Sparse Pairwise Re-ranking with Pre-trained Transformers

Lukas Gienapp[†] Maik Fröbe[‡] Matthias Hagen[‡] Martin Potthast[†]

[†]Leipzig University, Leipzig, Germany [‡]Martin-Luther-Universität Halle-Wittenberg, Halle (Saale), Germany

ABSTRACT

Pairwise re-ranking models predict which of two documents is more relevant to a query and then aggregate a final ranking from such preferences. This is often more effective than pointwise re-ranking models that directly predict a relevance value for each document. However, the high inference overhead of pairwise models limits their practical application: usually, for a set of k to-be-re-ranked documents, preferences for all $k^2 - k$ reasonable comparison pairs (back and forth) are aggregated. We investigate whether the efficiency of pairwise re-ranking can be improved by subsampling from all pairs. In an exploratory study, we evaluate three sampling methods and five preference aggregation methods. The best combination allows for an order of magnitude fewer comparisons at an acceptable loss of effectiveness, while competitive effectiveness is already achieved with about one third of the comparisons.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; **Rank aggregation**; **Retrieval effectiveness**; **Retrieval efficiency**.

KEYWORDS

Pairwise re-ranking; Sampling; Efficiency; Pre-trained transformers

ACM Reference Format:
Lukas Gienapp, Maik Fröbe, Matthias Hagen, and Martin Potthast. 2022. Sparse Pairwise Re-ranking with Pre-trained Transformers. In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '22), July 11–12, 2022, Madrid, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539813.3545140

1 INTRODUCTION

Pre-trained transformers have led to a new era in information retrieval: with a sufficient amount of training data, transformer-based re-rankers are significantly more effective than traditional retrieval models [23]. Two classes of re-rankers are implemented using pre-trained transformers [24]: (1) pointwise re-rankers that predict the relevance of a document d to a query q, and (2) pairwise re-rankers that predict which of two documents d_i, d_j is more relevant to q. To further maximize the effectiveness, the mono-duo design pattern [33] shown in Figure 1 applies both sequentially. Given a query q, a document collection D, and a ranking of D produced by a traditional retrieval model like BM25, the top-k' documents $d_1, \ldots, d_{k'}$ are re-ranked according to their pointwise relevance to q. Then, the top k documents, $k \ll k'$, are pairwise re-ranked in three steps. First, pairs of documents (d_i, d_j) are sampled; $0 < i, j < k$ and $i \neq j$. Second, each pair (d_i, d_j) is passed to a transformer model to predict a probability $p_{ij} = P(d_i > d_j \mid d_i, d_j, q)$ indicating whether the document d_i is more ($p_{ij} > 0.5$) or less ($p_{ij} \leq 0.5$) relevant to q compared to d_j. Third, for each document d_i, a score s_i is aggregated from all the comparison probabilities derived for d_i and these scores are then used to derive the final re-ranking.

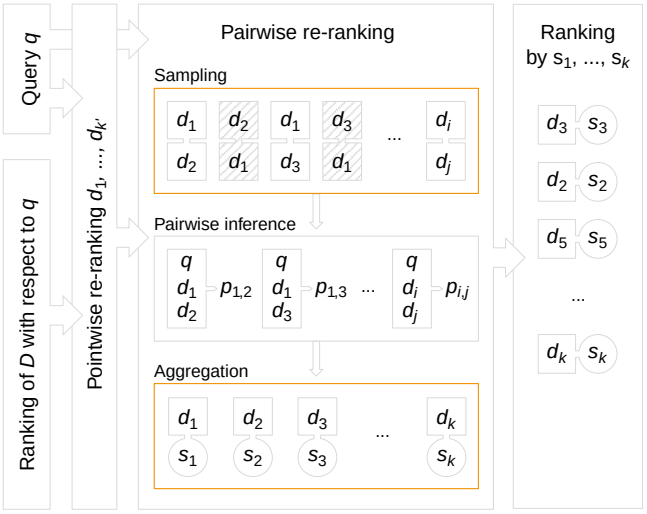


Figure 1: The mono-duo design pattern for re-ranking. Parts investigated in this paper are highlighted in orange.

Empirical evidence suggests that pairwise re-rankers are more effective than pointwise re-rankers since their relevance scores take the relative relevance differences between documents into account, rather than making independent relevance predictions [33]. To maximize the potential effectiveness gains, previous work has relied on exhaustive comparisons of all $k^2 - k$ pairs of the top k to-be-re-ranked documents. Given the high run time overhead of transformer inferences, this quadratic step led to the recommendation that in practice the re-ranking depth should be limited to $k \leq 50$.

However, many of the estimated comparison probabilities might be redundant in the sense that the same preferences could be predicted from already estimated other comparisons. In principle, a lower bound on the run time complexity of $O(k \log k)$ would be achievable with suitable sorting algorithms if the estimated comparison probabilities were "consistent" (i.e., $p_{ij} = 1 - p_{ji}$) and transitive. We investigate for the first time whether the efficiency of pairwise re-rankers can be increased without significant loss of effectiveness by subsampling and thus sparsifying the comparison set.

ICTIR '22, July 11–12, 2022, Madrid, Spain

ACM ISBN 978-1-4503-9412-3/22/07…$15.00
https://doi.org/10.1145/3539813.3545140

The two components of the mono-duo re-ranking pipeline that we study in this paper are highlighted in Figure 1: We introduce a sampling step before the pairwise inference to draw a subset of the $k^2 - k$ possible comparisons, and we revisit the aggregation step since its effectiveness directly depends on the sample it receives (Section 3). To investigate the effect of sparsification on the retrieval effectiveness, we study three sampling methods (global random, exhaustive window, skip window) and five aggregation methods (sorting, summation, regression, greedy, and graph-based) on three datasets (ClueWeb09, ClueWeb12, MS MARCO) using the monoT5 (pointwise) and duoT5 (pairwise) models [33] (Sections 4). Our results suggest that random sampling allows for an order of magnitude fewer comparisons at an acceptable loss of effectiveness, while competitive effectiveness to the all-pairs approach can already be achieved with about one third of the comparisons. (Section 5).

2 RELATED WORK

We briefly give some background on the history of learning to rank retrieval models before detailing the nature of pairwise learning to rank models. Then rank aggregation approaches are reviewed which we employ to aggregate inferred pairwise relevance preferences into a final relevance score. Last, related efforts at making transformer-based learning to rank more efficient are reviewed.

Learning to Rank. Machine learning has meanwhile for decades been successfully applied to improve retrieval effectiveness [16, 24]. Traditionally, feature-based learning to rank models evolved from pointwise to pairwise to listwise approaches [25]. While still successfully applied to this day [34], the recent paradigm shift introduced by pre-trained transformer models shifted the community's focus away from feature-based learning to rank, promising significantly more retrieval effectiveness [24]. History appears to repeat itself as the aforementioned evolution from pointwise approaches like monoBERT [31] and monoT5 [30] to pairwise approaches [24] has been observed as well.

Pairwise Learning to Rank. Pairwise learning to rank approaches predict which document in a pair is probably more relevant to a query and should therefore be ranked higher than the other [25]. Pairwise loss functions consider only the relative order of the two input documents [25]. In feature-based as well as transformer-based learning to rank, pairwise approaches usually outperform pointwise ones, which score documents independently of each other [25]. Yet, as they have to compute a prediction for all possible document pairs, their inference overhead increases quadratically. In an effort to reduce the comparison count, the theoretical properties of feature-based pairwise approaches are extensively studied [9, 22], and some are specifically designed to exhibit desirable characteristics. For example, SortNet [35] uses a learned preference function that is guaranteed to output symmetric preferences, allowing to skip the preference computation for those redundant pairs. Yet, recent pairwise transformer-based models like duoBERT [33] and the more effective duoT5 [33] yield excellent results, but lack the theoretical guarantees of models like SortNet. Additionally, their theoretical analysis is still in its infancy, and previous work found such models to be difficult to interpret [26, 40]. Hence, they fall back to computing preferences for all pairs of documents, to the detriment of their overall efficiency [44], which limits their applicability in user-facing search engines.

Rank Aggregation. Rank aggregation [25] derives from the pairwise relevance preferences between any two documents a relevance score for each individual document that is used to rank them. It has been proven NP-hard to find the optimal solution [10]. Two general approaches to the problem exist: static and dynamic rank aggregation. Static rank aggregation commences after all pairwise comparisons have been finished; dynamic rank aggregation decides during aggregation which documents to compare, where the optimal next comparison identified based on all predecessors [41]. While dynamic aggregation shows an empirical advantage over static aggregation, the latter has a better lower bound for least required comparisons [41]. We focus on static aggregation in this paper to establish a well-grounded baseline.

We identify five paradigms to emerge from common static rank aggregation approaches: (1) Sorting, where comparisons are assumed to be consistent and form a total order, and thus can be ranked by pairwise sorting algorithms such as Kwiksort [2]. (2) Additive aggregation [33], where the rank of a document is indicated by the sum of it's comparison probabilities. Prior to summation, transformations may be applied to the probability scores. (3) Regression-based aggregation [4, 37, 39, 45], where latent scores for documents are learned such that they optimally correspond to a given set of pairwise comparisons. (4) Greedy aggregation [3, 10], where the current best-ranked document is determined iteratively by a heuristic and appended to the final ranking. The next iteration then operates on a reduced document set with the previously chosen one removed. (5) Graph-based aggregation [32, 42], where comparisons are interpreted as directed edges between document nodes and a measure of graph centrality is used to derive score for ranking. We choose one representative from each paradigm to test in this paper. A detailed description of the chosen algorithms is given in Section 3.2.

Efficiency Improvements for Transformer-based Re-Rankers. The high computational cost of re-ranking documents with pre-trained transformers has recently received attention [19]. Even for pointwise approaches, the inference overhead can be prohibitive for practical applications [44]. Two approaches to increasing the efficiency of neural re-rankers can be discerned: (1) increase the efficiency of the ranking model, or (2) reduce the required amount of inferences. Approaches to the former include early-exiting from inference by intermediate between-layer classification in BERT-like models [43], model distillation [17, 18], and improved dense representations [38].

For the latter, Zhang et al. [44] propose to introduce filtering steps in multi-stage re-rank pipelines. They utilize feature-based learning to rank to compute a set of candidate documents that is then re-ranked using a BERT-like neural ranking model, increasing efficiency by a factor of up to 18 times compared to an unfiltered baseline at the same effectiveness. Yet, while document filtering techniques have been proposed for pointwise re-ranking, to our knowledge, filtering approaches for pairwise re-ranking have not been addressed to date.

3 METHODOLOGICAL APPROACH

In this section, we detail the steps of our adapted mono-duo re-ranking pipeline (Figure 1): the sampling methods to obtain a comparison set (Section 3.1), the aggregation methods to combine preferences into a relevance score (Section 3.2), and for completeness, the other steps adopted from the literature: initial retrieval, as well as pointwise and pairwise re-ranking (Section 3.3).

3.1 Sampling

Given the top-k results D_k of the mono-duo paradigm's pointwise re-ranking step for query q, we suggest to sparsify the set of all $k^2 - k$ possible comparisons[1] for the pairwise re-ranking by using a subsampled comparison set $C_q \subseteq D_k \times D_k$. The goal of the sampling is to select a C_q as small as possible without compromising the quality of the final ranking. We distinguish two sampling paradigms: random sampling and structured sampling. The main difference between the two is their determinism. With *structured* sampling, the same pointwise re-ranking always leads to the same comparison set, which is not the case for *random* sampling. We further impose two requirements for sampling: (1) each document should have the same number of comparisons, and (2) each comparison can be sampled at most once. Both requirements ensure compatibility with a wide range of aggregation approaches and minimize sampling-induced bias. To illustrate the differences of the three sampling procedures described below, Figure 2 shows the example comparison matrices for a document set of size $k = 20$ for two sample rates, one per row.

Global random sampling (G-Random). For a random sample, for each document $d_1, \ldots, d_k$, a fraction $r \in (0 \ldots 1]$ of the $(k-1)$ potential comparisons to the other top k documents of the pointwise re-ranking is randomly selected. The size of the sampled comparison set is $\lfloor r \cdot (k^2 - k) \rfloor$. The hypothesis underlying this sampling strategy is to not trust the pointwise ranking at all. This hypothesis is almost certainly false for most topics, yet, this sampling strategy still serves as a good baseline for comparison.

Exhaustive window sampling (E-Window). For structured sampling, a sliding window of size m is moved over the list of documents $d_1, \ldots, d_k$ to be re-ranked, comparing the i-th document to its m successors with index $(j \bmod k) + 1$, where $j \in \{i+1, \ldots, i+m\}$. The window wraps around to the beginning of the ranking, so that documents at its end with index $i > k-m$ are compared to the top-ranked ones. The size of the sampled comparison set is $k \cdot m$. The hypothesis underlying this sampling strategy is that the pointwise ranking is globally ranked in an approximate order of relevance, but local re-ranking is still needed. Under this assumption, we initially simply stopped advancing the window once $i + m \geq k$, effectively sampling an exhaustive comparison of the end of the ranking. However, pilot experiments showed the outlined procedure to outperform our initial intuition, likely due to a more globally uniform sampling and less local bias to affect the aggregation method.

Skip window sampling (S-Window). The E-Window approach is limited to the immediate local neighborhood of a document in the pointwise re-ranking. To incorporate the global neighborhood into the comparison process, we introduce a skip size λ to the window of size m, comparing the i-th document to its m successors with index $(j \bmod k) + 1$, where $j \in \{i + \lambda, i + 2\lambda, \ldots, i + m\lambda\}$. For example, with $\lambda = 3$, each document is compared to every third of its successors. For $\lambda = 1$, this sampling method is equivalent with exhaustive window sampling. The hypothesis underlying this sampling method basically corresponds to that of exhaustive window sampling; the λ-skip merely introduces a deterministic means to increase the coverage of the window without increasing the number of comparisons. When λ is chosen large enough, the whole ranking is covered, which essentially corresponds to the hypothesis of the global random sampling. The downside of this sampling method is its additional hyperparameter.

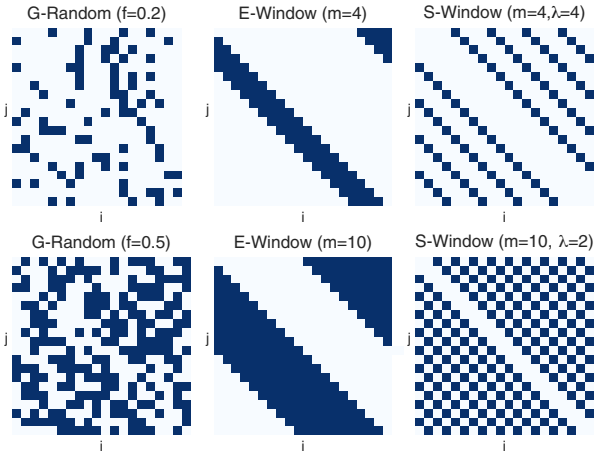


Figure 2: Example comparison matrices of different sampling procedures for 20 documents at different sampling rates. If comparison (d_i, d_j) is sampled, cell i, j of the matrix is colored blue. $|C_q| = 80$ in upper row, $|C_q| = 200$ in lower row.

3.2 Aggregation

Preference scores for all sampled comparisons in C_q are inferred, where p_{ij} is the preference probability for a document pair $(d_i, d_j) \in C_q$ given by the probability measure $P(d_i > d_j \mid d_i, d_j, q)$ induced by the pairwise model. These need to be combined into a singular score values $S_{i \in 1 \ldots k}$ for each document to derive a final output ranking. We test five different aggregation methods, each stemming from a different paradigm of aggregation.

Kwiksort Aggregation. As baseline method for the lower bound of time complexity, and thus the most efficient regarding comparison count ($n \log n$), we choose the *Kwiksort* method [2]. It is an extension of the QuickSort algorithm for pairwise data instead of item-wise sorting: first, a random document d_i is chosen to be the pivot. Then, all documents comparing to be lower-ranked than d_i and all documents comparing to be higher-ranked than d_i are placed in separate subsets. The algorithm is recursively applied to both sets until a final ranking is obtained. This aggregation methods differs from the other four in that it does not rely on a preceding sampling step to achieve efficiency: the reduced comparison count is a feature of the tournament-like aggregation itself.

[1] Usually, a document is not compared to itself.

Additive Aggregation. Pradeep et al. [33] propose four different aggregation techniques based on preference probability summation. They find the symmetric sum of preference probabilities to be the best performing:

$$S_i = \sum_{j \in 1...k} (p_{ij} + (1 - p_{ji})) \tag{1}$$

On sampled comparison data, a shortcoming of this method w.r.t the original setup of Pradeep et al. [33] is that for any pair (d_i, d_j) in the sampled set, the inverse case (d_j, d_i) is not guaranteed to be present. Thus, the symmetric sum defined above is not necessarily complete and zero is added for non-present comparisons.

Bradley-Terry Aggregation. The Bradley-Terry model [4] infers latent scores S_i for documents based on the comparison set using maximum-likelihood estimation. With exponential score functions, the model is reduced to a logistic regression [1] on pairwise data and can be expressed as follows:

$$\mathcal{L}(S, C) = \sum_{(d_i, d_j) \in C} \log \frac{e^{S_i}}{e^{S_i} + e^{S_j}} \tag{2}$$

The unknown latent score set S is then solved for using BFGS optimization to optimally fit the supplied comparisons. The Bradley-Terry model does not take into account comparison weights. Thus, the comparisons are modified to directly encode the preference, i.e. a document pair (d_i, d_j) with score $p_{ij} < 0.5$ is flipped to (d_j, d_i).

Greedy Aggregation. Cohen et al. [10] propose a greedy ordering algorithm that is proven to closely approximate the best total order. It is illustrated in Algorithm 1: it initializes a *potential* vector π, where potential of each document is the preference sum of comparisons where d_i occupies the first position in a comparison tuple, minus the preference sum of comparisons where d_i occupies the second position. In each iteration, the document with the highest potential is removed from the set; the remaining potentials are updated to reflect the documents' removal. The score of each document is set to the number of remaining documents after removal. In sampled setups, the preference set is incomplete. Missing document pairs are assigned a zero score.

Data: Document set D, preference set R
Result: Document scoring S
foreach $d_i \in D$ **do** $\pi(d_i) \leftarrow \sum_{d_j \in D} p_{ij} - \sum_{d_j \in D} p_{ji}$;
while $D \neq \emptyset$ **do**
  $d_i \leftarrow \arg\max_{d_j \in D} \pi(d_j)$;
  $S_i \leftarrow \#D$;
  $D \leftarrow D \setminus \{d_i\}$;
  foreach $d_j \in D$ **do** $\pi(d_j) \leftarrow \pi(d_j) + p_{ij} - p_{ji}$;
end

Algorithm 1: Greedy Sorting [10]

PageRank Aggregation. The comparison set can be interpreted as weighted directed graph $G = (V = D, E = \{(d_i, d_j, p_{ij}) \forall (d_i, d_j) \in C\})$, where documents are nodes and comparisons form directed edges establishing a *greater than* relation, weighted by the predicted comparison probability. The goal of graph-based methods such as *PageRank* [32] is to assign a score to each node indicating its importance to the network. A ranking is then distilled by sorting documents descending by their score. The fundamental principle of PageRank, that nodes with many incoming edges should be ranked higher, and even more so when the incoming relations stem from other high-ranking nodes, translates well to the sorting problem at hand. The score S_i of a node, i.e. document, d_i is expressed as

$$S_i = c \sum_{v \in B_i} \frac{S_i}{N_v} \tag{3}$$

where B_i is the set of nodes comparing to be less than d_i, N_i is the number of outgoing edges for a node d_i, and c is a normalization factor. This definition is recursive and can be extended to weighted edges [28]. Scores are solved for using the power iteration method.

3.3 Ranking Pipeline

Replicating the experimental setup of Pradeep et al. [33] as closely as possible, we first obtain an initial ranking for all collections and topics using BM25 (as implemented in PyTerrier [27]) in its default configuration. The top $k' = 1000$ documents are then re-ranked using the monoT5 ranking model[30] in the pointwise step. Sampling is applied to to the top $k = 50$ documents to identify a set of comparisons to be inferred. For each comparison, the duoT5 model [33] is used to infer a pairwise probability score in the pairwise step. Both transformer models are used in their largest available pretrained version.[2] T5 is chosen over BERT variants, as T5 has been shown to outperform those [24]. The resulting scores are finally aggregated into the output ranking. To avoid repeated inference, all $k^2 - k$ pairwise comparisons are cached once for each topic.

The maximum input length accepted by the transformer models is limited, thus a representative passage has to be chosen for inference. Following the method of Dai and Callan [15], each document is divided into 250-word fixed-length passages. Fixed-length passages have proven more effective than variable-length passages [21]. Passage division is done using the TREC CAsT4 tool[3]. We use the first passage as representative. Pilot experiments confirmed this to be the best-performing approach.

4 EXPERIMENTAL SETUP

This section introduces our evaluation measures, detailing in particular the measures for consistency and transitivity of the aggregated relevance scores, and reviews the data underlying our experiments.

4.1 Evaluation Measures

To evaluate the impact of sparsifying pairwise re-ranking on ranking effectiveness, we use nDCG@k [20] with $k = 10$, following similar experiments using the mono/duoT5 models in related work [33].

In addition to retrieval effectiveness, we investigate two properties of the sampled comparison set which characterize the quality of the pairwise relevance predictions. They are motivated by the two kinds of judgment inconsistencies the model can make during inference, assuming a latent total order of documents: it can contradict itself either (1) for a pair of documents, predicting different results depending on the order in which documents are fed into

[2] monoT5: https://huggingface.co/castorini/monot5-3b-msmarco
duoT5: https://huggingface.co/castorini/duot5-3b-msmarco
[3] https://github.com/grill-lab/trec-cast-tools

the transformer—one would expect $p_{ij} = p_{ji}$; and (2) for a triple of documents, predicting a non-transitive output.

The *consistency rate* of a comparison set is the ratio of the number of comparison pairs (d_i, d_j) for which $p_{ij} \geq 0.5$ and $p_{ji} < 0.5$ to the overall number of comparison pairs. While this indicates how many comparisons are consistent with regard to their predicted *direction*, the *magnitude* of the difference between p_{ij} and p_{ji} is also interesting. Thus, we define consistency in terms of the ε-complementarity rate of the comparison set C_q for a given query q as specialized case of antisymmetry, the ratio of complementary comparisons to all comparisons, with respect to a margin of error ε:

$$consistency(\varepsilon, q) \;=\; \frac{|\{(d_i, d_j) \in C_q \mid \varepsilon > |p_{ij} - (1 - p_{ji})|\}|}{|C_q|}, \quad (4)$$

The *transitivity rate* of the comparison graph is the ratio of the number of transitive triangles T (three comparisons that connect three vertices, but do not form a loop) to the number of triads U (two comparisons with one shared document):

$$transitivity(\varepsilon, q) \;=\; 3\frac{|T|}{|U|}, \quad (5)$$

where

$$T \;=\; \{(d_h, d_i, d_j) \mid (d_h, d_i), (d_i, d_j), (d_h, d_j) \in C_q \text{ and } p_{hi} > 0.5,\ p_{ij} > 0.5,\ \text{and}\ p_{hj} > 0.5\}$$

$$U \;=\; \{(d_h, d_i, d_j) \mid (d_h, d_i, p_{hi}), (d_i, d_j, p_{ij}) \in C\}$$

If both the transitivity and the consistency in terms of the ε-complementarity rate of a given set of comparisons C_q approach 1 for a small ε, a total order between documents is implied, which increases sampling robustness.

4.2 Data

ClueWeb09. The ClueWeb09 (CW09) corpus[4] consists of 1 billion documents crawled between January and February 2009. It was used for the ad-hoc search tasks of the TREC Web tracks 2009—2012 [5–8], where 70,575 graded relevance judgments were collected on a 4-point scale for 200 topics (avg. 356 judgments per topic).

ClueWeb12. The Clueweb12 (CW12)[5] corpus consists of 730 million documents crawled between April and May 2012. It was used for the ad-hoc search tasks of the TREC Web tracks 2013 [11] and 2014 [12], where 28,116 graded relevance judgments were collected on a 4-point scale for 100 topics (avg. 281 judgments per topic).

MS MARCO Passage Corpus. The MS MARCO passage corpus [29] consists of 8.8 million passages extracted from Bing search engine results. It was used for the passage ranking task of the TREC Deep Learning tracks 2019 [14] and 2020 [13], where 20,646 graded relevance judgments were collected on a 4-point scale for 97 topics (avg. 213 judgments per topic). With this corpus, we replicate the experimental setup of Pradeep et al. [33].

Remark. Only the judged documents of each dataset are subject to our re-ranking experiments. This guarantees fully judged runs and consistency. Evaluation scores calculated excluding unjudged documents correlate well with evaluations including them [36].

[4]http://lemurproject.org/clueweb09.php/
[5]https://www.lemurproject.org/clueweb12/

5 EVALUATION RESULTS

We conduct two experiments to evaluate the suitability of sampling and aggregation methods for efficient pairwise re-ranking. The first experiment (Section 5.1) explores the properties of the comparison sets inferred for each of the three collections. This supplies context to the ranking effectiveness evaluation in the second experiment (Section 5.2), in which we simulate rankings for different combinations of samplers and aggregators on each experiment collection.

5.1 Evaluation of Pairwise Prediction Properties

For each collection, relevance preference probabilities for the full pairwise comparison set for the $k = 50$ top-ranked documents according to the pointwise re-ranking are inferred and cached separately for each topic on each collection. Across all topics per collection, three key statistics are computed: In Figure 3, Plot (a) shows the distribution of predicted scores; Table (b) details the rate of consistent pairs aggregated over topics; Table (c) gives the transitivity rate aggregated over topics; and Plot (d) shows the cumulative ratio of complementary comparisons per ϵ-complementarity rate.

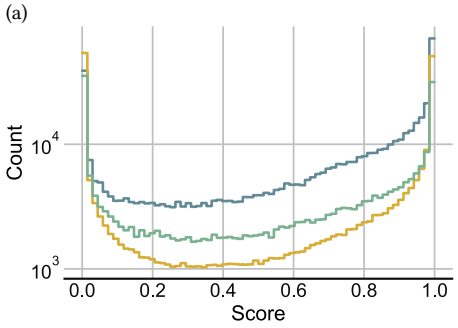
The predicted scores are highly skewed towards the extremes of the scale: for the majority of document pairs, p is near either one or zero. This effect is more strongly apparent for the MS MARCO collection (on which the model was trained) than for the ClueWeb collections. Further, the score distributions are not symmetric, but have a slight skew towards 1.0 equally present for all three collections. This already suggests that complementary predictions are not the norm, as otherwise (since the comparison set is complete), the distribution would have to be fully symmetric.

Indeed, on average, only between half (MS MARCO) and a third of the comparisons (CW09) are consistent in their direction. Some variation across topics exists, yet the majority of topic-wise comparison sets contains a very high degree of inconsistency. Further, the graph plotting the cumulative ratio of pairs per ϵ-complementarity value confirms that the pairwise model is not invariant to the order in which passages are fed into it. Only for a large value of $\epsilon = 0.4$ all scores for all collections are complementary, and more than half the comparisons require an ϵ-value between 0.3 (MS MARCO) and 0.2 (CW09). The ClueWeb collections reaching a lower ϵ-value could be due to their differing score distributions compared to MS MARCO: as their scores tend to be less skewed towards the extremes, the potential mismatch between a pair of scores is lower as well. Yet, for all collections, almost no comparison pairs reach a score disparity of $\epsilon < 0.1$. This suggests that sampling by only inferring one direction of each comparison pair, which would already cut the required amount of inferences in half, is not feasible. Transitivity rates are consistently only between 0.7 and 0.8 across all collections with very little variation per topic, undermining the prospect of sorting aggregation, as it depends on a total order.

5.2 Evaluation of Ranking Effectiveness

We simulate runs on sparsified comparison sets at sample rates ranging from 0.05 to 0.95 in 0.05 increments. All combinations of samplers and aggregators are tested for each sample rate and collection. Each simulation is repeated ten times to account for random variation in both the sampling and the aggregation step. In addition,

(a)



(b)

Collection	Mean	Std.	Min	Max
CW09	0.312	0.136	0.054	0.637
CW12	0.383	0.115	0.120	0.643
MS MARCO	0.498	0.126	0.136	0.731

(c)

Collection	Mean	Std.	Min	Max
CW09	0.783	0.079	0.616	0.954
CW12	0.742	0.064	0.611	0.898
MS MARCO	0.693	0.073	0.580	0.901

(d)

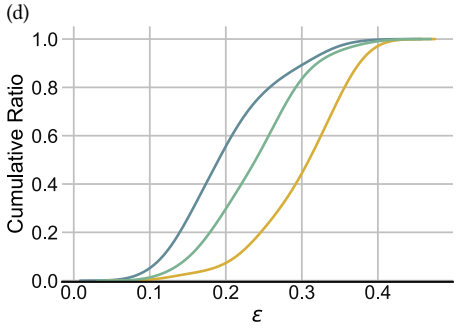


Figure 3: (a) Distribution of predicted scores per collection (log-scaled). (b) Antisymmetry rate over all topics, per collection. (c) Transitivity score over all topics, per collection. (d) Cumulative ratio of complementary comparison pairs dependent on ϵ. Collections are color-coded as CW09, CW12, and MS MARCO.

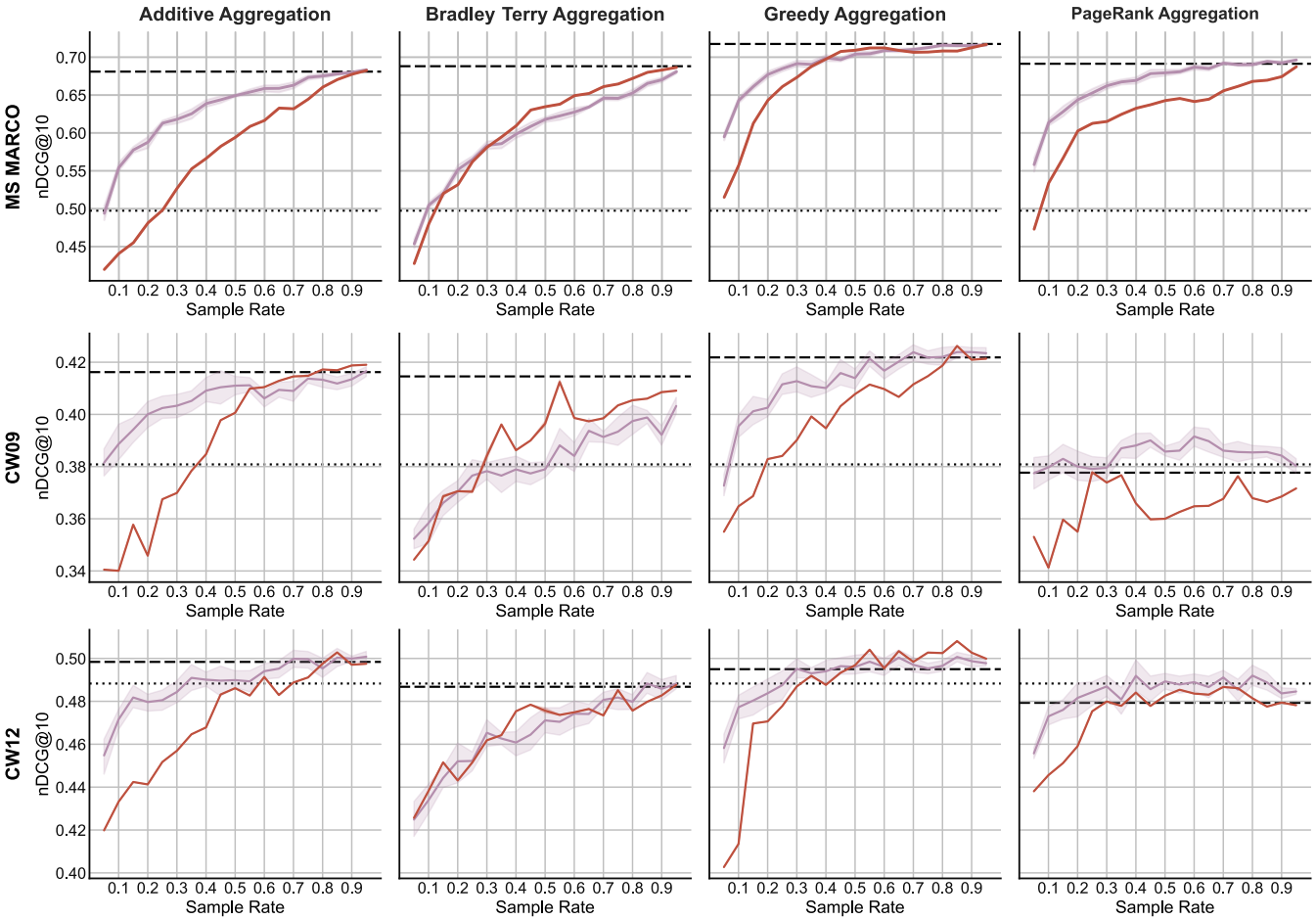


Figure 4: nDCG@10 for each aggregator at different sampling rates for G-Random and E-Window sampling per collection. Dotted line is pointwise re-ranking, dashed line is unsampled effectiveness.

baseline runs for each aggregator are established on the unsampled comparisons. In total, 4950 runs are simulated. Figure 4 shows the nDCG@10 performance of each aggregation method except Kwiksort at different sampling rates for G-Random and E-Window sampling. The reference lines indicate the baseline effectiveness of the pointwise re-ranking (dotted) and the effectiveness or pairwise re-ranking on the unsampled complete comparison set (dashed).

Effectiveness of Kwiksort. Judging from the comparison properties investigated in Section 5.1, the Kwiksort method is expected to perform sub-par, as it would require a total order to be present in the comparisons. Indeed, the Kwiksort method scores low, with nDCG@10 scores ranging from 0.34 (CW09), 0.39 (CW12), to 0.42 (MS MARCO). This is a net decrease from the pointwise re-ranking, which scores at 0.38 (CW09), 0.49 (CW12) and 0.50 (MS MARCO). Kwiksort is by design only using a small subset of the comparisons, and therefore operates without a dedicated sampling step. Different methods of choosing a pivot element were tested, with no increase in effectiveness. Using Kwiksort leads to diminished effectiveness in all cases, and thus, lacking robustness to deal with inconsistent comparisons, its use as an aggregation method is discouraged as per this setting.

Effectiveness under no sampling. On the full comparison set, greedy aggregation performs best across two out of three collections, beaten only by additive aggregation and only on CW12. On MS MARCO, the other three aggregation methods score approximately equal. On the ClueWeb collections, only additive aggregation continues to yield competitive results. Both PageRank and Bradley-Terry score below the pointwise re-ranking. For greedy and additive aggregation, the overall gain in effectiveness is diminshed on the ClueWeb collections, too, compared to MS MARCO: on CW09, a delta of ca. 0.04, and on CW12, a delta of ca. 0.01 is observable, while on MS MARCO, the delta exceeds 0.2 nDCG@10. This is expected, since (1) the ranking model was trained on MS MARCO, and (2) the TREC Web track qrels used to evaluate effectiveness on the ClueWeb collections assess entire documents, while we only rank one passage per document due to input length limitations.

Effectiveness under parameter-free sampling. On sparse, sampled comparisons, four trends are apparent across all collections. First, E-Window sampling performs worse than G-Random sampling in nearly all cases, especially at smaller sampling rates. This effect is particularly noticeable for additive aggregation, to no surprise: the locally bounded comparisons of E-Window sampling are likely to yield less extreme comparison probabilities. Thus, the additive sum when aggregating is not impacted as much by each single comparison, decreasing the overall separability of documents in very sparse setups. Also, inconsistencies in pairwise judgments are more likely for documents within local proximity of each other in the pointwise re-ranking. Overall, this indicates that the global context of documents (which is better represented by G-Random sampling) is important to obtain accurate aggregated scores.

Second, greedy aggregation performs best out of all aggregation methods. This is observable for both for G-Random as well as E-Window sampling. It is the only aggregation method that matches its own unsampled performance on sparsified comparison sets, and also the aggregation method that is most indifferent towards the choice of sampling scheme. Third, the effectiveness degradation is not linear with respect to the sample rate, but drops off sharply below 15-20% of comparisons. This suggests a lower bound under which no meaningful ranking can be extracted from the pairwise comparisons, likely because a minimum amount of information is needed to overcome the inconsistencies in the judgments made by the pairwise model. Finally, Bradley-Terry aggregation performs worst, which is likely due to it taking only the direction, not the magnitude of each comparison into account. This suggests that optimizing future pairwise models not only towards consistency in directions, but also complementarity in probabilities is warranted.

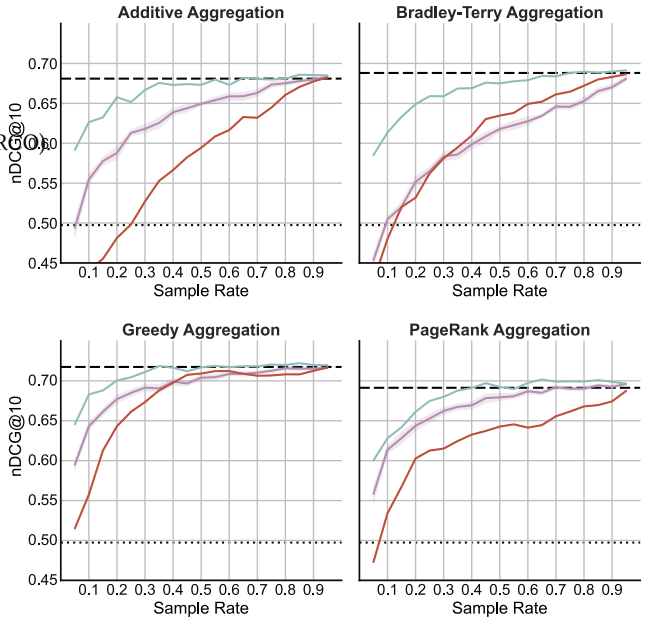


Figure 5: nDCG@10 for each aggregator at different sampling rates for S-Window, G-Random, and E-Window sampling on MS MARCO. Dotted line is pointwise prior ranking, dashed line is unsampled effectiveness.

Effectiveness under parametric sampling. To find an optimal value for λ in S-Window sampling, a grid search is carried out over values between 2 and 15, separately for all sample rates previously evaluated. Five-fold cross validation is used to determine the best choice. This experiment is only carried out on the MS MARCO collection, as overall effectiveness gains on CW09/CW12 were too small to achieve meaningful separation between setups. Figure 5 shows the nDCG@10 effectiveness of the run with optimal λ-value for each sample rate. Runs for parameter-free G-Random and S-Window sampling are also shown for reference.

S-Window sampling outperforms both other sampling methods by a margin for each of the aggregation methods at all sampling rates. It proves to be the best choice across the whole range. In conjunction with the best performing aggregation methods, greedy aggregation, S-Window sampling allows for stable performance down to 30% of comparisons. Even when using an inference expenditure lower by an order magnitude ($\leq$ 10%), a competitive ranking is achieved, scoring only 0.04 less than the unsampled run.

The value for λ is not correlated with the window size ($\bar{\rho}$ = 0.04), with a value between 7 and 10 found to be optimal in most cases. Given the favorable performance of G-Random sampling over E-Window sampling this suggests that the global context is important when sampling comparisons. The high optimal λ-values for S-Window sampling corroborate this, as for small window sizes, they expand the window to cover more global context. For large windows, λ is not as important as they (1) already cover a large

Table 1: nDCG@10 for full comparison set and lowest sample rate that has non-significantly different ranking per sampling method and aggregator. Delta to full nDCG@10 in brackets. Bonferroni-correction applied for all tests per row.

Aggregator	nDCG@10	Lowest Attainable Sampling Rate		
	Unsampled	S-Window	G-Random	E-Window
Additive	0.691	0.35 (-0.014)	0.85 (-0.019)	0.95 (-0.004)
Bradley-Terry	0.691	0.50 (-0.012)	1.00 (-0.000)	0.90 (-0.008)
Greedy	0.707	0.30 (-0.013)	0.85 (-0.006)	0.50 (-0.010)
PageRank	0.695	0.30 (-0.016)	0.65 (-0.012)	0.95 (-0.004)

amount of the comparison set, and (2) windows are cyclical, such that any λ value increases the coverage of the window.

Statistical Significance. Table 1 lists the minimum attainable sampling rate for which the resulting nDCG@10 is not significantly different to the unsampled baseline for each aggregator and sampling method. Additionally, the nDCG delta relative to the baseline per aggregator is given. We use a conservative testing strategy with paired t-tests and Bonferroni correction to account for multiple testing at an overall α-level of 0.05. Per aggregator, each of the 19 sampled runs is tested against the unsampled ones. Bonferroni correction is applied to each of these test series per aggregator individually. For G-Random sampling, where performance can differ between repetitions, the (mean) worst performing run at each sampling step is taken as representative to reach the highest overall confidence. Significance tests were only carried out on the MS MARCO collection, as overall effectiveness gains on CW09/CW12 were too small to achieve meaningful separation between runs.

Among the sampling strategies, S-Window sampling outperforms both other methods by far, achieving the same ranking at less than a third of the original comparison amount. G-Random sampling reaches a minimum sampling rate of 0.85, and an E-Window a rate of 0.50. However, on average, G-Random performs better, undercutting E-Window in three out of four cases. E-Window sampling only fares well for greedy aggregation, and provides virtually no benefit for other aggregation methods. Among the aggregation strategies, Bradley-Terry performs worst on average, followed by additive aggregation. Greedy and PageRank aggregation perform nearly on-par with regard to the sample rate, as they both reach 0.30 with S-Window sampling. However, greedy aggregation produces the higher absolute nDCG@10 score, and a lower delta to the baseline than PageRank.

Overall, the choice of the sampling strategy seems to have more impact on results than the choice of the aggregation method. The best result is achieved by greedy aggregation with S-Window sampling: the resulting run is on-par with the unsampled case at only one third of the inference expenditure.

6 CONCLUSION

In this paper, we propose to sparsify the comparison set of pairwise re-ranking approaches to overcome the issue of quadratic inference complexity. Instead of computing the full $k^2 - k$ comparisons for k to-be-re-ranked documents as was done before, we introduce a sampling step to reduce the number of inferences and compare combinations of three different sampling approaches with five different comparison aggregation methods.

All newly proposed aggregation approaches (Greedy, Bradley-Terry, PageRank) exceed the performance of the previously established one (Additive) on the full comparison set, with greedy aggregation performing best by a margin. When subject to one of the three proposed sampling methods, even on a highly sparsified comparison set, accurate rankings can be inferred, at much lower cost. In the best case (skip window sampling with greedy aggregation), the ranking effectiveness of the full comparison set can be matched at as low as a third of inference expenditure. Even higher degrees of sparsification are attainable, with acceptable rankings still being produced at comparison counts lower by an order of magnitude compared to the full set. While globally random sampling outperforms the locally bounded exhaustive window sampling, introducing a skip parameter increased the effectiveness of the structured approach tremendously and yields the best overall result after optimization. This suggests that the pointwise re-ranking often leaves room for improvement and that the global context in the comparison set is needed to accurately re-rank in the pairwise step. Further, the skip window sampling method is deterministic, a feature notably lacking for random sampling.

Overall, this paper shows that sparsification in pairwise retrieval is not only possible, but also highly effective. It increases the efficiency of the method, rendering it more practicable in real-world settings. The proposed aggregation and sampling schemes establish a strong baseline, and also open up new areas of future research. As the tested sampling paradigms of random vs. structured, and locally vs. globally bounded, respectively, suggest that deterministic sampling outperforms the random paradigm, yet global context is important, formulating domain-specific sampling approaches seems promising for further improvements. Similarly, dynamic sampling schemes that iteratively sample a comparison set in a tik-tok pattern, deciding the optimal comparisons to sample next based on previously inferred scores have successfully been applied in other contexts and could transfer well to this task. Similar to the pointwise filtering approach by Zhang et al. [44], pairwise feature-based filtering could also be used to judge whether a comparison is worthwhile to compute using neural models. Also, feature-based approaches could be employed to create pseudo-scores that can be used as proxy for neural scoring.

Sparsification could also be used not to increase the efficiency, but instead the depth of pairwise retrieval: instead of optimizing the inference budget for a fixed depth, one could set a fixed inference budget and in turn increase the amount of considered documents, promising higher effectiveness especially for recall-oriented retrieval tasks. Yet, this paper also reveals that consistency is lacking in current pairwise models. Increasing both the consistency (in direction) and complementarity (in magnitude) of predicted comparisons is warranted, and could make even greater degrees of sparsification possible. Also, highly sparsified comparisons could be used to probe whether pairwise reranking yields improvements at all. As illustrated by only minimal improvements on Clueweb data, the pairwise step could first sample a small subset of comparisons and if no improvement is detectable, skip the costly pairwise re-ranking step altogether.

REFERENCES

[1] Alan Agresti and Maria Kateri. 2011. Categorical Data Analysis. In *International Encyclopedia of Statistical Science*. Springer, 206–208.

[2] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating Onconsistent Information: Ranking and Clustering. *J. ACM* 55, 5 (2008), 23:1–23:27.

[3] Juan A. Aledo, José A. Gámez, and Alejandro Rosete. 2021. A Highly Scalable Algorithm for Weak Rankings Aggregation. *Inf. Sci.* 570 (2021), 144–171.

[4] Ralph Allan Bradley and Milton E Terry. 1952. Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[5] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2009. Overview of the TREC 2009 Web Track. In *Proc. of TREC 2009*.

[6] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon V. Cormack. 2010. Overview of the TREC 2010 Web Track. In *Proc. of TREC 2010*.

[7] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. 2011. Overview of the TREC 2011 Web Track. In *TREC'11*.

[8] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. 2012. Overview of the TREC 2012 Web Track. In *Proc. of TREC 2012*.

[9] Stéphan Clémençon, Gábor Lugosi, and Nicolas Vayatis. 2008. Ranking and Empirical Minimization of U-Statistics. *The Annals of Statistics* 36, 2 (2008), 844–874.

[10] William W. Cohen, Robert E. Schapire, and Yoram Singer. 1999. Learning to Order Things. *J. Artif. Intell. Res.* 10 (1999), 243–270.

[11] Kevyn Collins-Thompson, Paul Bennett, Fernando Diaz, Charles L. A. Clarke, and Ellen M. Voorhees. 2013. Overview of the TREC 2013 Web Track. In *Proc. of TREC 2013*.

[12] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M. Voorhees. 2014. Overview of the TREC 2014 Web Track. In *Proc. of TREC'14*.

[13] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 Deep Learning Track. *CoRR* abs/2102.07662 (2021).

[14] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. *CoRR* abs/2003.07820 (2020).

[15] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *Proc. of WWW 2020*. ACM / IW3C2, 1897–1907.

[16] Norbert Fuhr. 1989. Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle. *ACM Trans. Inf. Syst.* 7, 3 (1989), 183–204.

[17] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding BERT Rankers Under Distillation. In *Proc. of ICTIR 2020*. ACM, 149–152.

[18] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *CoRR* abs/2010.02666 (2020). arXiv:2010.02666

[19] Sebastian Hofstätter and Allan Hanbury. 2019. Let's Measure Run Time! Extending the IR Replicability Infrastructure to Include Performance Aspects. In *Proc. of OSIRRC@SIGIR 2019 (CEUR, Vol. 2409)*. CEUR-WS.org, 12–16.

[20] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[21] Marcin Kaszkiel and Justin Zobel. 1997. Passage Retrieval Revisited. In *Proc. of SIGIR 1997*. ACM, 178–185.

[22] Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Tie-Yan Liu. 2012. Statistical Consistency of Ranking Methods in A Rank-Differentiable Probability Space. In *Proc. of NeurIPS 2012*. 1241–1249.

[23] Jimmy Lin. 2019. The Neural Hype, Justified! A Recantation. *SIGIR Forum* 53, 2 (2019), 88–93. https://doi.org/10.1145/3458553.3458563

[24] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. *Synthesis Lectures on Human Language Technologies* 14, 4 (2021), 1–325.

[25] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer, Berlin Heidelberg.

[26] Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. 2020. ABNIRML: Analyzing the Behavior of Neural IR Models. *CoRR* abs/2011.00696 (2020).

[27] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in Information Retrieval using PyTerrier. In *Proc. of ICTIR 2020*. ACM, 161–168.

[28] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proc. of EMNLP 2004*. ACL, 404–411.

[29] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR* abs/1611.09268 (2016).

[30] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Proc. of Findings EMNLP 2020*. ACL, New York, 708–718.

[31] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *CoRR* abs/1910.14424 (2019), 1–13.

[32] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report.

[33] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *CoRR* abs/2101.05667 (2021), 1–23.

[34] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *Proc. ICLR 2021*.

[35] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. 2011. SortNet: Learning to Rank by a Neural Preference Function. *IEEE Trans. Neural Networks* 22, 9 (2011), 1368–1380.

[36] Tetsuya Sakai. 2007. Alternatives to Bpref. In *Proc. of SIGIR 2007*. ACM, 71–78.

[37] Hal Stern. 1992. Are All Linear Paired Comparison Models Empirically Equivalent? *Mathematical Social Sciences* 23, 1 (1992), 103–117.

[38] Hongyin Tang, Xingwu Sun, Beihong Jin, Jingang Wang, Fuzheng Zhang, and Wei Wu. 2021. Improving Document Representations by Generating Pseudo Query Embeddings for Dense Retrieval. In *Proc. of ACL 2021*. ACL, 5054–5064.

[39] Louis Thurstone. 1927. The Method of Paired Comparisons for Social Values. *The Journal of Abnormal and Social Psychology* 21, 4 (1927), 384.

[40] Michael Völske, Alexander Bondarenko, Maik Fröbe, Benno Stein, Jaspreet Singh, Matthias Hagen, and Avishek Anand. 2021. Towards Axiomatic Explanations for Neural Ranking Models. In *Proc. of ICTIR 2021*. ACM, 13–22.

[41] Yue Wu, Tao Jin, Hao Lou, Pan Xu, Farzad Farnoud, and Quanquan Gu. 2021. Adaptive Sampling for Heterogeneous Rank Aggregation from Noisy Pairwise Comparisons. *CoRR* abs/2110.04136 (2021).

[42] Yu Xiao, Hongzhong Deng, Xin Lu, and Jun Wu. 2021. Graph-Based Rank Aggregation Method for High-Dimensional and Partial Rankings. *J. Oper. Res. Soc.* 72, 1 (2021), 227–236.

[43] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early Exiting BERT for Efficient Document Ranking. In *Proc. of SustaiNLP 2020*. ACL, 83–88.

[44] 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 *Proc. of SustaiNLP 2021*. ACL, 64–73.

[45] Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2008. Learning To Rank With Ties. In *Proc. of SIGIR 2008*. ACM, New York, NY, USA, 275–282.