Sparse Pairwise Re-ranking with Pre-trained Transformers

Lukas Gienapp
Leipzig University

Maik Fröbe
Martin-Luther-Universität Halle-Wittenberg

Matthias Hagen
Martin-Luther-Universität Halle-Wittenberg

Martin Potthast
Leipzig University

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 documents to be re-ranked, preferences for all k^2-k comparison pairs excluding self-comparisons are aggregated. We investigate whether the efficiency of pairwise re-ranking can be improved by sampling 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 retrieval 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 ushered in a new era in information retrieval: with a sufficient amount of training data, transformer-based re-ranking models can significantly outperform traditional retrieval models [24]. Two classes of re-rankers are implemented using pre-trained transformers [25]: (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 re-ranking effectiveness, the mono-duo design pattern [34] shown in Figure 1 applies both sequentially.

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

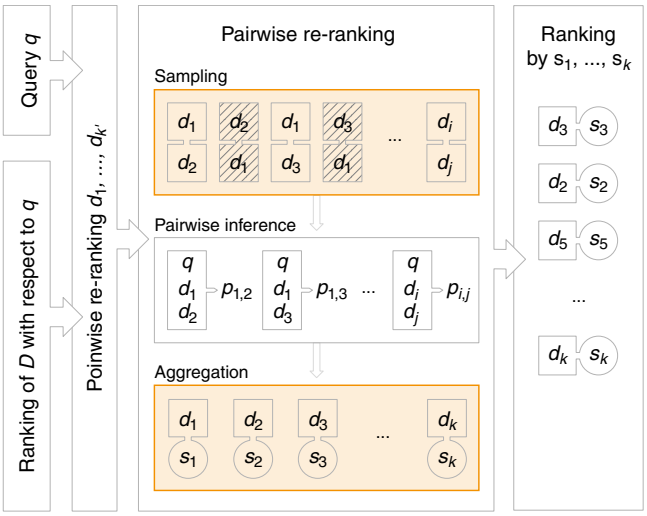


Figure 1: The mono-duo design pattern for re-ranking. Parts investigated in this paper are highlighted in orange. Comparisons omitted by sampling are striped.

Given a query q, a document set D, and a ranking of D produced by a traditional retrieval model like BM25, the top-k' documents $D_{k'} = \{d_1, \ldots, d_{k'}\} \subset D$ are re-ranked according to their pointwise relevance to q. Then, the top-k documents $D_k \subset D_{k'}$, $k \ll k'$, are re-ranked based on pairwise comparisons in three steps. First, pairs of documents (d_i, d_j) are sampled, where $i, j \in [1, k]$ and $i \neq j$. Second, each pair (d_i, d_j) is passed to a transformer model to predict a probability p_{ij} indicating whether the document d_i ($p_{ij} \geq 0.5$) or the document d_j ($p_{ij} < 0.5$) is more relevant to q. Third, for each document d_i, a relevance score s_i is aggregated from all probabilities of comparisons including d_i, and these scores are then used to derive the final pairwise re-ranking.

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 [34]. To maximize the potential effectiveness gains, previous work has relied on exhaustive comparisons of all $k^2 - k$ pairs of the top-k documents D_k to be re-ranked. Given the high run time overhead of transformer inferences, this quadratic step led to the recommendation that the re-ranking depth should be limited to $k \leq 50$.

However, many of the estimated comparison probabilities may be redundant in that they can be predicted from those of other comparisons. A theoretical lower bound on the run time complexity is $O(k \log k)$ using a suitable sorting algorithm if the estimated

comparisons were "consistent" (i.e., $p_{ij} = 1 - p_{ji}$) and transitive. We investigate for the first time if the efficiency of pairwise re-rankers can be increased without a significant loss of effectiveness by sampling from and thus sparsifying the comparison set.

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 kind of 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 pointwise monoT5 and the pairwise duoT5 models [34]. Our results show that skip-window sampling with greedy aggregation 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 only one third of the comparisons (Section 5). All code and data underlying our experiments are publicly available.[1]

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 and reviewing rank aggregation approaches, which we employ to aggregate pairwise preferences into a final ranking. Finally, we describe some related efforts at making transformer-based learning to rank more efficient.

Learning to Rank. Since decades, machine learning has been applied to improve retrieval effectiveness [17, 25]. Traditional feature-based learning-to-rank models evolved from pointwise over pairwise to listwise approaches [26]. While feature-based models are still successful [35], the recent promising retrieval effectiveness results of pre-trained transformer models [25] has shifted the community's focus away from feature-based learning to rank. But history appears to repeat itself as the aforementioned evolution from pointwise approaches like monoBERT [32] and monoT5 [31] to pairwise approaches [25] can 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 be ranked higher [26]. In feature-based as well as transformer-based learning to rank, pairwise approaches usually outperform pointwise ones that score documents independently of each other [26]. Yet, when the inference step of a pairwise model compares all pairs of documents, the run time requirement is quadratic in the number of documents to be ranked. In an effort to reduce the comparison count, extensive studies of theoretical properties of feature-based pairwise approaches [9, 23] have led to suggestions for better run time characteristics. For example, SortNet [36] uses a learned preference function that is guaranteed to output symmetric preferences, allowing to skip half of the comparisons. Yet, recent pairwise transformer-based models like duoBERT [34] and the more effective duoT5 [34] lack the desirable symmetry property of models like SortNet. Additionally, the theoretical analysis of duoBERT and duoT5 is still in its infancy; previous work even found such models difficult to be interpreted [27, 41]. The effectiveness of duoBERT or duoT5 relies on computing preferences for *all* pairs of documents, at the expense of their efficiency [45] limiting their applicability in search scenarios with run time constraints.

Rank Aggregation. Rank aggregation [26] uses pairwise relevance preference probabilities to derive a ranking—often by computing a score for each individual document. Finding an optimal aggregation with arbitrarily-sized inputs is an NP-hard problem [10], but many approaches are known to work well in practice. While dynamic aggregation methods decide which documents to compare next based on all previous comparisons, static aggregation methods assume that the required pairwise comparisons are conducted before the actual aggregation starts [42].

In our study, we employ the following five aggregation methods (details in Section 3.2): (1) Sorting via the KwikSort method [2], which assumes that the comparisons are consistent and form a total order, (2) additive aggregation [34], where the rank of a document is indicated by the sum of the document's comparison probabilities (potentially transforming the probabilities before summation), (3) regression-based aggregation [4, 38, 40, 46], where latent scores for documents are learned so that they optimally correspond to a given set of pairwise comparisons, (4) greedy aggregation [3, 10], in which a heuristic iteratively selects and removes the best document from a given set and then proceeds with the rest, and (5) graph-based aggregation [43], where comparisons are interpreted as directed edges between document nodes and a measure of graph centrality is used to derive a ranking score.

Efficiency Improvements for Transformer-based Re-Rankers. The high computational cost of re-ranking documents with pre-trained transformers has recently received attention [20]. Even for pointwise approaches, the inference overhead can be prohibitive for practical applications [45]. There are two ideas to improve the efficiency of neural re-rankers: (1) improving the efficiency of the ranking model, and (2) reducing the required number of inferences.

Approaches to the former include early-exiting from inference by intermediate between-layer classification in BERT-like models [44], model distillation [18, 19], or improved dense representations [39]. For the latter, Zhang et al. [45] propose to introduce filtering steps in multi-stage re-ranking 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 model, increasing efficiency by a factor of up to 18 compared to an unfiltered baseline at the same effectiveness. But while document filtering has been studied for pointwise re-ranking, to our knowledge, filtering approaches for pairwise re-ranking have not been addressed to date.

3 SPARSIFIED PAIRWISE RE-RANKING

In this section, we describe the steps we adapted in the mono-duo re-ranking pipeline (Figure 1): sampling methods to select the to-be-compared document pairs (three methods, Section 3.1), and aggregation methods that derive a ranking from the ranking preferences (five methods, Section 3.2). For completeness, we also briefly detail the steps adopted from the literature: initial retrieval, as well as pointwise and pairwise re-ranking (Section 3.3).

[1] https://github.com/webis-de/ICTIR-22

3.1 Sampling

Based on the top-k results D_k of the pointwise re-ranking step of the mono-duo paradigm, we propose to sparsify the set C_{all} of all $k^2 - k$ comparisons (no self-comparisons) and to use a sampled comparison set $C \subset C_{all}$ as input for the pairwise re-ranking step. The goal is to minimize the size of C and thus the effort of pairwise re-ranking without compromising the quality of the final ranking.

We distinguish random from structured sampling, with the main difference being their (non-)determinism. Independent random samples from a given C_{all} very likely contain different comparisons, but structured samples always choose the same comparisons. To be compatible with a variety of aggregation methods, a sampling must meet two requirements: (1) each document is part of at least one comparison, (2) each comparison is sampled at most once. Figure 2 illustrates the three sampling methods introduced below for a document set D_k of size $k = 20$ at two sampling rates, one per line.

Global random sampling (G-Random). For each of the top-k documents $d \in D_k$ from the pointwise ranking, a fraction $r \in (0...1]$ of the remaining $k - 1$ documents is randomly selected for the comparison set C that then has the size $|C| = \lfloor r \cdot (k^2 - k) \rfloor$.

Neighborhood window sampling (N-Window). A sliding window of size $m \leq k - 1$ is moved over the top-k documents D_k from the pointwise ranking. For document $d_i \in D_k$, its m direct successors in the ranking are sampled for comparison. But since the last m documents in the ranking of D_k have less than m successors, we let the window "wrap around" to the top-ranked documents. For document d_i, the m sampled comparisons $(d_i, d_j) \in C$ thus fulfill $j = 1 + (a \bmod k)$ for $a \in \{i, \ldots, i + m - 1\}$. The size of the sampled comparison set C is $k \cdot m$ and each document is the first entry of m comparisons and the second entry of another m comparisons.

An assumption underlying the neighborhood sampling is that the "global" pointwise ranking is sensible but that "local" re-ranking leads to an improved effectiveness. If true, however, it would be plausible to stop the window for $i > k - m$ rather than to wrap it around. However, pilot experiments have shown that this leads to poorer effectiveness, possibly because fewer comparisons are sampled at both the beginning and the end of a top-k ranking.

Skip-window sampling (S-Window). N-Window samples from the "local" neighborhood in a ranking. To enable more "global" comparisons, we introduce a skip size $\lambda \in \mathbb{N}^+$. For $d_i \in D_k$, comparisons (d_i, d_j) to m successors are sampled so that $j = 1 + (a \bmod k)$ for $a \in \{i+\lambda-1, i+2\lambda-1, \ldots, i+m\lambda-1\}$; when $j = i$ for some a, that comparison is not included in the sample. For $\lambda = 1$, S-Window corresponds to N-Window, and for $\lambda = 3$, for example, each document is compared to every third of its successors. The λ-skip deterministically controls the "globality" of a sample without increasing the amount $|C|$ of sampled comparisons compared to N-Window.

3.2 Aggregation

For each comparison $(d_i, d_j) \in C$, a pairwise model computes a preference probability p_{ij}, which indicates how likely d_i should be ranked above ($p_{ij} \geq 0.5$) or below d_j ($p_{ij} < 0.5$). From the probabilities computed for C, an aggregation method derives a relevance value s_i for each document d_i. We study five paradigmatically different aggregation methods.

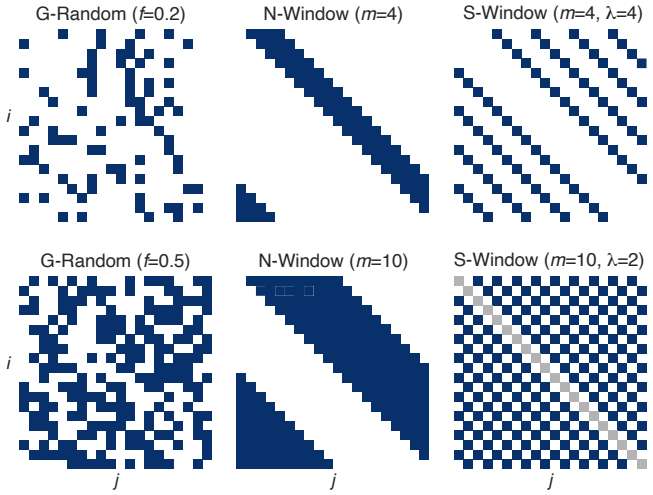


Figure 2: Example comparison sets of different sampling procedures for 20 documents at different sampling rates. If comparison (d_i, d_j) is sampled, cell i, j is colored blue; the grey cell for S-Window illustrates the omitted case $j = i$.

KwikSort. As a baseline, we use the KwikSort method [2]. It is an extension of the Quicksort algorithm for data with preferences and, in our case of top-k ranking, has an expected number of comparisons in $O(k \log k)$. First, a random document d_i is chosen to be the pivot. Then, all other documents are compared to the pivot placing the ones to be lower-ranked than d_i and the ones to be higher-ranked in separate subsets. These subsets are recursively ranked until a final ranking is obtained. KwikSort does not rely on a preceding sampling step; that the expected number of comparisons is in $O(k \log k)$ is a feature of the dynamic aggregation itself.

Additive Aggregation. Pradeep et al. [34] propose four different aggregation techniques based on preference probability summation. They find the symmetric sum of preference probabilities to yield the best effectiveness:

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

However, in our samples, not all comparisons are present so that we replace missing summands p_{ij} or $(1 - p_{ji})$ by 0.

Bradley-Terry Aggregation. The Bradley-Terry model [4] infers a latent score $s_i \in S$ for each document $d_i \in D_k$ based on the preferences expressed in the sampled comparison set C using maximum-likelihood estimation. In its original form, exponential score functions were used, which corresponds to a logistic regression on pairwise data [1] and can be expressed as:

$$\mathcal{L}(S, C) = \sum_{d_i > d_j} \log \frac{e^{s_i}}{e^{s_i} + e^{s_j}} + \sum_{d_i < d_j} \log \frac{e^{s_j}}{e^{s_i} + e^{s_j}} .$$

Here, $d_i > d_j$ denotes all comparisons $(d_i, d_j) \in C$ with $p_{ij} \geq 0.5$ and $d_i < d_j$ denotes all comparisons $(d_i, d_j) \in C$ with $p_{ij} < 0.5$. The unknown latent score set S is usually found via BFGS optimization [16] so that a ranking according to the s_i violates as few of the preferences from the comparison sample C as possible.

Input: Document set D_k, preference probabilities p_{ij}
Output: Score s for each $d \in D_k$

foreach $d_i \in D_k$ **do** $t_i \leftarrow \sum_{d_j \in D_k} p_{ij} - \sum_{d_j \in D_k} p_{ji}$;
while $D_k \neq \emptyset$ **do**
  $d_j \leftarrow \arg\max_{d_i \in D_k} t_i$;
  $s_j \leftarrow |D_k|$;
  $D_k \leftarrow D_k \setminus \{d_j\}$;
  foreach $d_i \in D_k$ **do** $t_i \leftarrow t_i - p_{ij} + p_{ji}$;
end

Algorithm 1: Greedy aggregation of preferences [10].

Greedy Aggregation. Cohen et al. [10] propose a greedy ordering algorithm that is proven to closely approximate the best total order in terms of the number of violated preferences. Algorithm 1 shows its pseudocode. In every iteration, the document d_j with the highest "potential" t_j[2] is appended to the re-ranking on the highest still unoccupied rank by setting score s_j accordingly. The potentials of the remaining documents are updated by canceling out the respective terms that include d_j. With sampling, the comparison set is incomplete; missing probabilities p_{ij} are set to zero.

PageRank Aggregation. A comparison set C induces a directed graph with D_k as nodes and comparisons as directed edges weighted with preference probabilities. We introduce a new aggregation method that computes the graph centrality measure PageRank [33], extended for weighted graphs [29], to rank the documents. The fundamental principle of PageRank is that nodes with incoming edges from nodes with high PageRank scores should also receive high PageRank scores. The respective PageRank-style aggregation of a score s_i for a document d_i then is

$$s_i = \gamma \cdot \frac{1}{|D_k|} + (1-\gamma) \cdot \sum_{(d_j,d_i)\in C} \frac{p_{ji}}{\sum_{l\in[1,k]} p_{jl}} \cdot s_j \,,$$

where using the components with the damping factor $\alpha \in [0,1]$ ensure convergence when computing the PageRank scores iteratively.

3.3 Initial Retrieval and Re-ranking

Following the experimental setup of Pradeep et al. [34] closely (see Figure 1), for each query, we first obtain an initial ranking using BM25 (PyTerrier implementation [28], default configuration). The top-1000 BM25 results are then re-ranked using monoT5 [31] in the pointwise re-ranking step. For the top-50 monoT5 results, duoT5 [34] infers preference probabilities in the second step of pairwise re-ranking, after sampling. For both, monoT5 and duoT5, we apply the largest available pre-trained version.[3] We use T5 instead of BERT variants, as T5 has been shown to be more effective [25]. To avoid repeated inferences in our experiments, all k^2-k pairwise preference probabilities are cached once for each query.

The maximum input length of transformer models is limited, so that a representative passage has to be chosen from each document for inference. Following the method of Dai and Callan [15], we split each document into fixed-length non-overlapping passages of about 250 words (using the TREC CAsT Y4 tools;[4] splits at sentence boundaries). Fixed-length passages have been shown to be more effective than variable-length passages [22]. In our pilot experiments, using the first passage was the most effective heuristic, so that we use them for preference probability inference.

[2]The "potential" basically tallies d_i's "wins" against other documents compared to its "losses" in terms of preference probabilities.
[3]monoT5: https://huggingface.co/castorini/monot5-3b-msmarco
duoT5: https://huggingface.co/castorini/duot5-3b-msmarco

4 EXPERIMENTAL SETUP

In this section, we introduce our evaluation measures, detailing in particular measures for consistency, complementarity, and transitivity of the aggregated relevance scores, and recap the used datasets.

4.1 Evaluation Measures

We follow similar studies of the mono/duoT5 models [34] and use nDCG@10 [21] to evaluate the retrieval effectiveness. When re-ranking the top-k results of a pointwise model, the k^2-k comparisons C_{all} usually performed by a pairwise model may result in inconsistent preference probabilities (1) at the level of a document pair and (2) at the level of document triples. We further examine these potential inconsistencies, as they can "complicate" the aggregation step and affect the retrieval effectiveness. At the document pair level, one would expect $p_{ij} \approx 1 - p_{ji}$ but pairwise models do not guarantees this and may predict both $d_i \succ d_j$ for the input pair (d_i, d_j) and $d_j \succ d_i$ for the input pair (d_j, d_i), or vice versa, where $d_i \succ d_j$ denotes a ranking preference of the left document d_i over the right one d_j. At the document triple level, transitivity may be violated as a model may predict $d_i \succ d_j$ and $d_j \succ d_l$ but $d_l \succ d_i$.

The consistency of an all-(k^2-k)-pairs comparison set C_{all} at the level of document pairs is the fraction of pairs $(d_i, d_j) \in C_{all}$ for which $p_{ij} \geq 0.5$ and $p_{ji} < 0.5$:

$$\textit{consistency}(C_{all}) \;=\; \frac{|\{(d_i,d_j) \in C_{all} : p_{ij} \geq 0.5 \text{ and } p_{ji} < 0.5\}|}{|C_{all}|}.$$

While consistency captures the comparison direction, also the numerical complementarity of how close $p_{ij} + p_{ji}$ is to the "ideal" 1 can be interesting. Thus, we also measure the ε-complementarity with respect to a margin of error ε as:

$$\varepsilon\textit{-complementarity}(C_{all}) \;=\; \frac{|\{(d_i,d_j) \in C_{all} : |p_{ij} + p_{ji} - 1| < \varepsilon\}|}{|C_{all}|}.$$

Finally, the transitivity of C_{all} measures the fraction of document triples for which the pairwise comparisons are transitive:

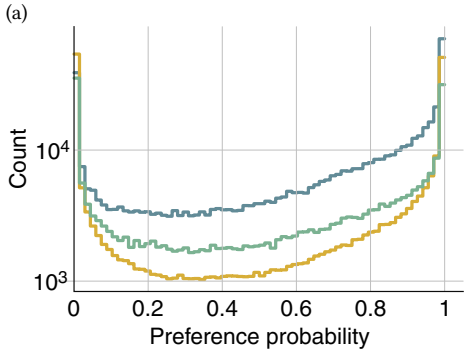
$$\textit{transitivity}(C_{all}) \;=\; \frac{|T|}{|T| + |I|}\,, \text{ where}$$

$$\begin{aligned} T \;=\; & \{(d_i,d_j,d_l) : p_{ij} \geq 0.5,\ p_{jl} \geq 0.5,\ \text{and}\ \ p_{il} \geq 0.5\} \cup \\ & \{(d_i,d_j,d_l) : p_{ij} < 0.5,\ p_{jl} < 0.5,\ \text{and}\ \ p_{il} < 0.5\} \qquad \text{and} \\ I \;=\; & \{(d_i,d_j,d_l) : p_{ij} \geq 0.5,\ p_{jl} \geq 0.5,\ \text{but}\ \ p_{il} < 0.5\} \cup \\ & \{(d_i,d_j,d_l) : p_{ij} < 0.5,\ p_{jl} < 0.5,\ \text{but}\ \ p_{il} \geq 0.5\}\,. \end{aligned}$$

The more the ε-complementarity for some small ε and the more the transitivity of some C_{all} approach 1, the more a total order between the documents is implied that probably can also be derived from some smaller comparison sample $C \subset C_{all}$.

[4]https://github.com/grill-lab/trec-cast-tools

(a)



(b) Consistency

Corpus	Mean	Std.	Min	Max
ClueWeb09	0.312	0.136	0.054	0.637
ClueWeb12	0.383	0.115	0.120	0.643
MS MARCO	0.498	0.126	0.136	0.731

(c) Transitivity

Corpus	Mean	Std.	Min	Max
ClueWeb09	0.783	0.079	0.616	0.954
ClueWeb12	0.742	0.064	0.611	0.898
MS MARCO	0.693	0.073	0.580	0.901

(d)

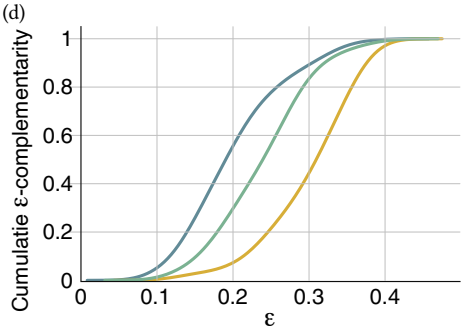


Figure 3: (a) Distribution of the pairwise duoT5 preference probabilities for the comparison set C_{all} of the top-50 pointwise results per corpus (log-scaled y-axis). (b) Preference probability consistency, (c) transitivity, and (d) cumulative ε-complementarity over ε. In the plots, the corpora are color-coded as ClueWeb09, ClueWeb12, and MS MARCO (Passage).

4.2 Evaluation Data

We employ three standard retrieval corpora in our experiments: the ClueWeb09, the ClueWeb12, and the MS MARCO passage corpus.

ClueWeb09. The ClueWeb09 corpus[5] 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 corpus[6] consists of 733 million documents crawled between April and May 2012. It was used for the ad-hoc search tasks of the TREC Web tracks 2013/14 [11, 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). The MS MARCO passage corpus [30] 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 2019/20 [13, 14], 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. [34].

Remark. In our evaluation of re-ranking results, we only consider judged documents. Evaluation scores calculated excluding unjudged documents correlate well with evaluations including them [37].

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 corpora. This supplies context to the ranking effectiveness evaluation in the second experiment (Section 5.2), in which we analyze rankings for different combinations of samplers and aggregators.

[5]http://lemurproject.org/clueweb09.php/
[6]http://lemurproject.org/clueweb12.php/

5.1 Evaluation of Pairwise Prediction Properties

For each topic from each corpus, we derive the duoT5 preference probabilities for the set C_{all} of all $k^2 - k$ pairwise comparisons for the top-50 results of the pointwise re-ranking and compute the statistics and measures per corpus shown in Figure 3.

The preference probabilities are highly skewed towards the extremes of the scale (cf. Figure 3a): for the majority of document pairs, the preference probability is approximately zero or one. This effect is stronger for the MS MARCO passage corpus (on which the model was trained) than for the ClueWeb corpora. Interestingly, the score distributions are not symmetric, but are slightly skewed towards 1.0 for all three corpora. Since the comparison set C_{all} contains both comparison directions for every pair, the observed skew directly suggests to further inspect how consistent, transitive, and complementary the preferences are for document pairs or triples.

Indeed, on average, only between half (MS MARCO) and a third of the comparisons (ClueWeb09) are consistent in their direction (cf. Figure 3b). Some variation across topics exists, yet the consistency is rather low in the majority of the topic-wise comparison sets. Further, also the cumulative ϵ-complementarity (cf. Figure 3d) confirms that the preferences of the pairwise duoT5 model are not that complementary (i.e., $p_{ij} + p_{ji} \neq 1$) for a document pair's two possible input orders. Only for a rather large value of $\varepsilon = 0.4$ all probabilities for pairs in all corpora are ε-complementary (i.e., $|p_{ij} + p_{ji} - 1| \neq 0.4$), and more than half of the pairs require an ϵ-value between 0.3 (MS MARCO) and 0.2 (ClueWeb09). For all corpora, almost no comparison pairs reach a complementarity of $\epsilon < 0.1$. Also the transitivity rates are consistently between 0.7 and 0.8 across all corpora with very little variation per topic. These observations (consistency and transitivity not that high) suggest that the comparison-based KwikSort aggregation will not output the most effective re-ranking and that very likely more than $O(k \log k)$ comparison pairs are needed in a sample for the other aggregation methods to "work around" the low consistency and lacking complementarity using more information.

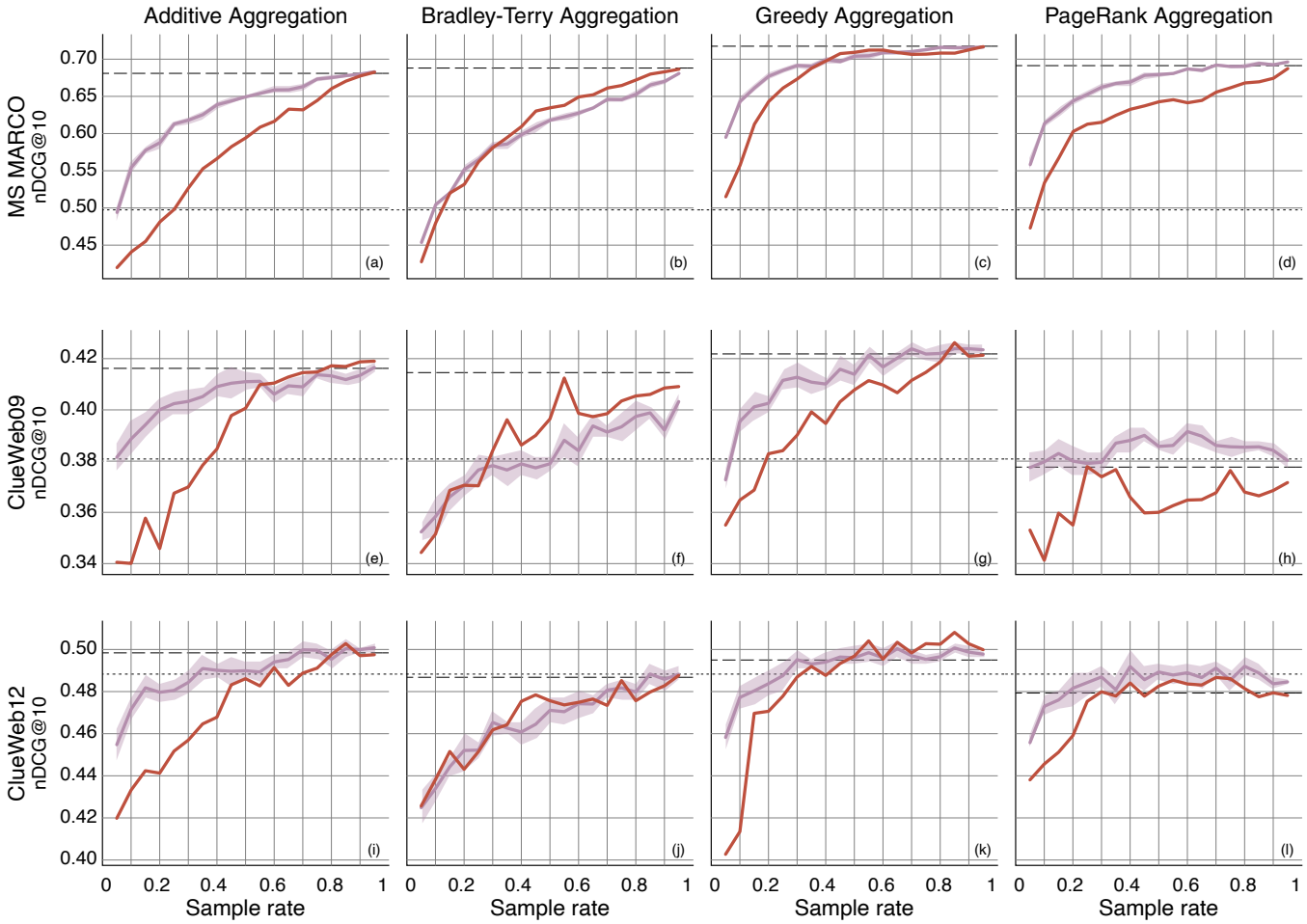


Figure 4: Effectiveness measured as nDCG@10 on three corpora (ClueWeb09, ClueWeb12, MS MARCO) for each aggregator and the two samplers G-Random and N-Window with different sampling rates. Dotted: pointwise re-ranking, dashed: unsampled C_{all}.

5.2 Evaluation of Ranking Effectiveness

To evaluate the effectiveness of different combinations of sampling and aggregation methods on all three corpora, we simulate runs on sparsified comparison sets at sample rates ranging from 0.05 to 0.95 in steps of 0.05. Each simulation is repeated ten times and the effectiveness is averaged to account for random variation in both the sampling and the aggregation step. In addition, baseline runs for each aggregator use the full comparison sets C_{all} per topic without any sampling. In total, 4950 runs are simulated. The pointwise ranking to be re-ranked by a pairwise model achieves nDCG@10 scores of 0.38 (ClueWeb09), 0.49 (ClueWeb12) and 0.50 (MS MARCO).

Effectiveness of KwikSort. From the observations in Section 5.1 (low consistency and low transitivity), it is clear that KwikSort with its pivot-based dynamic sampling of a rather "few" $O(k \log k)$ comparisons will not be able to achieve a really good effectiveness. Indeed, the nDCG@10 scores of KwikSort of 0.34 (ClueWeb09), 0.39 (ClueWeb12), and 0.42 (MS MARCO) are even lower than the pointwise effectiveness. We tested different pivot selection methods but all resulted in a similarly bad effectiveness. Using KwikSort for pairwise rank aggregation can thus not be recommended in settings with inconsistent and intransitive comparisons.

Effectiveness on the full comparison set C_{all}. Figure 4 shows the nDCG@10 of the non-KwikSort aggregation methods on each corpus. The dotted and dashed horizontal lines indicate the baseline effectiveness of the pointwise ranking and the pairwise re-ranking, respectively, when using the respective aggregation method on the complete comparison set C_{all} without sampling. Since the pointwise and the C_{all}-aggregation effectiveness are thus independent of the sampling rate, they are depicted as horizontal lines.

Using the full comparison set C_{all} without any sampling, greedy aggregation yields the most effective re-rankings for the ClueWeb09 (Subplot 4c) and the MS MARCO passage corpus (Subplot 4g) while additive aggregation is the most effective on the ClueWeb12 (Subplot 4i). On MS MARCO (first row of Figure 4), all aggregation methods are approximately equally effective when using the full comparison set C_{all} (dashed lines show about the same nDCG@10).

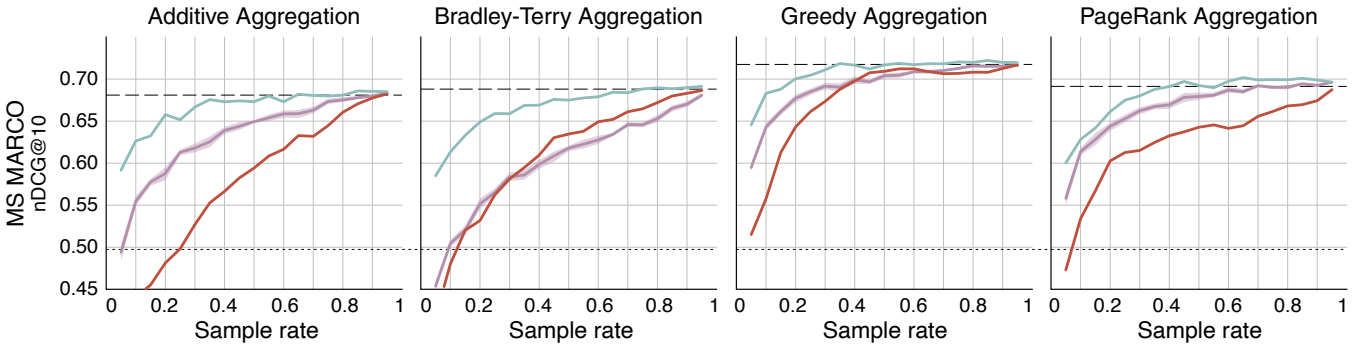


Figure 5: nDCG@10 on MS MARCO for each aggregator at different sampling rates for S-Window, G-Random, and N-Window. Dotted line: pointwise ranking, dashed line: effectiveness on unsampled C_{all}.

On the ClueWeb corpora, though, PageRank aggregation is less effective than the pointwise ranking (Subplots 4h and l; dashed line below dotted line) and also Bradley-Terry aggregation struggles on the ClueWeb12 (Subplot 4j). Only additive and greedy aggregation always improve upon the pointwise ranking when using the full comparison set C_{all} (Subplots 4e, g, i, and k) but the improvement is smaller on the ClueWeb corpora. That the effectiveness and the improvement over the pointwise ranking are the highest on the MS MARCO passage corpus is not surprising since the duoT5 re-ranking model was trained on MS MARCO, and since the TREC Web track relevance judgments used to evaluate the effectiveness on the ClueWeb corpora are at the document level, while we only rank one passage per document due to input length limitations.

Effectiveness with G-Random and N-Window sampling. The color-coded curves in the plots of Figure 4 show the effectiveness of the different aggregation methods with G-Random or N-Window sampling at different sampling rates from the full comparison set C_{all}. Four trends are apparent across all corpora.

First, N-Window sampling results in less effective re-rankings than G-Random sampling in nearly all cases, especially at smaller sampling rates. This effect is particularly noticeable for additive aggregation (Subplots 4a, e, and i). One reason probably is that the more "local" comparisons of N-Window are likely to yield less extreme comparison probability differences that decrease the overall separability of document pairs in sparse sampling setups. Also, inconsistencies in pairwise judgments are more likely for "local" pairs. Overall, this indicates that the global context of documents (which is better represented by G-Random) is important to obtain effective re-rankings via aggregation.

Second, greedy aggregation is the most effective aggregation method for both, G-Random and N-Window sampling (comparing Subplots 4c, g, and k to the rest). It is also the only aggregation method for which the effectiveness on the full C_{all} is reached by some sparsified comparison sets.

Third, the effectiveness degradation is not linear with respect to the sample rate (all subplots), but drops sharply below 15–20%. This suggests a lower bound of comparisons needed to derive good rankings from the pairwise comparisons of duoT5 which lack in consistency and transitivity.

Fourth, Bradley-Terry- and PageRank-aggregated re-rankings often are the least effective (Subplots 4b, f, and j, as well as d, h, and l). A possible reason for Bradley-Terry is similar to the bad effectiveness of KwikSort-aggregated re-rankings: Bradley-Terry only takes the direction of a comparison into account but not the magnitude of the respective probability. With the inconsistent probabilities of duoT5 that lead to inconsistent comparison directions, Bradley-Terry cannot derive good final re-rankings—just like KwikSort. In case of PageRank aggregation, also the inconsistent probabilities that are used as edge weights, might "confuse" the actual derivation of the PageRank scores.

Effectiveness with S-Window sampling. To find a good value for λ in S-Window sampling, we run a grid search over $\lambda = 2 \ldots 15$, separately for all sample rates (0.05 to 0.95 in steps of 0.05). We use five-fold cross validation to determine the best choice on the MS MARCO corpus, as the overall effectiveness gains on the ClueWeb corpora were too small to meaningfully distinguish between setups. Figure 5 shows the nDCG@10 effectiveness on MS MARCO of the run with the optimal λ-value for each sample rate. The best runs for G-Random and N-Window are also shown for reference.

Re-rankings aggregated from S-Window samples are more effective by a margin for each of the aggregation methods at all sampling rates. The combination of S-Window sampling with greedy aggregation allows for a rather stable effectiveness down to sampling only 30% of the comparisons. Even when using an order of magnitude fewer comparisons (i.e., $\approx$ 10% of C_{all}), a competitive effectiveness is achieved (nDCG@10 only 0.04 less).

The best values for λ are between 7 and 10 in most cases; they are not correlated with the window size (Pearson's $\bar{\rho} = 0.04$). Already the better effectiveness of aggregated rankings using G-Random sampling over N-Window sampling suggests that the global context is important when sampling comparisons. Also the rather large best-working λ-values for S-Window sampling corroborate this since even for small sample sizes m they ensure that the sampled comparisons cover a pretty "global" context. For large sample sizes m, λ is not as important as the sample then already covers a larger amount of the full comparison set C_{all}.

Table 1: Effectiveness on the MS MARCO corpus as nDCG@10 for the full comparison set C_{all} and the lowest similarly effective sampling rate (non-significant nDCG@10 difference; delta in brackets) per sampling method and aggregator. Bonferroni-correction for all (incl. hidden) tests per row.

Aggregator	nDCG@10	Lowest Similarly Effect. Sampl. Rate		
	Unsampled C_{all}	S-Window	G-Random	N-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)

Minimal sampling rates. Table 1 shows the minimum attainable sampling rates on the MS MARCO corpus for which each combination of sampling and aggregation method is not significantly less effective in terms of nDCG@10 than the respective aggregation on the full comparison set C_{all}. Per aggregator, the difference of the runs for each of the 19 sampling rates is tested against the run that aggregates a ranking from the full comparison set C_{all} using a paired Student's t-test with an α-level of 0.05 and Bonferroni correction for the multiple tests. For G-Random with potentially different effectiveness scores for the 10 runs per sampling rate, we use the least effective run per sampling rate in terms of nDCG@10 to increase the overall confidence in case of observed differences.

Among the sampling strategies, S-Window achieves the by far lowest sampling rates per aggregator without hurting the retrieval effectiveness too much. About the same effectiveness is possible with S-Window for additive, greedy, and PageRank aggregation with just one third of the usually used unsampled comparisons. G-Random and N-Window need more comparisons with any aggregation method to achieve the same re-ranking effectiveness and, in fact, only lead to some substantial savings compared to the unsampled C_{all} for PageRank aggregation (G-Random) or greedy aggregation (N-Window).

Among the aggregation strategies, additive and Bradley-Terry aggregation are the least effective and all sampling methods need larger sample rates for Bradley-Terry than for the other aggregators. Greedy aggregation leads to the best effectiveness and G-Random and N-Window achieve their lowest sampling rates without effectiveness loss for greedy aggregation.

Overall, the best combination in terms of effectiveness and sampling rate is greedy aggregation with S-Window sampling: with about one third of the usual $k^2 - k$ comparisons, the best effectiveness can be reached.

6 CONCLUSION

In this paper, we analyze several methods to substantially reduce the quadratic number of document comparisons usually conducted in pairwise re-ranking with transformers. To this end, we introduce a sampling step at the beginning of the pairwise re-ranking and adapt the aggregation step to derive relevance scores for a re-ranking from smaller samples of the pairwise comparisons. By comparing combinations of three sampling methods and five aggregation methods, we show that only one third of the comparisons are needed to achieve competitive re-ranking effectiveness and that also an order of magnitude less comparisons still can yield only a very slightly decreased effectiveness.

Compared to the usually applied additive rank aggregation without sampling, our new combination of skip-window sampling with greedy aggregation achieves an even better effectiveness at only about one third of the comparisons. When tolerating a very slight loss in effectiveness, even an order of magnitude fewer comparisons suffice. The more local exhaustive window sampling method leads to less effective rankings for which larger samples are needed than for skip-window or a global random sample. This suggests that a good sample of pairwise comparisons should not just sample from a very local environment per rank in the pointwise ranking.

Sparsification in pairwise re-ranking opens up new areas of research. Of the sampling paradigms evaluated (random vs. structured and local vs. global), the global structured sampling works better than the random one. Still, the samples are static in the sense that the sample is pre-computed before aggregation. Dynamic sampling techniques, in which new comparisons could be selected even during aggregation could merit further analyses. Sparsification could also be used to increase the depth of the pairwise re-ranking rather than its efficiency. Instead of minimizing the comparison budget for a fixed depth k, a fixed comparison budget can be used to maximize k. For example, the usually recommended depth $k = 50$ requires 2,450 comparisons ($k^2 - k$) for traditional pairwise re-ranking. A sampling rate of 30% (or 10%) now allows a re-ranking depth of $k = 90$ ($k = 157$) for a budget of about 2,450 comparisons. This may have a strong effect for recall-intensive retrieval tasks.

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] Roger Fletcher. 1987. *Practical Methods of Optimization* (2 ed.). John Wiley & Sons, New York.

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

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

[19] 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

[20] 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.

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

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

[23] 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.

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

[25] 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.

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

[27] 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).

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

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

[30] 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).

[31] 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.

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

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

[34] 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.

[35] 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*.

[36] 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.

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

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

[39] 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.

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

[41] 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.

[42] 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).

[43] 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.

[44] 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.

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

[46] 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.