0}] \\ &\text{s.t. } \forall 1 \leq i \leq K_0, \forall d \in D \setminus [d_1, \ldots, d_{i-1}], \quad M_0(d_i) \geq M_0(d). \end{aligned} \tag{1}$$

It is important that $M_0(d)$ is scalable so that R_0 can be computed over a document collection D with reasonable costs. There is extensive literature on efficient first-stage retrieval [12, 23, 26, 37].

Subsequently, the next ranking model M_1 is applied to re-rank R_0 and produce a shorter ranking R_1. Because R_0 does not include the entire collection ($|D| \gg K_0$), the costs of M_1 can be higher as it is applied to much fewer documents. This process repeats for several stages that each apply a more costly model to fewer documents [9]. In our notation, the ranking at stage i is:

$$\begin{aligned} R_i = [d_1, d_2, \ldots, d_{K_i}] \text{ s.t. } \forall 1 \leq j \leq K_i, \ \forall d \in D \setminus [d_1, \ldots, d_{j-1}], \\ d_j \in R_{i-1} \wedge (d \notin R_{i-1} \vee M_i(d_j) \geq M_i(d)). \end{aligned} \tag{2}$$

In the framing of our work, the cascading ranking model is a subclass of the compound retrieval system, where $\phi_{\text{casc}}(M_0, \ldots, M_N) = R_N$, i.e., the aggregation function ϕ_{casc} only lets components interact through sequential top-K re-ranking operations [38].

3.2 Pairwise relevance prompting

LLMs are better at prediction relevance differences than absolute relevance values [20, 21, 31]. Thus, using an LLM to predict whether

one document is more relevant than another leads to better rankings than using predictions about individual documents [31, 42]. Accordingly, the *pairwise relevance prompting* (PRP) approach [31] uses an LLM with a pairwise prompt to predict the probability that one document should be ranked before another:

$$M_{\text{PRP}}(d_1, d_2) = \hat{P}_{\text{LLM}}(v_{d_1} > v_{d_2}). \tag{3}$$

To turn pairwise predictions into a valid ranking, PRP ranks documents by their predicted win-rate. Thus, formally, PRP uses the following scoring function to re-rank the first-stage ranking R_0:

$$M_{\text{PRP}}(d \mid R_0 \setminus d) = \frac{1}{2} \sum_{d' \in R_0 \setminus d} M_{\text{PRP}}(d', d) + 1 - M_{\text{PRP}}(d, d'). \tag{4}$$

Due to the costly nature of LLM predictions and since the number of pairs grows quadratically with the number of documents ($|R_0|^2 - |R_0|$), previous work applies PRP only to re-rank a top-K from a first stage ranker (M_0) [31, 42].

Qin et al. [31] propose strategies for reducing the number of pairs to consider, e.g., by mimicking sorting algorithms. However, this creates sequential processes that cannot be parallelized, for their impracticality, this work will not consider such approaches.

4 Method: A Framework for Optimizing Compound Retrieval Systems

This section proposes a compound retrieval system (Section 4.1) that is a generalization of a cascading system (Section 4.2). Subsequently, we introduce our optimization framework that learns how predictions can be aggregated (Section 4.3), and which predictions to gather (Section 4.4), we also show how our system can be made deterministic (Section 4.5). Finally, we summarize our approach (Section 4.7). A visual overview of our system is shown in Figure 1.

4.1 A compound retrieval system

In this method section, we describe a compound retrieval system with three components $\{M_0, M_1, M_2\}$: M_0 is a first-stage retrieval model, M_1 is a pointwise prediction model and M_2 is a pairwise prediction model. Our framework can easily be extended for other choices of component models (see Section 6.3 and 9).

Formally, our compound retrieval system describes a new ranking function that is based on the predictions of its underlying components. In contrast with cascading systems, our compound system can choose what $M_1(d) \in \mathbb{R}$ and $M_2(d, d') \in \mathbb{R}$ predictions it wishes to gather. Let π be the selection policy of our compound system, i.e, a probability distribution over the possible selections of the predictions that can be gathered. The vector s_1 indicates the selections made for M_1 and the matrix s_2 the selections for M_2:

$$s_1, s_2 \sim \pi, \qquad s_1 \in \{0,1\}^{K_0}, \qquad s_2 \in \{0,1\}^{K_0 \times K_0}. \tag{5}$$

Thus, s_1 is a binary vector of size K_0 whose indices correspond to the ranking of the first-stage model M_0, e.g., $s_1[1] = 1$ indicates that the prediction of M_1 of the document ranked first by M_0 is selected. Similarly, s_2 is a binary matrix of size $K_0 \times K_0$ representing every ordered pair of documents corresponding to the first-stage ranking R_0,[1] e.g., $s_2[4, 8] = 1$ indicates that the M_2 prediction for the 4th and 8th documents, as ranked by M_0 are selected. To keep notation brief, we indicate whether a document or a pair is selected by: $s_1(d) = s_1[R_0^{-1}(d)]$ and $s_2(d, d') = s_2[R_0^{-1}(d), R_0^{-1}(d')]$, where $R_0^{-1}(d)$ indicates the rank of d in ranking R_0.[2] Thus, $s_1(d)$ indicates whether $M_1(d)$ is selected and $s_2(d, d')$ whether $M_2(d, d')$ is.

Predictions that are not selected should be distinguishable from zero predictions, i.e., $s_1(d) = 1$ and $M_1(d) = 0$ should be differentiated from $s_1(d) = 0$ and $M_1(d) \neq 0$. For this reason, we introduce the masking operator $\otimes$ which outputs tuples that contain both selection indicators and masked predictions:

$$\begin{aligned} (s_1 \otimes M_1)(d) &= (s_1(d),\ s_1(d) \cdot M_1(d)), \\ (s_2 \otimes M_2)(d, D) &= ((s_2(d, d'),\ s_2(d, d') \cdot M_2(d, d')) : d' \in D). \end{aligned} \tag{6}$$

Finally, in order to construct a ranking, we use a function f that takes three inputs: $(R_0^{-1}(d), (s_1 \otimes M_1)(d), (s_2 \otimes M_2)(d, R_0))$. This function produces the scores to determine the final ranking, and thus, it can be interpreted as a final ranking function:

$$M^*(d \mid f, s) = f(R_0^{-1}(d), (s_1 \otimes M_1)(d), (s_2 \otimes M_2)(d, R_0)), \tag{7}$$

where the final ranking is the corresponding re-ranking of R_0:

$$\begin{aligned} R^*(\pi, f, s) = [d_1, d_2, d_3, \dots d_{K_i}] \\ \text{s.t. } \forall 1 \leq j \leq K_i,\ \forall d \in D \setminus [d_1, \dots, d_{j-1}], \\ d_j \in R_0 \wedge (d \notin R_0 \vee M^*(d_j \mid f, s) \geq M^*(d \mid f, s)). \end{aligned} \tag{8}$$

Accordingly, the component aggregation function of our compound retrieval system is defined as:

$$\phi_{\text{comp}}(M_0, M_1, M_2 \mid \pi, f) = R^*(\pi, f), \tag{9}$$

and determined by selection policy π and score aggregator f.

Correspondingly, the choice of π and f is highly important as it completely defines the system behavior. For this work, we propose a method that searches for the π and f by optimizing a given trade-off between efficiency and effectiveness. Formally, we aim to minimize a linear interpolation of a ranking loss and the cost of the system, for given $\alpha \in [0, 1]$ the loss for our compound system is:

$$\mathcal{L}_{\text{comp}}(f, \pi) = \alpha \mathcal{L}_{\text{ranking}}(\pi, f) + (1 - \alpha)\mathcal{L}_{\text{cost}}(\pi). \tag{10}$$

Our cost is the expected number of LLM calls:

$$\mathcal{L}_{\text{cost}}(\pi) = \mathbb{E}_{(s_1, s_2) \sim \pi}\left[\sum_{d \in R_0} s_1(d) + \sum_{d \in R_0} \sum_{d' \in R_0} s_2(d, d') \right]. \tag{11}$$

Section 4.4 further details how we optimize π. For the ranking loss $\mathcal{L}_{\text{ranking}}(\pi, f)$, we utilize existing learning-to-rank (LTR) loss functions for supervised learning (from labels) [14] and self-supervised learning (ranking distillation) [32], Section 4.3 discusses how these can be applied to our compound retrieval systems framework.

4.2 Comparison with cascades and PRP

As laid out in Sections 2 and 3, cascading retrieval systems are particular instances of compound retrieval models. In our framework, we can choose π and f such that the system becomes equivalent to a specific cascading system. For example, for a cascade re-ranking with M_1, we choose π as a deterministic policy that selects all pointwise predictions, and a ranking function f that outputs them:[3]

$$s_1 = \mathbf{1}, \quad s_2 = \mathbf{0}, \quad f(\cdots) = M_1(d). \qquad \text{(Cascade)} \tag{12}$$

[1] Pairs in s_2 are ordered since for PRP the order matters: $M_2(d, d') \neq M_2(d', d)$.

[2] Hence $R_0^{-1}(d)$ is the inverse of $R_0[i]$ which gives the document at rank i in R_0.

[3] We are describing equivalence with a cascade model with a single re-ranking step, but our framework is easily extended to multiple re-ranking steps, see Section 9.

Similarly, for PRP, we choose π to only select all pairwise predictions and rank with the PRP aggregation function (Eq. 4):

$$s_1 = \mathbf{0}, \quad s_2 = \mathbf{1}, \quad f(\cdots) = M_2(d, R_0). \qquad \text{(PRP)} \qquad (13)$$

Moreover, our framework also allows us to ignore all predictions from M_1 and M_2 and leave the first-stage ranking unchanged:

$$s_1 = \mathbf{0}, \quad s_2 = \mathbf{0}, \quad f(\cdots) = -R_0^{-1}(d) \equiv M_0(d). \text{ (First-Stage)} \quad (14)$$

Furthermore, our framework can also capture changes in parameters for these systems, e.g., using to re-rank a smaller top-K by only selecting predictions of that top-K and scoring accordingly.

This means our compound framework provides a generalization of all of these three approaches. Consequently, our optimization can choose from any of these approaches, e.g., PRP when high costs are allowed ($\alpha = 1$), first-stage retrieval when no costs are allowed ($\alpha = 0$) or a pointwise cascade re-ranking for a trade-off in between. Importantly, there are also many other possible interactions between components that fit in our framework. Thus, our framework can produce many compound retrieval systems that do no match existing cascading retrieval systems. Since there is no precedent, it is unclear what these interactions should be like, and our optimization framework might discover them for the first time.

To the best of our knowledge, this is the first work that explores these alternative compound retrieval systems.

4.3 Learning how to aggregate predictions

There are many choices possible for the scoring aggregation function f; for this work, we propose an extremely simple model as it allows us to interpret its behavior and ensures that it can be implemented in a large scale practical setting.

We break down the scoring function f into three parts: $f = (f_0, f_1, f_2)$, corresponding to the three component models. Accordingly, each deals with the rank in the first-stage ranking and the (masked) outputs of one of the components (Eq. 6 and 7):

$$f_0 : \mathbb{Z} \to \mathbb{R}, \quad f_1 : \mathbb{Z}_2 \times \mathbb{R} \times \mathbb{Z} \to \mathbb{R}, \quad f_2 : \mathbb{Z}_2 \times \mathbb{R} \times \mathbb{Z}^2 \to \mathbb{R}.$$

To start, f_0 takes as input the rank of document according to the first-stage ranker M_0 as r and returns the variable A_r:

$$f_0(r) = A_r. \qquad (15)$$

This is a *default* score: the score a document gets when no related predictions of M_1 or M_2 are used. We note that this score only depends on the rank of document in the first-stage ranking R_0.

The second function f_1 has as input a selection indicator $s \in \{0, 1\}$, m the (masked) prediction of a document given by M_1 (i.e., $M_1(d)$) and the rank in the first stage ranking r, its output is based on two variables B_r and C_r which are unique per r:

$$f_1(s, m, r) = s \cdot (B_r + C_r \cdot m). \qquad (16)$$

In other words, f_1 provides a zero score if no selection is made ($s = 0$), if a selection is made ($s = 1$) the score starts with value B_r and adds C_r proportional to the prediction m ($M_1(d)$). If $m \in [0, 1]$, this is a linear interpolation between B_r and $B_r + C_r$.

The final function f_2 also takes the selection indicator $s \in \{0, 1\}$ as input, along with m, the masked prediction for a document pair given by M_2 (i.e., $M_2(d_1, d_2)$) and the ranks of both document in the first stage ranking r_1 and r_2. Its scoring is also based on two variables B_{r_1,r_2} and C_{r_1,r_2} which are unique per rank pair (r_1, r_2):

$$f_2(s, m, r_1, r_2) = s \cdot (B_{r_1,r_2} + C_{r_1,r_2} \cdot m). \qquad (17)$$

Similarly, it is zero if the prediction is not selected; when selected, it starts with B_r and C_r is added in proportion to the prediction.

The ranking score of our final ranking model is the addition of all predictions related to a document transformed by our functions, this includes a summation over all pairs where d is the first document:

$$\begin{aligned} M^*(d \mid f, s) &= f(R_0^{-1}(d), (s_1 \otimes M_1)(d), (s_2 \otimes M_2)(d, R_0)) \\ &= f_0(R_0^{-1}(d)) + f_1(s_1(d), s_1(d) \cdot M_1(d), R_0^{-1}(d)) \\ &\quad + \sum_{d' \in R_0 \setminus d} f_2(s_2(d, d'), s_2(d, d') \cdot M_2(d, d'), R_0^{-1}(d), R_0^{-1}(d')). \end{aligned} \qquad (18)$$

Our ranking model is completely differentiable w.r.t. the variables A, B and C, and therefore, f can be optimized with standard LTR methods [14, 24]. However, since s are sampled binary variables, it does not work with standard stochastic gradient descent; the next section describes how we optimize its underlying selection policy.

4.4 Learning where to apply components

The optimization of the selection policy π is more complicated as the sampling operation is not differentiable. Many solutions to this issue exist, most notably one can compute its gradient through estimating it based on many samples, i.e., with REINFORCE [40]. However, this often has high variance which can prevent optimal convergence, and it is computationally very costly. For this reason, we apply the cheaper and more stable straight-through-estimator [3]; thereby, we estimate the gradient of each selection to be its probability:

$$\frac{\delta s_1(d)}{\delta \pi(s_1(d) = 1)} = \pi(s_1(d) = 1), \quad \frac{\delta s_2(d, d')}{\delta \pi(s_2(d, d') = 1)} = \pi(s_2(d, d') = 1).$$

This is a biased estimate of the actual gradient, nevertheless, for most purposes its is accurate enough whilst also providing simplicity and stability at low computational costs.

As a result, we can now apply stochastic gradient descent to our entire compound retrieval system from the loss in Eq. 10. For the policy π, what happens is that if the LTR loss indicates that the ranking score of a document should increase, the accompanying gradient will increase the probabilities of all selections that would result in an increase in score, or decrease probabilities if that results in a score increase. Vice versa, if document score should decrease, probabilities are pushed in the other direction. Similarly, the cost part of the loss will produce a gradient that decreases the probability of each selection, according to how much a selection would increase costs. Together, the interpolation of the ranking and costs losses results in a gradient that decreases most selection probabilities except for those that lead to an increase in ranking performance that outweighs the added selection cost.

4.5 Finding a deterministic policy

We have described our compound retrieval system and how its score aggregation function f and selection policy π can be optimized. So far, we have let π be a probabilistic policy, as this makes its optimization much easier. However, in practice, it is often more beneficial to have a deterministic policy, as these generally allow for

substantially more efficient implementations. For this reason, we add an additional step at the end of our optimization where we take a number of selection samples from π and select the sample with the lowest loss on a validation set. Let s_1^*, s_2^* indicate our best sample, the selection policy is then changed to a deterministic policy that always selects this sample: $\pi(s_1 = s_1^*, s_2 = s_2^*) = 1$.

4.6 Efficient representation and computation

In our description of our score aggregation function in Section 4.3, we noted that it was chosen for its simplicity that enables it to be implemented for large scale settings. We believe this is the case because it can be computed solely via basic matrix operations:

To start, all the model variables of our system in Eq. 15, Eq. 16 and Eq. 17 can be captured in one vector and two matrices where the top rows of the matrices are for f_1 and the rest for f_2. Similarly, a sample of selections can be represented in a single matrix, with the top row indicating the selection of pointwise predictions:

$$A \in \mathbb{R}^{K_0}, \quad B, C \in \mathbb{R}^{K_0 \times (K_0+1)}, \quad S = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \mathbb{Z}_2^{(K_0+1)\times K_0}. \tag{19}$$

Accordingly, we can gather all model predictions in a single matrix $X \in \mathbb{R}^{(K_0+1)\times K_0}$ following the same placement pattern:

$$X = \begin{bmatrix} M_1(R_0[1]) & M_1(R_0[2]) & \cdots & M_1(R_0[K_0]) \\ M_2(R_0[1], R_0[1]) & M_2(R_0[1], R_0[2]) & \cdots & M_2(R_0[1], R_0[K_0]) \\ \cdots & \cdots & \cdots & \cdots \\ M_2(R_0[K_0], R_0[1]) & \cdots & \cdots & M_2(R_0[K_0], R_0[K_0]) \end{bmatrix}.$$

During inference, X should only be populated with predictions that were selected in S. However, during training, X can be pre-computed once and stored, subsequently, we do not have to call the underlying component models during optimization.

Finally, with the matrix representations and the element-wise multiplication $\odot$, our ranking function can be re-formulated to:

$$M^*(D \mid f) = A + BS + C(S \odot X). \tag{20}$$

Due to the high number of variables, we recommend using an underlying model to produce them. In our experiments, we use a simple neural network and give it as input matrix indices to produce our variables A, B, C and our selection probabilities π. Accordingly, back-propagation can be used to optimize the neural network. One can also learn the variables directly, but we found that using an underlying model gives more regularity in model behavior which leads to better generalization and little overfitting. Because A, B, C and π can be pre-computed, there is no added costs during inference.

4.7 Summary

This section has described a framework for optimizing a compound retrieval system that can utilize the predictions of three component models: the first-stage retrieval model M_0, the pointwise relevance prediction model M_1, and the pairwise relevance prediction model M_2. For clarity, we summarize the system that results from our optimization by describing each step that it performs for inference:

(1) Compute a first ranking R_0 according to component M_0.
(2) Sample a selection indicator matrix S from the selection policy π (Eq. 19); selection probabilities can be pre-computed and kept ready in memory. We note that selections are based on document positions in the first ranking R_0, i.e., selections have the form of *the relevance prediction for the ith document in R_0*, or *the pairwise prediction for ith and jth documents in R_0*.
(3) Obtain the relevance predictions *only* for the selected single documents $M_1(d)$ and pairs $M_2(d, d')$. Fill the X matrix with the selected predictions (the rest remains empty).
(4) Compute a ranking score for each document through matrix operations on S and X according to Eq. 20. All other variables for this operation can be pre-computed and kept in memory.
(5) Return the final ranking by sorting according to the ranking scores (descendingly).

The main difference for optimization is that for gradient computation all predictions are needed, thus X has to be filled completely. Fortunately, X can be pre-computed for a static dataset. Gradient descent is applied to a combination of a ranking loss on the final ranking and a cost loss on the selection policy (Eq. 10).

As discussed in Section 4.2, an important property of our compound retrieval system is that it is a generalization of its component models. This means that its parameters can be set to give identical rankings to any of its components, or any top-K re-ranking of its components (if $K < K_0$). In other words, it can perform any top-K re-ranking with the pointwise model M_1, or with the pairwise model M_2, as a cascade retrieval model would do. It can also simply provide the first-stage retrieval ranking of M_0. Thus, if the optimization is successful, our compound retrieval system should provide an effectiveness-efficiency trade-off that is at least as good as any of its components. Since it can also capture behavior that none of the components can, it has a high potential of further improvements.

5 Method: Novel Ranking Losses

Our proposed framework aims to maximize ranking utility under efficiency constraints (see the trade-off loss in Eq. 10). Let ω_i be a weight per rank i and v_d a relevance label per document d, our utility function is a sum over the relevance label multiplied with the weight at each rank, e.g., for discounted cumulative gain (DCG) [15]:

$$U(y, v) = \sum_{i=1}^{|y|} \omega_i v_{y_i}, \qquad \omega_i^{\text{DCG@K}} = \frac{\mathbb{1}[i \leq K]}{\log_2(i+1)}. \tag{21}$$

5.1 Ranking distillation by lower bounding

Ranking distillation losses penalize the difference between two rankings to optimize a ranking model to behave similarly to another [35]. We wish to apply such a ranking loss in our trade-off loss in Eq. 10 for settings where relevance labels are unavailable. However, to optimize the trade-off there needs to be a correspondence between the distillation loss and ranking utility, existing distillation losses are not based on metrics, and thus, miss this property [32].

We propose a novel ranking distillation approach that is a bound on the difference in utility between two rankings [13]. Let y' be a new ranking and y be the original ranking, we aim to bound the maximum loss in utility when switching from y to y':

$$\mathcal{L}(y, y') \leq -\max_v U(y, v) - U(y', v) = -\min_v U(y', v) - U(y, v). \tag{22}$$

For ease of notation, we define $\omega_{d,y}$ as the weight at the rank of document d in ranking y, with $\omega_{d,y} = 0$ if d is not in y. Furthermore, we assume that relevance is always between zero and some maximum value V: $v_d \in [0, V]$. This allows us to make the following

derivation that leads to our novel loss $\mathcal{L}(y, y')$:

$$\begin{aligned}\min_v U(y', y) - U(y, y) &= \min_v \sum_{d \in y \cup y'} (\omega_{d,y'} - \omega_{d,y}) v_d \\ &= \min_v \sum_{d \in y} \min(0, \omega_{d,y'} - \omega_{d,y}) v_d = V \sum_{d \in y} \min(0, \omega_{d,y'} - \omega_{d,y}) \\ &\propto \sum_{d \in y} \min(0, \omega_{d,y'} - \omega_{d,y}) := -\mathcal{L}(y, y').\end{aligned} \tag{23}$$

We note that the step after (23) is valid because $v_d = 0$ is possible, hence any positive difference of $\omega_{d,y'} - \omega_{d,y}$ is always cancelled out by a zero relevance label in the minimization. For similar reasons, we can discard all documents that are in y' but not in y.

We see that our loss penalizes all documents that are ranked lower in y' than in the original y, but gives no reward for ranking documents higher. Penalties are based on differences in metric weights. Thus, the loss is minimized when the rankings are identical ($\mathcal{L}(y, y) = 0$). Importantly, $\mathcal{L}(y, y')$ stays within the range of possible DCG differences, and is proportional to the largest possible loss in DCG between the rankings. In theory, this makes it a good choice for optimizing our effectiveness-efficiency trade-off.

5.2 Optimizing ranking metrics with cutoffs

For our experiments, we adopt the approach of Qin et al. [30] by applying differentiable approximation of the ranks of documents: $\widetilde{\text{rank}}(d, y) \approx \text{rank}(d, y)$. These are straightforwardly used to approximate the metric weights of documents, e.g., for DCG:

$$\tilde{\omega}^{\text{DCG}}_{d,y} = \log_2(\widetilde{\text{rank}}(d, y) + 1)^{-1} \approx \omega^{\text{DCG}}_{d,y}. \tag{24}$$

However, for ranking metrics with cutoffs, e.g., DCG@K, an additional approximation is needed for the cutoff ($\mathbb{1}[\text{rank}(d, y) \leq K]$). Jagerman et al. [14] apply a sigmoid function to approximate cutoffs ($\mathbb{1}[\text{rank}(d, y) \leq K] \approx \text{sigmoid}(\widetilde{\text{rank}}(d, y) - k)$), but we find that its gradient diminishes too quickly. As a result, if a document's (approximated) rank is far from the cutoff, the gradient is virtually zero, and thereby, gradient descent is ineffective.

As an alternative, we propose the following approximation which has a gradient that diminishes much slower:

$$\mathbb{1}[i \leq K] \approx \max(i - K + 1, 1)^{-1}. \tag{25}$$

Additionally, since the progression of the weight function beyond the cutoff rank has no effect on the ranking metric, we prevent it from affecting our approximation. This is done by making K the maximum of the input to the discounting function, together with our cutoff approximation our differentiable document weight is:

$$\omega^{\text{DCG@K}}_{d,y} \approx \tilde{\omega}^{\text{DCG@K}}_{d,y} = \frac{\max(\widetilde{\text{rank}}(d, y) - K + 1, 1)^{-1}}{\log_2(\min(\widetilde{\text{rank}}(d, y), K) + 1)}. \tag{26}$$

6 Experimental Setup

Our experiments compare the effectiveness-efficiency trade-offs of compound and cascade systems. Our implementation is available at: https://github.com/google-deepmind/compound_retrieval.

6.1 Dataset

For our experiments we use the TREC-DL (2019-2022) dataset [7]. It consists of documents and queries together with human-annotated relevance judgments for query-document pairs. We randomly split the queries to create validation and test sets of 50 queries each, the remaining queries are the training set. For cross-validation, 50 random splits are generated to repeat our experimental runs over. Our setting is a top-1000 re-ranking task with BM25 as the first stage-ranker, thus, per query, only documents in the top-1000 of BM25 are used, the rest are discarded.

6.2 LLM prompts and predictions

Our compound retrieval model is built on the first-stage ranker BM25 [33] and the *Gemini 1.5 Flash* LLM [36]. We utilize both a pointwise and a pairwise prompt and use *scoring mode* to obtain the probabilities of predictions. For the pointwise prompt, we use the binary relevance prompt, first introduced by Liang et al. [17], using the same prompt as the one described in Figure 1a of [31]:

$$\text{LLM}_{\text{point}}(d) = \hat{P}_{\text{LLM}}(\text{'}Yes\text{'} \mid \text{prompt}, d). \tag{27}$$

We normalize the predicted probability based on the model logits for the answers '*Yes*' and '*No*', such that the probabilities of these two answers sum to one. Similarly, we use the pairwise PRP prompt described in Figure 2 of [31].

$$\text{LLM}_{\text{pair}}(d, d') = \hat{P}_{\text{LLM}}(\text{'}Passage\ A\text{'} \mid \text{prompt}, d, d'). \tag{28}$$

Importantly, passage A is always presented first (corresponding to d) and passage B second (d'), and again, we normalize the probabilities based on the logits for the phrases '*Passage A*' and '*Passage B*'.

6.3 Component models

For increased performance, we provide our system with different transformations of each prediction, this makes it an extension of the system described in Section 4. In the terms of our formal framework, this results in eight component models in our system:

$$\begin{aligned} M_0(d) &= \text{BM25}(d), \\ M_1(d) &= \text{LLM}_{\text{point}}(d), & M_2(d) &= \text{round}(M_1(d)), \\ M_3(d, d') &= \text{LLM}_{\text{pair}}(d, d'), & M_4(d, d') &= \text{round}(M_3(d, d')), \\ M_5(d, d') &= 1 - \text{LLM}_{\text{pair}}(d', d), & M_6(d, d') &= \text{round}(M_5(d', d)), \\ M_7(d, d') &= \text{sign}(\text{LLM}_{\text{point}}(d) - \text{LLM}_{\text{point}}(d')). \end{aligned} \tag{29}$$

In actuality, all component models are based on the underlying LLM (and BM25), but in our formal framework each model outputs a single prediction, thus formally we have eight components. Interestingly, this means we are not just learning where to apply the LLM but also how to prompt it and how to transform its predictions.

Part of our transformations are based on rounding (M_2, M_4, M_6, M_7) as previous work on PRP found rounding to result in better ranking [32]. Additionally, previous work found that the order of presentation in a pairwise prompt matters, i.e., $\text{LLM}_{\text{pair}}(d, d') \neq 1 - \text{LLM}_{\text{pair}}(d', d)$ [32]. Therefore, we also use predictions of pairs where a document is the second in the prompt (M_5, M_6). Lastly, we found that most pointwise LLM predictions of relevance are close to zero or one, making them uncalibrated for ranking. As a solution, we include pseudo-pairwise predictions that are simply the sign of the difference between two pointwise predictions (M_7), this adds stability to the optimization without any re-calibration of our LLM.

Our selection policy is still based on the original pointwise and pairwise predictions, we simply consider one of the transformed predictions selected if its underlying LLM prediction(s) are selected.

Thus, our selections are still a sampled vector $s_{\text{point}} \in \mathbb{Z}_2^{K_0}$ and a sampled matrix $s_{\text{pair}} \in \mathbb{Z}_2^{K_0 \times K_0}$ from which the corresponding selections $(s_1, \ldots, s_7)$ can be inferred by a simple mapping:

$$\begin{aligned} s_{\text{point}}(d) = 1 &\longleftrightarrow s_1(d) = 1 \wedge s_2(d) = 1, \\ s_{\text{pair}}(d, d') = 1 &\longleftrightarrow s_3(d, d') = 1 \wedge s_4(d, d') = 1, \\ s_{\text{pair}}(d', d) = 1 &\longleftrightarrow s_5(d, d') = 1 \wedge s_6(d, d') = 1, \\ s_{\text{point}}(d) = 1 \wedge s_{\text{point}}(d') = 1 &\longleftrightarrow s_7(d, d') = 1. \end{aligned} \tag{30}$$

Similarly, our scoring function f consists of eight linear subfunctions $f = (f_0, f_1, \ldots, f_7)$, each corresponding to one of the models. The first function f_0 is the same as in Eq. 15; the functions corresponding to pointwise components (f_1, f_2) follow Eq. 16 but have separate B_r and C_r variables per function; likewise, the functions for the pairwise components $(f_3, \ldots, f_7)$ follow Eq. 17 with separate variables per function as well. Our final scoring function becomes:

$$\begin{aligned} M^*(d \mid f, s) = {} & f_0(R_0^{-1}(d)) + f_1(s_1(d), s_1(d) \cdot M_1(d), R_0^{-1}(d)) \\ & + f_2(s_2(d), s_2(d) \cdot M_2(d), R_0^{-1}(d)) + \\ & \sum_{i \in \{3,\ldots,7\}} \sum_{d' \in R_0} f_i(s_i(d, d'), s_i(d, d') \cdot M_i(d, d'), R_0^{-1}(d), R_0^{-1}(d')). \end{aligned} \tag{31}$$

Analogous to our simple model, this can be rewritten to matrix operations following the approach of Section 4.6.

6.4 Selection policy and scoring function

As discussed in the final paragraph Section 4.6, we use two underlying neural networks to build the variable matrices to increase regularity and generalization capability. Both networks have three hidden layers of 64 units with sigmoid activation functions. The first network takes as input a rank $r \in \{1, \ldots, |R_0|\}$ (corresponding to an index for a vector) and outputs a value for A_r; values for B_r and C_r for each pointwise model; and a value for the selection policy $\pi(s_{\text{point}}[r] = 1)$ (Eq. 5). Similarly, the second network takes as input a pair of ranks (r, r') and outputs a value for $A_{r,r'}$; values for $B_{r,r'}$ and $C_{r,r'}$ for each pairwise model; and a value for the selection policy $\pi(s_{\text{pair}}[r, r']) = 1$. Thus, with only two small neural networks, every variable for our scoring function is generated.

6.5 Optimization of trade-off curves

Optimization is performed through stochastic gradient descent, our entire dataset and pre-computed LLM predictions fit in memory enabling full batch optimization. As our loss functions, in the supervised setting, we optimize the DCG@K loss using the rank and cutoff approximations from Section 5.2 [14]; in the self-supervised setting, we apply the ranking distillation loss from Section 5.1 with DCG@K weights, we refer to this loss as distil-DCG@K. The distillation loss is computed with respect to the PRP ranking, as this is expected to maximize effectiveness. Thus, in our supervised setting, we use labels to optimize the DCG of our system under efficiency constraints; and in the self-supervised setting, we optimize a system to produce rankings similar to PRP but at higher efficiency.

We optimize each individual system through 15000 gradient steps with the Adamax optimizer [16]. At the end of optimization, the parameters are chosen that produced the highest validation loss, subsequently, the learned selection policy is made deterministic as described in Section 4.5 with 250 samples.

To create our trade-offs, we use 200 values for α (Eq. 10) in a geometric sequence from 1 to 10^{-5}. A compound system is optimized for each value with a different random seed, subsequently, we discard all systems that have a higher costs and a worse validation loss than others. From the remaining points, a curve is created by linear interpolation between the points, this is required for the computing of a mean curve and its standard deviation for our evaluation. Separate curves are generated for $K \in \{25, 100, 1000\}$.

6.6 Evaluation and baselines

Our evaluation metrics match our optimization settings: top-K normalized DCG (nDCG@K) [15] and distil-DCG@K. For the cutoff K, we always report the performance of the method that optimized a matching K (e.g., for nDCG@100, we report the mean for our systems that optimized DCG@100 or distil-DCG@100). All our reported results are (test-set) averages of 50 trade-off curves created by 50 independent runs of 50 different splits of data.

As baselines, we apply the pointwise ranking model and PRP as a top-K re-rankers over the top-1000 of BM25, thus simulating a cascading retrieval system with a single re-ranking stage. Additionally, we apply a version of PRP that uses half the number of pairwise comparisons, by only selecting pairwise comparisons in one direction. To create trade-off curves, we vary K from 1 to 1000; baseline results are also averages over the same 50 data splits.

7 Experimental Results

7.1 Effectiveness-efficiency trade-offs

Figure 2 displays the effectiveness-efficiency trade-off curves of our compound retrieval systems and baseline systems, in terms of nDCG@K and distil-DCG@K over the number of LLM calls per query (N). Table 1 reports nDCG@100 for specific N values of all systems with results of significance testing.

The trade-off curves for the pointwise re-ranker and PRP reveal that PRP ultimately reaches much higher nDCG, but for equal N, pointwise always outperforms PRP. The salient difference is that after using $N = 1000$ predictions to re-rank the entire top-1000, the pointwise system cannot increase N any further. Interestingly, PRP with half the pair predictions is more effective than standard PRP.

Surprisingly, for all $K \in \{25, 100, 1000\}$, our supervised compound system reaches higher nDCG@K than the baselines even at maximal costs: It outperforms pointwise with $N = 10^3$ LLM calls, PRP with $N = 10^6$ and PRP half-pair with $N = \frac{10^6}{2}$. Thus, the compound system must have learned aggregation strategies that improve the standard behavior of its component models. We think this is achieved because the compound model can consider a mixture of predictions when re-ranking, whereas the cascading system only uses the predictions of a single model. Thus, for example, the compound system might penalize documents ranked low by BM25, so that they require more positive pointwise and PRP predictions to reach the front of the ranking. This is surprising since compound retrieval systems are designed for balancing effectiveness with efficiency, and not for direct effectiveness improvements.

Furthermore, the supervised compound system clearly provides the best trade-off curves for nDCG when $N > 200$, where it matches and outperforms pointwise, reaches PRP's performance with over ten times fewer LLM calls, and has similar gains over PRP half-pairs.

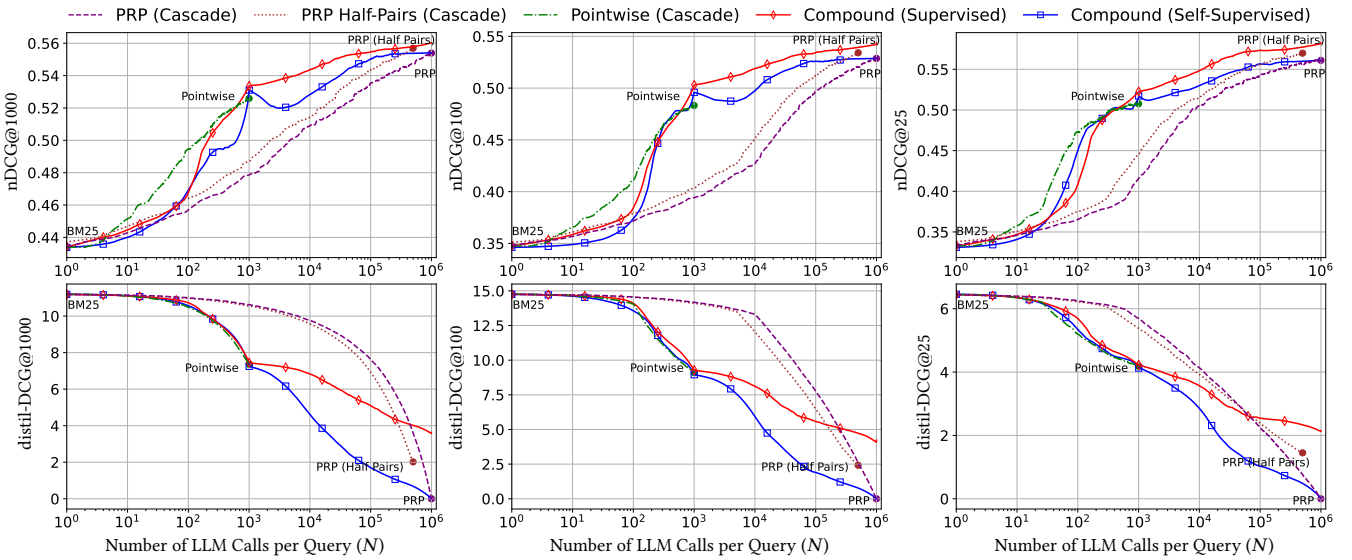

Figure 2: Effectiveness-efficiency trade-off curves (averages over 50 runs), created by optimizing compound systems with various trade-off weight, and varying the top-K to re-rank for cascade systems. Annotations indicate highest baseline effectiveness.

Table 1 indicates that these improvements are statistically significant for nDCG@100. However, when $N < 200$, the nDCG from both compound systems drops below that of pointwise, thus, it appears our optimization is less effective for DCG under such extreme efficiency constraints. We have ruled out overfitting but could not determine the cause, we speculate our gradient approximations degrade under extreme sparsity. The self-supervised compound system does not reach the same nDCG curve as its supervised counterpart, this is likely because it is optimized for distillation instead of ranking performance. Nevertheless, without additional labels, it provides a considerably better nDCG trade-off than PRP.

For distil-DCG, an outstanding trade-off curve is achieved by the self-supervised compound system; it closely matches pointwise until $N = 10^3$ when it starts to considerably outperform all other systems until perfect distil-DCG is reached at $N = 10^6$. This reveals that the self-supervised compound system can provide rankings much more similar to PRP's top-1000 re-ranking than when using PRP as a re-ranker on a smaller top-K. As expected, the supervised compound system provides a curve that performs worse and does not converge at zero. This can be explained by the fact that it outperforms PRP in terms of nDCG, it is thus intentionally converging on a different ranking than PRP, whilst this leads to significantly higher nDCG, it naturally produces a non-zero distillation loss.

To summarize, our optimized compound retrieval systems provide significantly better overall effectiveness-efficiency trade-off curves. In particular, supervised optimization leads to the best nDCG results, including 10× efficiency improvements over PRP, but we note that our systems falter on very sparse strategies ($N < 200$).

7.2 Strategies learned by compound systems.

To investigate the behavior of our compound systems, we examined their learned selection policies. We found these to vary widely, depending on the optimization loss and efficiency constraints applied. Figure 3 displays eight example policies that exhibit several recurring patterns: 1) Example (a) learned a cascading behavior in the self-supervised setting that only gathers pointwise predictions on the top-274 of BM25. 2) Example (b) gathers pairwise comparisons of the top-50 with the top-100 in one direction, appearing as a small triangle. 3) Examples (c) and (d) combine pointwise and pairwise predictions, a typical behavior in the supervised setting. 4) Many policies gather all pairwise comparison with a top-K in a single direction, which appears as black bars on the matrix edges, e.g., in examples (d), (e) and (g). This seems to be an efficient strategy to find top documents. 5) Example (f) and (h) display a typical pattern where all pairwise comparisons are gathered for pairs that are both in a top-K, fewer are gathered for pairs with only one in the top-K, and very few where neither are in the top-K. Importantly, except for example (a), none of these behaviors can be recreated by a cascading retrieval system, thus, each reveals a newly discovered strategy that only compound retrieval systems can perform.

Table 1: NDCG@100 for varying numbers of LLM calls. Results are means over 50 independent runs, brackets show standard deviations; △/▽ indicate significantly higher/lower results over the best baseline ($\rho < 0.01$, two-sided t-test).

# LLM calls	$N = 10^2$	$N = 10^3$	$N = 10^4$	$N = 10^5$
Cascading Retrieval Systems				
PRP	0.373 (0.0311)	0.395 (0.0321)	0.427 (0.0322)	0.497 (0.0329)
PRP (Half)	0.379 (0.0320)	0.404 (0.0319)	0.451 (0.0328)	0.513 (0.0326)
Pointwise	**0.411 (0.0326)**	0.483 (0.0335)	-	-
Compound Retrieval Systems				
Self-super.	$0.374^{\triangledown}$ (0.0285)	$0.496^{\triangle}$ (0.0331)	$0.497^{\triangle}$ (0.0368)	0.526 (0.0331)
Supervised	$0.384^{\triangledown}$ (0.0323)	**$0.502^{\triangle}$ (0.0336)**	**$0.519^{\triangle}$ (0.0320)**	**$0.534^{\triangle}$ (0.0320)**

8 Related Work

Our work is most related to existing literature on cascading retrieval system optimization [9, 19, 38]. When introducing cascading systems Wang et al. [38] propose using AdaRank [41] to optimize the cascade structure and parameters; Gallagher et al. [10] extends this approach with a joint optimization of the underlying ranking models; a direction continued by Qin et al. [29] and Zheng et al. [47].

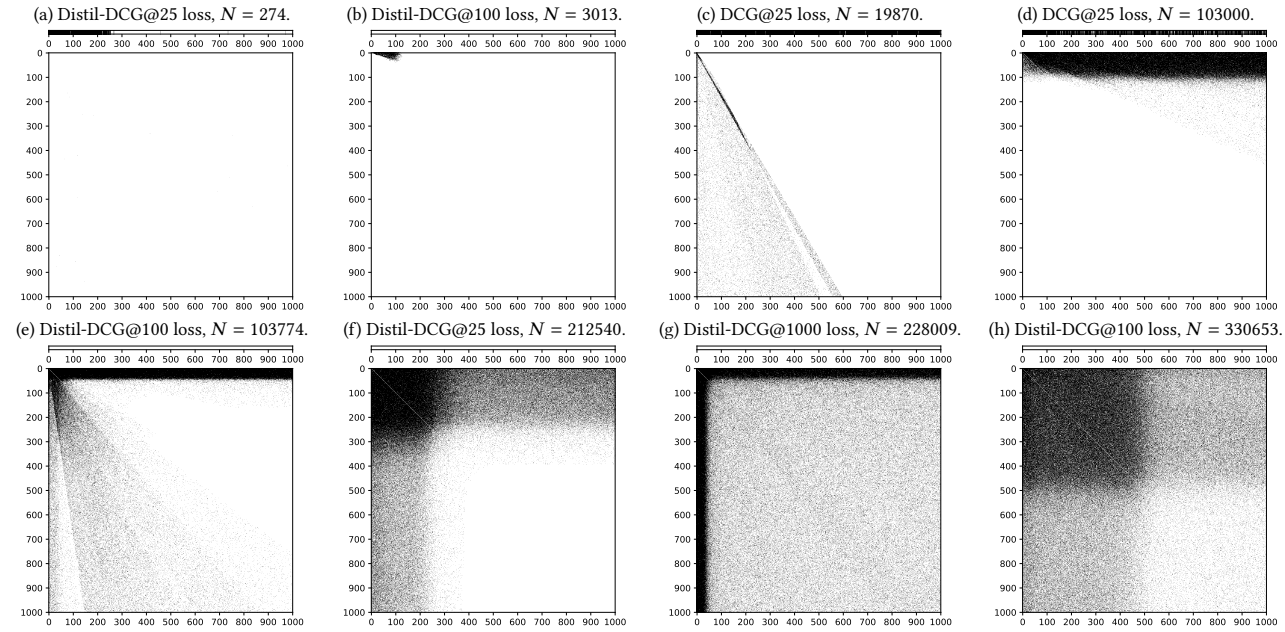

Figure 3: Examples of (deterministic) selection policies of our optimized compound retrieval systems. Top bars display the selection of pointwise predictions, square matrices that of pairwise predictions (black means selected). Pixels indices correspond to the first-stage ranking (e.g., the ith pixel in the top-bar indicates whether the pointwise prediction for ith document in the BM25 ranking was selected). Above each policy is the ranking loss used for optimization and its total number of selections N.

To the best of our knowledge, no previous work has considered alternatives to cascading retrieval system designs.

Several works use LLMs to re-rank by directly generating a permutation [22, 25, 28, 34], they encode the entire document set [28, 34], or predict over subsets and aggregate (a.k.a. listwise comparisons) [22, 25]. Because we find scalability essential, we only consider approaches that execute a single round of LLM predictions in parallel. Pointwise relevance generation [17] and pairwise comparisons [31] meet this requirement. Qin et al. [31] propose sorting strategies for PRP with fewer pairwise predictions, but these are no longer scalable. Liusie et al. [21] also optimize a selection policy to gather pairwise comparisons from an LLM, but for evaluating natural language generation performance, not retrieval.

9 Conclusion and Future Work

This work has proposed the concept of compound retrieval systems as any retrieval system that uses multiple prediction models to create rankings. This is a generalization of cascade retrieval systems that are limited to sequential top-K ranking. Furthermore, we have introduced a framework for optimizing compound retrieval systems for a given effectiveness-efficiency trade-off in supervised or self-supervised settings. Our framework optimizes system design through two fundamental tasks: (i) learning what predictions to gather from component models, and (ii) learning how to aggregate these predictions to construct a ranking. In our experiments, we optimized compound retrieval systems that combine BM25 with an LLM under pointwise and pairwise prompting. Our experimental results show that our optimized systems provide significantly better trade-off curves than its component models (applied in a cascading system), but leave room for improvement under extreme efficiency constraints. An analysis of the learned policies reveals that our compound systems apply a diverse set of strategies that adapt to the optimization objective, including existing strategies such as top-K re-ranking, but mostly consisting of never-before-seen strategies that are unique to compound retrieval systems. Lastly, we note that many of the observed advantages were achieved without supervision of relevance labels, indicating a very wide applicability.

This work challenged the cascading paradigm in the IR field, and thereby, revealed an enormous potential for optimized compound retrieval systems, that we hope opens many exciting new directions for future research. Finally, there are limitations of our work that also hint at the many possible extensions of our framework: Firstly, the compound system we have presented in this work only performs a single re-ranking step, and accordingly, was also only compared with single re-ranking cascading systems. A natural next direction is to extend our framework to allow for intermediate re-ranking steps, i.e., with an approach as Gallagher et al. [10], this would make it a generalization of multi-stage cascades. Secondly, our scoring aggregation functions were limited to simple linear transformations, more complex non-linear functions could potentially improve performance much further. Thirdly, our component models were limited to pointwise and pairwise prediction models, but many other models could be included, e.g., for set-wise predictions, query-transformations, etc. With the introduction of compound retrieval systems, the possibilities appear endless.

Acknowledgments

This research was supported by the Google Visiting Researcher program. Any opinions, findings and recommendations expressed in this work are those of the authors and are not necessarily shared or endorsed by their respective employers or sponsors.

We thank Don Metzler, Michael Bendersky, Mohanna Hoveyda and our reviewers for useful discussions and constructive feedback.

References

[1] Nima Asadi and Jimmy Lin. 2013. Document vector representations for feature extraction in multi-stage document ranking. *Information retrieval* 16 (2013), 747–768.

[2] Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 997–1000.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[4] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[5] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J Shane Culpepper. 2017. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 445–454.

[6] Charles LA Clarke, J Shane Culpepper, and Alistair Moffat. 2016. Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. *Information Retrieval Journal* 19 (2016), 351–377.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M Voorhees, and Ian Soboroff. 2021. TREC deep learning track: Reusable test collections in the large data regime. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2369–2375.

[8] W Bruce Croft, Donald Metzler, and Trevor Strohman. [n. d.]. *Search engines: Information retrieval in practice*. Vol. 520.

[9] J Shane Culpepper, Charles LA Clarke, and Jimmy Lin. 2016. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proceedings of the 21st Australasian Document Computing Symposium*. 17–24.

[10] Luke Gallagher, Ruey-Cheng Chen, Roi Blanco, and J Shane Culpepper. 2019. Joint optimization of cascade ranking models. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 15–23.

[11] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43*. Springer, 280–286.

[12] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)* 40, 4 (2022), 1–42.

[13] Shashank Gupta, Harrie Oosterhuis, and Maarten de Rijke. 2024. Practical and Robust Safety Guarantees for Advanced Counterfactual Learning to Rank. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 737–747.

[14] Rolf Jagerman, Xuanhui Wang, Honglei Zhuang, Zhen Qin, Michael Bendersky, and Marc Najork. 2022. Rax: composable learning-to-rank using Jax. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3051–3060.

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

[16] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=iO4LZibEqW Featured Certification, Expert Certification.

[18] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1557–1565.

[19] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural Re-ranking in Multi-stage Recommender Systems: A Review. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5512–5520. doi:10.24963/ijcai.2022/771 Survey Track.

[20] Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulic, Anna Korhonen, and Nigel Collier. 2024. Aligning with Human Judgement: The Role of Pairwise Preference in Large Language Model Evaluators. *CoRR* abs/2403.16950 (2024). https://doi.org/10.48550/arXiv.2403.16950

[21] Adian Liusie, Vatsal Raina, Yassir Fathullah, and Mark Gales. 2024. Efficient LLM Comparative Assessment: A Product of Experts Framework for Pairwise Comparisons. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 6835–6855. doi:10.18653/v1/2024.emnlp-main.389

[22] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked List Truncation for Large Language Model-based Re-Ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 141–151.

[23] Alistair Moffat and Justin Zobel. 1996. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems (TOIS)* 14, 4 (1996), 349–379.

[24] Harrie Oosterhuis. 2021. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[25] Andrew Parry, Sean MacAvaney, and Debasis Ganguly. 2024. Top-Down Partitioning for Efficient List-Wise Ranking. *arXiv preprint arXiv:2405.14589* (2024).

[26] Matthias Petri, J Shane Culpepper, and Alistair Moffat. 2013. Exploring the magic of WAND. In *Proceedings of the 18th Australasian Document Computing Symposium*. 58–65.

[27] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667* (2021).

[28] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724* (2023).

[29] Jiarui Qin, Jiachen Zhu, Bo Chen, Zhirong Liu, Weiwen Liu, Ruiming Tang, Rui Zhang, Yong Yu, and Weinan Zhang. 2022. Rankflow: Joint optimization of multi-stage cascade ranking systems as flows. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 814–824.

[30] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* 13 (2010), 375–397.

[31] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. 2024. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 1504–1518.

[32] Zhen Qin, Rolf Jagerman, Rama Kumar Pasumarthi, Honglei Zhuang, He Zhang, Aijun Bai, Kai Hui, Le Yan, and Xuanhui Wang. 2023. RD-Suite: a benchmark for ranking distillation. *Advances in Neural Information Processing Systems* 36 (2023).

[33] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[34] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 14918–14937.

[35] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2289–2298.

[36] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[37] Nicola Tonellotto, Craig Macdonald, Iadh Ounis, et al. 2018. Efficient query processing for scalable web search. *Foundations and Trends® in Information Retrieval* 12, 4-5 (2018), 319–500.

[38] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 105–114.

[39] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022). https://openreview.net/forum?id=yzkSU5zdwD Survey Certification.

[40] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[41] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 391–398.

[42] Le Yan, Zhen Qin, Honglei Zhuang, Rolf Jagerman, Xuanhui Wang, Michael Bendersky, and Harrie Oosterhuis. 2024. Consolidating Ranking and Relevance Predictions of Large Language Models through Post-Processing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 410–423. doi:10.18653/v1/2024.emnlp-main.25

[43] Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. 2024. The Shift from Models to Compound AI Systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

[44] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.

[45] Yue Zhang, ChengCheng Hu, Yuqi Liu, Hui Fang, and Jimmy Lin. 2021. Learning to rank in the age of muppets: Effectiveness–efficiency tradeoffs in multi-stage ranking. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. 64–73.

[46] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[47] Kai Zheng, Haijun Zhao, Rui Huang, Beichuan Zhang, Na Mou, Yanan Niu, Yang Song, Hongning Wang, and Kun Gai. 2024. Full Stage Learning to Rank: A Unified Framework for Multi-Stage Systems. In *Proceedings of the ACM on Web Conference 2024*. 3621–3631.