Set-Encoder: Permutation-Invariant Inter-Passage Attention for Listwise Passage Re-Ranking with Cross-Encoders

Ferdinand Schlatt[1], Maik Fröbe[1], Harrisen Scells[2], Shengyao Zhuang[3,4], Bevan Koopman[3], Guido Zuccon[4], Benno Stein[5], Martin Potthast[6,7,8], Matthias Hagen[1]

[1]Friedrich-Schiller-Universität Jena, [2]Leipzig University, [3]CSIRO, [4]University of Queensland, [5]Bauhaus-Universität Weimar, [6]University of Kassel, [7]hessian.AI, [8]ScadDS.AI

Correspondence: ferdinand.schlatt@uni-jena.de

Abstract

Existing cross-encoder re-rankers can be categorized as pointwise, pairwise, or listwise models. Pair- and listwise models allow passage interactions, which usually makes them more effective than pointwise models but also less efficient and less robust to input order permutations. To enable efficient permutation-invariant passage interactions during re-ranking, we propose a new cross-encoder architecture with inter-passage attention: the Set-Encoder. In Cranfield-style experiments on TREC Deep Learning and TIREx, the Set-Encoder is as effective as state-of-the-art listwise models while improving efficiency and robustness to input permutations. Interestingly, a pointwise model is similarly effective, but when additionally requiring the models to consider novelty, the Set-Encoder is more effective than its pointwise counterpart and retains its advantageous properties compared to other listwise models. Our code and models are publicly available at `https://github.com/webis-de/set-encoder`.

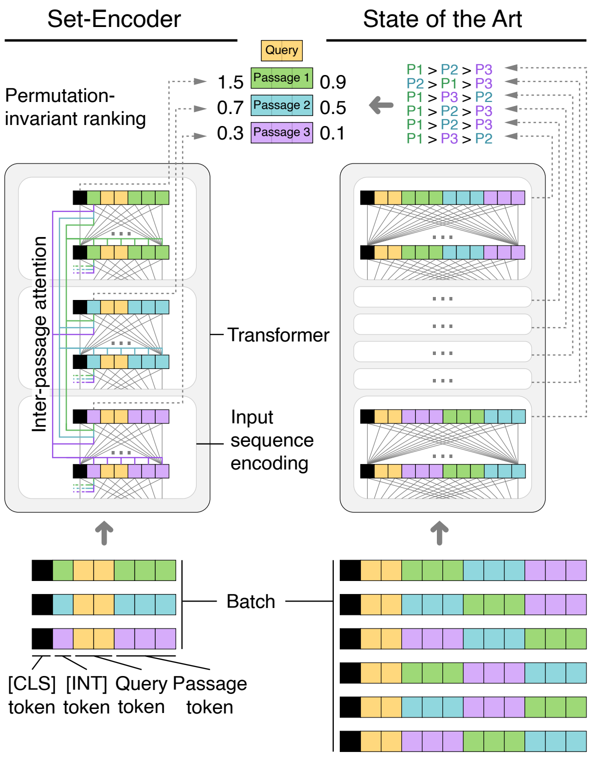

Figure 1: Comparison of our Set-Encoder architecture with state-of-the-art listwise re-rankers for three input passages. List-wise re-rankers concatenate the input passages, leading to potentially inconsistent rankings. Many (or all) permutations are re-ranked for optimization. Instead, the Set-Encoder uses novel [INT] tokens for permutation-invariant inter-passage attention.

1 Introduction

Existing cross-encoders for passage re-ranking (Nogueira et al., 2019; Nogueira and Cho, 2020; Nogueira et al., 2020; Pradeep et al., 2021, 2022; Zhuang et al., 2022) lack a central desirable property of learning-to-rank models (Pang et al., 2020): permutation-invariant interactions between the input passages. Passage interactions help to improve the ranking effectiveness through information exchange, while permutation invariance ensures that the same ranking is output independent of the ordering of the input passages.

Pointwise cross-encoders process passages independently of one another (Nogueira and Cho, 2020; Nogueira et al., 2020; Pradeep et al., 2022; Zhuang et al., 2022). As such, they are permutation-invariant by design but cannot cannot model passage interactions. Previous pairwise (Nogueira et al., 2019; Pradeep et al., 2021) and listwise cross-encoders (Sun et al., 2023; Pradeep et al., 2023a,b; Tamber et al., 2023) model passage interactions by concatenating passages in the input sequence. While these models are often more effective, the derived rankings depend on the order in which the input passages are concatenated. To avoid potentially inconsistent rankings for different input permutations, many (or all) permutations are re-ranked to optimize a final fused re-ranking's effectiveness; visualized in Figure 1 (right).

We propose the *Set-Encoder*, a new cross-encoder architecture that models passage interactions in a permutation-invariant way; visualized in Figure 1 (left). Instead of concatenating the input passages, the Set-Encoder processes them like a poitnwise cross-encoder as different sequences in a batch. To enable passage interactions, the Set-Encoder lets each sequence aggregate information in special [INT] interaction embedding tokens that all other sequences can attend to. The model is permutation-invariant as all [INT] tokens share the same positional encoding. We call this new attention pattern *inter-passage attention*.

In experiments on the TREC 2019 and 2020 Deep Learning tracks (Craswell et al., 2019, 2020) and in TIREx (Fröbe et al., 2023), our Set-Encoder is as effective as state-of-the-art LLM-based rerankers (Zhuang et al., 2022; Pradeep et al., 2023b). At the same time, it is robust to ranking permutations and orders of magnitude more efficient. However, contrasting previous work (Nogueira et al., 2020; Pradeep et al., 2021; Buyl et al., 2023), we find inter-passage attention to not be necessary, as a pointwise model (Schlatt et al., 2024) is similarly effective. To demonstrate the advantages of listwise models, we fine-tune the Set-Encoder to re-rank passages not just according to relevance but also novelty and find the Set-Encoder to then be more effective than its pointwise counterpart.

2 Related Work

Pre-trained transformer-based language models are currently the most effective passage re-rankers (Lin et al., 2022). The models can roughly be divided into two types: bi-encoders and cross-encoders. Bi-encoders process the query and passage separately, aggregate the semantic information into embeddings, and compare these embeddings for similarity (Reimers and Gurevych, 2019). Cross-encoders receive a query and a passage as input and output a relevance score (Nogueira and Cho, 2020). Note that, for brevity, we refer to all transformer-based language models that process a query and a passage simultaneously, be they encoder-only (Nogueira and Cho, 2020), decoder-only (Zhuang et al., 2022; Pradeep et al., 2023a,b), or encoder–decoders (Nogueira et al., 2020; Zhuang et al., 2022), as cross-encoders.

Cross-encoders are especially effective as they can explicitly model query–passage interactions. However, current cross-encoders lack a central property that previously improved the effectiveness and efficiency of feature-based learning-to-rank models (Lee et al., 2019; Pang et al., 2020; Pobrotyn et al., 2020; Pasumarthi et al., 2020; Buyl et al., 2023): modelling passage interactions in a permutation-invariant way. Passage interactions improve the effectiveness of re-ranking models by allowing passages to exchange information and permutation invariance is essential for efficiency. Without permutation invariance, a model is sensitive to ranking perturbations and must test multiple permutations to achieve a good effectiveness.

Previous work demonstrated that passage interactions can improve cross-encoders' effectiveness. For example, duo cross-encoders concatenate the query with two passages to predict which passage of the pair should be ranked higher (Pradeep et al., 2021; Nogueira et al., 2019). However, predictions of duo cross-encoders are neither symmetric nor transitive (Gienapp et al., 2022); switching the order of the input pair can lead to a different ranking preference. Therefore, duo cross-encoders must score all passage pairs for maximum effectiveness.

More recently, decoder-only LLMs simultaneously re-rank up to 20 passages (Sun et al., 2023; Pradeep et al., 2023a,b). However, next to being expensive to run, LLM-based cross-encoders are also sensitive to ranking perturbations. Several permutations need to be tested for maximum effectiveness, despite heuristics to minimize the number of permutations to test (Tang et al., 2023).

Existing cross-encoders that model passage interactions are not permutation-invariant because they concatenate passages in the input. Our Set-Encoder follows a different strategy and processes passages in parallel. Like a bi-encoder, the Set-Encoder encodes the semantic information of a passage in a single embedding vector. This embedding vector is shared with all other passages, enabling passage interactions while ensuring permutation invariance. Our respective inter-passage attention pattern is somewhat similar to the attention pattern of the Fusion-in-Encoder (FiE) model proposed for question answering (Kedia et al., 2022). FiE enables passage interactions within an encoder by adding additional so-called global tokens to which all passage tokens can attend to, and the global tokens can attend to all passage tokens. However, no direct interaction between passage tokens is possible. In contrast, our Set-Encoder allows for direct passage interactions as all passage tokens can attend to a single new [INT] token of each other passage.

3 The Set-Encoder Model

Our Set-Encoder introduces inter-passage attention, a novel attention pattern to model permutation-invariant interactions between passages. Inter-passage attention addresses three challenges: (1) input passages must be able to attend to each other, (2) the interactions between the passages must not encode any positional information about the passage ordering, and (3) the interactions should be "lightweight" enough for efficient fine-tuning.

Previous work has addressed the first challenge (attention between passages) by concatenating the query and multiple passages into one input sequence, visualized in Figure 1 (right). However, concatenation violates the second challenge: the language model's positional encodings (used to determine a token's position) span across passage boundaries and thus encode the order of passages in the sequence. Furthermore, concatenation also violates the third challenge (efficiency), as the language model's computational cost scales quadratically with the length of the input sequence.

Instead of concatenating passages, inter-passage attention processes the passages in parallel with individual input sequences per passage, as visualized in Figure 1 (left). Each input sequence's positional encodings start from zero, so no information about the passage order is encoded (second challenge). To still let the passages attend to each other (first challenge), we use a special passage interaction [INT] token that aggregates semantic information about its passage and to which tokens from other sequences can attend. The Set-Encoder allows passage interactions solely through these [INT] tokens, making inter-passage attention computationally efficient (third challenge).

3.1 Permutation-Invariant Input Encoding

The standard cross-encoder computes a relevance score for query–passage pairs (q, d) given as token sequences $t_{1_q} \dots t_{m_q}$ and $t_{1_d} \dots t_{m_d}$. In this work we use BERT-style encoding (Devlin et al., 2019), meaning the final tokens t_{m_q} and t_{m_d} are special [SEP] separator tokens. The concatenated sequence is prepended by a special [CLS] classification token t_c. The resulting input sequence $t_c\, t_{1_q} \dots t_{m_q}\, t_{1_d} \dots t_{m_d}$ is then passed through a transformer-based encoder model. Finally, the relevance score of d for q is computed by a linear transformation on the final contextualized embedding of the [CLS] token t_c.

The Set-Encoder receives a query q and a set of passages $\{d_1, \dots, d_k\}$ as input and computes a relevance score for each passage. To this end, the Set-Encoder builds a set of input sequences by prepending a [CLS] token, an [INT] token, and the query sequence to each passage individually. The set of input sequences is then processed simultaneously, similar to batched processing with a standard cross-encoder, but with attention from an input sequence's tokens to the [INT] tokens of the other input sequences (see Section 3.2). The batched input sequences are passed through an encoder, and the relevance scores of the passages d_i for q are computed by a linear transformation on the passage's final [CLS] token embedding.

Our batched input encoding is permutation-invariant as each input sequence's positional encodings start from zero. For example, the [INT] token is always at position 1 and the first token of each passage at position $m_q + 1$. The model then cannot distinguish between different permutations of the input sequences and, therefore, also not between different permutations of the passages.

3.2 Inter-Passage Attention

The Set-Encoder allows every sequence to attend to the [INT] tokens of every other sequence. Our intuition is that these [INT] tokens aggregate the semantic information from their sequence and can share it with all other sequences.

Using single tokens for passage interactions is on the efficiency end of a spectrum of possible variants for exchanging information between passages. The other extreme is allowing all tokens to attend to one another, as is done by previous listwise cross-encoder by concatenating input passages. Sharing more tokens can make the information exchange between passages more fine-grained but makes it computationally more expensive. Increasing the number of passage interaction tokens increases the computational cost quadratically. We hypothesize that using single tokens for information exchange suffices for effective passage interactions while being drastically more efficient.

Our hypothesis is motivated by the token-based semantic information aggregation in many model architectures. For instance, the [CLS] token captures a query or passage's semantic information in bi-encoders. In standard cross-encoders, the [CLS] token aggregates query–passage information to compute a relevance score. We hypothesize that our additional [INT] token can also aggregate semantic information and share this information with other passages. Note that we tested directly using the [CLS] token for passage interactions. We found this variant of passage interaction to produce effective re-rankers but incapable of learning passage interactions (see Section 4.2).

To implement inter-passage attention, we modify the original dot-product attention function of the transformer (Vaswani et al., 2017):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{h}}\right) V\,,$$

where Q, K, and V are matrices of embedding vectors and h is the dimensionality of the embeddings. Each vector in the matrices corresponds to a token from the encoder's input sequence.

In the Set-Encoder's input representation, we instead have vector matrices $Q^{(i)}$, $K^{(i)}$, and $V^{(i)}$ for each passage d_i's input sequence. To allow the i-th input sequence to attend to the [INT] tokens of all other input sequences, we append the [INT] token's embedding vectors from the other input sequences' embedding matrices $K^{(j)}$ and $V^{(j)}$ to the input sequence's embedding matrices $K^{(i)}$ and $V^{(i)}$.

Specifically, let $\bar{K}^{(i)} = [K_2^{(j)} : j \neq i]$ and $\bar{V}^{(i)} = [V_2^{(j)} : j \neq i]$ be the concatenated matrices of the [INT] token's embedding vectors from $K^{(j)}$ and $V^{(j)}$ of every passage $d_j \neq d_i$, with $[\cdot\,\cdot]$ denoting column-wise vector / matrix concatenation (i.e., $[MM']$ is equal to a matrix whose "left" columns come from M and whose "right" columns come from M'). Once the [INT] tokens from the other passages' embedding vectors have been appended to the sequence matrices, we obtain the inter-passage attention pattern: $\text{Attention}(Q^{(i)}, [K^{(i)}\bar{K}^{(i)}], [V^{(i)}\bar{V}^{(i)}])$.

3.3 Fine-tuning

We mostly follow the two-stage cross-encoder fine-tuning approach proposed by Schlatt et al. (2024) to fine-tune the Set-Encoder. See Appendix B for details on hyperparameters. In the first stage, a cross-encoder is fine-tuned on MS MARCO (Nguyen et al., 2016) relevance labels using Localized Contrastive Estimation (LCE) (Gao et al., 2021):

$$\mathcal{L}_{\text{LCE}} = -\log \frac{\exp(s_+)}{\sum_{i=1}^{k} \exp(s_i)}\,.$$

For a single query, the loss is computed over the predicted relevance scores s_i for a set of passages $\{d_1, \dots d_k\}$ of which one is a labeled relevant passage d_+. The other $k-1$ passages are sampled from the top 200 passages retrieved by ColBERTv2 for the training query (Santhanam et al., 2022).

In the second stage, the model is distilled from LLM-based cross-encoder rankings using the Rank-DistiLLM dataset (Schlatt et al., 2024) and the RankNet loss (Burges et al., 2005):

$$\mathcal{L}_{\text{RankNet}} = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{1}_{r_i<r_j} \log(1 + e^{s_i - s_j})\,,$$

where $\mathbb{1}$ is the indicator function and r_i is the rank of a passage d_i assigned by an RankZephyr (Pradeep et al., 2023b), a state-of-the-art LLM-based cross-encoder.

3.4 Fine-tuning to Detect Duplicates

In our experiments, we found the Set-Encoder does not require passage interactions to be effective and, therefore, does not make use of the [INT] token to share information between passages (see Section 4.1 for more details). To ensure the Set-Encoder learns to model passage interactions, we modify the first-stage fine-tuning step to include a duplicate detection loss.

We randomly sample a passage $d_\times$ from the set of input passages, duplicate it, and add it to the set of input passages as passage d_{k+1}. We then add a binary classification head to the model that outputs a probability p_i whether a passage d_i is the duplicate passage. Our duplicate-aware LCE is then LCE plus the binary cross-entropy loss over the model's output probabilities:

$$\mathcal{L}_{\text{DA-LCE}} = \mathcal{L}_{LCE} + \sum_{i=1}^{k} \mathbb{1}_{d_i = d_\times} \log p_i + \mathbb{1}_{d_i \neq d_\times} \log(1 - p_i)\,.$$

3.5 Fine-tuning to Rank According to Novelty

To demonstrate the advantages of listwise models, we fine-tune the Set-Encoder to rank passages according to relevance and novelty, meaning a passage's utility to a user depends on its relevance and if it provides novel information compared to passages ranked above it (Clarke et al., 2008). Due to a lack of large-scale training datasets for this task,[1] we create a new dataset, Rank-DistiLLM-Novelty.

[1] We attempted to fine-tune the Set-Encoder using the diversity tracks from TREC Web 2009–2014 but found the dataset to be too small to yield effective models.

It takes advantage of the fact that MS MARCO contains many near-duplicate passages. See Appendix C for examples. For a ranking from the Rank-DistiLLM dataset, we use scikit-learn's (Pedregosa et al., 2011) agglomerative clustering algorithm to group all passages with a word-based Jaccard similarity greater than 0.5. We apply the same strategy to the relevance labels of the TREC Deep Learning 2019 and 2020 tasks for evaluation (Craswell et al., 2019, 2020).

To fine-tune a model to rank passages according to relevance and novelty, we propose a new novelty-aware RankNet loss function. Intuitively, the loss penalizes a model if it ranks a less relevant passage higher than a more relevant one or a near-duplicate passage higher than a non-duplicate one. Let c_i be the cluster of near-duplicate passages to which d_i belongs. The novelty-aware RankNet loss then is

$$\mathcal{L}_{\text{NA-RankNet}} = \sum_{i=1}^{k}\sum_{j=1}^{k} \mathbb{1}_{\bar{r}_i<\bar{r}_j} \log(1+e^{s_i-s_j}),$$
$$\bar{r}_i = r_i \cdot (1 - \max_{j=1\ldots k} \mathbb{1}_{c_i=c_j} \cdot \mathbb{1}_{s_i<s_j}),$$

where $\bar{r}_i$ is an adjusted relevance label that sets the relevance of a passage to zero if the model ranks a near-duplicate passage higher.

4 Evaluation

To evaluate the effectiveness of the Set-Encoder, we conduct experiments on the TREC Deep Learning (DL) 2019 and 2020 passage tracks (Craswell et al., 2019, 2020) and the TIREx platform (Fröbe et al., 2023). We refer to Appendix D for details on the corpora and tasks in TIREx. For TREC DL, we report nDCG@10 when re-ranking the top 100 passages retrieved by either BM25 (Robertson et al., 1994) or ColBERTv2 (Santhanam et al., 2022). For our novelty-based ranking task, we report α-nDCG@10 (Clarke et al., 2008) with $\alpha = 0.99$ and consider all passages from a near-duplicate cluster to belong to a subtopic. With a high α value, only the first passage from a set of near-duplicates contributes to the gain. See Section 3.5 for further details on how we determine near-duplicate passage clusters. For TIREx, we report nDCG@10 micro-averaged across all tasks from a corpus and the macro-averaged arithmetic and geometric mean across all corpora.

We compare the Set-Encoder with RankGPT-4o (Sun et al., 2023) and RankZephyr (Pradeep et al., 2023b), two state-of-the-art listwise LLM-based cross-encoders. RankGPT-4o uses OpenAI's most recent GPT-4o model, while RankZephyr is open-source and uses a fine-tuned 7B LLaMa model (Touvron et al., 2023). Due to limited input length, LLM-based cross-encoders commonly use a windowed strategy to re-rank passages. Since GPT-4o has a maximum input length of up to 128k tokens, we also test a version that re-ranks all 100 passages simultaneously. We refer to this model as RankGPT-4o Full. We also compare the Set-Encoder with LiT5-Distill (Tamber et al., 2023), a recently proposed smaller LLM-based cross-encoder based on the T5 model (Raffel et al., 2020) and distilled from RankGPT-3.5 rankings. Finally, we include a pointwise monoELECTRA model that is identical to the Set-Encoder but does not use inter-passage attention (Schlatt et al., 2024).

Table 1: Comparison of the effectiveness in nDCG@10 of various cross-encoders on the TREC DL 2019 and 2020 tracks using BM25 or ColBERTv2 as the first stage retrieval models. The highest and second-highest scores per track are bold and underlined, respectively. Model sizes are given in number of parameters. † denotes a significant difference ($p < 0.05$, Holm-Bonferroni-corrected) to the Set-Encoder with 330M parameters.

Model	**Parameters**	**TREC DL 19**		**TREC DL 20**	
	First Stage	BM25	CBv2	BM25	CBv2
First Stage	–	0.480†	0.732†	0.494†	0.724†
RankGPT-4o	?	0.725	0.784	0.719	0.793
RankGPT-4o Full	?	<u>0.732</u>	0.781	0.711	0.796
RankZephyr	7B	0.719	0.749	0.720	<u>0.798</u>
LiT5-Distill	220M	0.696	0.753	0.679†	0.744†
monoELECTRA	110M	0.720	0.768	0.711†	0.770
	330M	**0.733**	0.765	<u>0.727</u>	**0.799**
Set-Encoder	110M	0.724	<u>0.788</u>	0.710†	0.777
	330M	0.727	**0.789**	**0.735**	0.790

4.1 Effectiveness of Inter-Passage Attention

In-domain Re-ranking Table 1 reports the effectiveness on TREC DL. The Set-Encoder is on par with substantially larger state-of-the-art LLM-based cross-encoders. None of the differences to LLM-based cross-encoders are statistically significant ($p < 0.05$, Holm-Bonferroni-corrected), but the 330M parameter variant is the most effective model in two of four retrieval settings. The smaller 110M-parameter Set-Encoder is slightly less effective in most settings but significantly worse than its larger variant in only a single setting.

Table 2: Effectiveness in nDCG@10 of various cross-encoders micro-averaged across all queries from a collection from the TIREx framework (Fröbe et al., 2023). Macro-averaged arithmetic and geometric means are computed across all corpora. Model sizes are given in the number of parameters. The highest and second-highest averaged scores and scores per corpus are bold and underlined, respectively.

Model	Parameters	Antique	Args.me	ClueWeb09	ClueWeb12	CORD-19	Cranfield	Disks4+5	GOV	GOV2	MEDLINE	NFCorpus	Vaswani	WaPo	A. Mean	G. Mean
First Stage	–	0.510	0.404	0.177	**0.364**	0.586	**0.008**	0.436	0.235	0.466	0.358	0.268	0.447	0.364	0.356	0.385
RankZephyr	7B	0.528	0.363	0.213	0.303	**0.767**	0.006	0.556	**0.294**	0.560	**0.457**	0.299	0.512	**0.508**	**0.413**	**0.453**
LiT5-Distill	220M	0.571	0.394	0.214	0.275	0.686	**0.008**	0.509	0.266	0.534	0.334	0.278	0.429	0.470	0.382	0.419
m.ELECTRA	110M	0.587	0.375	0.209	0.295	0.692	0.007	0.521	0.264	0.541	0.326	0.291	0.522	0.458	0.391	0.430
	330M	0.570	0.368	0.221	0.313	0.716	0.006	**0.559**	0.288	0.572	0.376	**0.301**	0.526	0.504	0.409	0.448
Set-Encoder	110M	0.588	0.374	0.216	0.299	0.683	0.007	0.534	0.272	0.543	0.321	0.291	0.523	0.461	0.393	0.431
	330M	**0.600**	**0.408**	**0.226**	0.310	0.702	0.006	0.553	0.285	**0.573**	0.348	0.297	**0.530**	**0.508**	0.411	0.450

Interestingly, the pointwise monoELECTRA model features similar effectiveness. The model using 330M parameters is the most effective in the other two out of four settings, and the smaller model is again only slightly less effective. The fact that a pointwise monoELECTRA model is on par with the Set-Encoder and state-of-the-art listwise LLM re-rankers suggests passage interactions are unnecessary for effective in-domain re-ranking.

Out-of-domain Re-ranking We find this result also translates to out-of-domain re-ranking. Table 2 reports the results for out-of-domain re-ranking across the diverse corpora contained in TIREx. We had to exclude RankGPT-based models since TIREx requires models be executed in a sandboxed environment without internet access. We want to highlight two noteworthy results.

First, the Set-Encoder and monoELECTRA are again on par with the current state-of-the-art model on TIREx, RankZephyr. On average, the large 330M-parameter Set-Encoder is slightly more effective than the large monoELECTRA model, but both feature very similar effectiveness compared to RankZephyr. This result suggests that passage interactions are also unnecessary for effective out-of-domain re-ranking. We attribute this in part to the TREC-style relevance assessments—Assessors judge query–passage pairs independently of one another. Therefore, a model can also score the passages independently to be effective. Passage interactions are beneficial in tasks where, for example, fairness, diversity, or novelty are considered, and we investigate this in more detail in Section 4.2.

Second, despite being fine-tuned on RankZephyr rankings, the Set-Encoder and monoELECTRA are more effective in most out-of-domain re-ranking tasks than their teacher model. The Set-Encoder achieves a higher effectiveness than RankZephyr on 6 out of 13 corpora, RankZephyr is more effective on 5 corpora, and they reach same effectiveness on 2. The results are similar for monoELECTRA, which is more effective than RankZephyr on 8 corpora and less effective on 4 with 1 tie. RankZephyr only reaches marginally higher average effectiveness than the other two cross-encoers by being substantially more effective on the two medical corpora, CORD-19 and MEDLINE. We attribute this to RankZephyr not being permutation invariant. Consequently, it is sensitive to the quality of the first-stage retrieval, as we investigate in Section 4.3.

4.2 Novelty Ranking

To better investigate the effect of passage interactions on re-ranking effectiveness, we compare the cross-encoders on a novelty ranking task. Models should rank passages according to relevance and place all but one of a set of lexically near-duplicate passages at the bottom of the ranking.

We fine-tune 110M parameter variants of the Set-Encoder and monoELECTRA on this task using our newly proposed novelty-aware RankNet loss (see Section 3.5). We also test the effect of fine-tuning our Set-Encoder to detect duplicate passages during first-stage fine-tuning using our newly proposed duplicate-aware LCE loss (see Section 3.4). Since monoELECTRA is incapable of detecting duplicates by design, we refrain from fine-tuning it on duplicate detection. For comparison, we also test augmenting the prompt for the LLM-based cross-encoders to account for novelty. See Appendix E for details on the prompt. Table 3 compares the nDCG@10 and α-nDCG@10 of various models.

Table 3: Comparison of nDCG@10 and α-nDCG@10 ($\alpha = 0.99$) of various cross-encoders on the TREC Deep Learning 2019 and 2020 tracks clustered into near-duplicate subtopics. The highest and second-highest scores per track are bold and underlined, respectively. † denotes a significant difference ($p < 0.05$, Holm-Bonferroni-corrected) compared to the Set-Encoder fine-tuned using $\mathcal{L}_{\text{DA-LCE}}$ and then $\mathcal{L}_{\text{NA-RN}}$.

Model	**Prompt**		**nDCG**		**α-nDCG**	
			2019	2020	2019	2020
First Stage	–		0.732	0.724	$0.700^\dagger$	$0.722^\dagger$
R.GPT-4o	Relevance		$0.784^\dagger$	$0.793^\dagger$	0.750	0.759
	Novelty		$0.778^\dagger$	$\mathbf{0.806}^\dagger$	0.741	$\underline{0.773}$
R.GPT-4o Full	Relevance		$0.781^\dagger$	$0.796^\dagger$	0.738	0.763
	Novelty		$\underline{0.785}^\dagger$	$\underline{0.803}^\dagger$	0.750	0.771
RankZephyr	Relevance		0.749	$0.798^\dagger$	$0.699^\dagger$	0.765
	Novelty		0.753	$0.800^\dagger$	$0.700^\dagger$	0.760
Model	**1. $\mathcal{L}$**	**2. $\mathcal{L}$**				
monoELEC.	$\mathcal{L}_{\text{LCE}}$	$\mathcal{L}_{\text{RN}}$	$0.768^\dagger$	$0.770^\dagger$	$0.718^\dagger$	$0.745^\dagger$
		$\mathcal{L}_{\text{NA-RN}}$	0.704	0.675	$\underline{0.785}$	0.753
Set-Encoder	$\mathcal{L}_{\text{LCE}}$	$\mathcal{L}_{\text{RN}}$	$0.780^\dagger$	$0.757^\dagger$	$0.733^\dagger$	$0.747^\dagger$
		$\mathcal{L}_{\text{NA-RN}}$	0.714	0.651	0.779	$0.743^\dagger$
	$\mathcal{L}_{\text{DA-LCE}}$	$\mathcal{L}_{\text{RN}}$	$\mathbf{0.788}^\dagger$	$0.777^\dagger$	$0.740^\dagger$	$0.752^\dagger$
		$\mathcal{L}_{\text{NA-RN}}$	0.710	0.690	**0.821**	**0.803**
Set-Enc. ~~[INT]~~	$\mathcal{L}_{\text{DA-LCE}}$	$\mathcal{L}_{\text{NA-RN}}$	0.707	0.670	0.773	$0.748^\dagger$

Relevance Unsurprisingly, we find that fine-tuning for relevance and novelty reduces the effectiveness of models when only evaluating relevance. Regarding nDCG@10, the Set-Encoder first fine-tuned with duplicate-aware LCE and fine-tuned with novelty-aware RankNet is significantly worse ($p < 0.05$, Holm-Bonferroni-corrected) than when fine-tuned with normal RankNet and also significantly worse than most LLM-based cross-encoders. The LLM-based cross-encoders do not seem to be affected by our novelty-aware prompt and only show minor differences in effectiveness.

Relevance and Novelty While the pointwise monoELECTRA has no access to listwise information, the model still improves in terms of α-nDCG@10 when fine-tuned with novelty-aware RankNet. The model learns to assign higher ranks to short passages and lower ranks to long passages, as short passages are less likely to have lexical near-duplicates in MS MARCO. See Appendix F for details. In contrast, the LLM-based cross-encoders are again largely unaffected by the novelty prompt.

The Set-Encoder is noticeably more effective than all other cross-encoders. While the differences are only significant for a subset of comparisons, it has a 0.036 and 0.030 higher α-nDCG@10 than the second-best models on the 2019 and 2020 tracks, respectively—the largest difference between two neighboring models when sorted by effectiveness.

However, the Set-Encoder is only effective when initially fine-tuned with duplicate-aware LCE to detect exact duplicates. When initially fine-tuned with normal LCE, the Set-Encoder cannot profit from inter-passage attention and is on par with a pointwise monoELECTRA model. In other words, when initially only fine-tuned with a relevance signal, the Set-Encoder does not learn to use inter-passage attention. Only when "priming" the model during initial fine-tuning to exchange information via inter-passage attention can it learn passage interactions and improve on the novelty ranking task.

Ablation Finally, we investigate the effectiveness of the Set-Encoder without adding an additional [INT] token and using the [CLS] token for interaction. Table 3 shows the model is unable to learn to detect duplicates during initial fine-tuning and is on par with the Set-Encoder fine-tuned with normal LCE. This result suggests that the [INT] token is crucial for the model to learn passage interactions.

4.3 Permutation Invariance

Positional Biases All previous listwise cross-encoders have an implicit positional bias by concatenating input passages. Most LLM-based cross-encoders also have an explicit positional bias due to the limited input length. For example, to re-rank 100 passages, the original RankGPT model, RankZephyr, and LiT5-Distill use a windowed strategy and re-rank only 20 passages at a time.

We visualize these explicit and implicit biases in Figure 2. It shows the average proportional rank changes for various cross-encoders when re-ranking the top 100 passages retrieved by BM25 for TREC DL 2019 and 2020. The lighter a pixel, the more frequently the model ranks a passage from a particular position to another particular position.

For RankGPT-4o, RankZephyr, and LiT5-Distill, the explicit bias from the windowed strategy is immediately evident in a distinct step pattern. RankGPT-4o Full, which does not use the windowed strategy and re-ranks all 100 passages at once, does not exhibit the step pattern but instead has a distinct diagonal line. The line indicates that the model tends to keep passages in the same po-

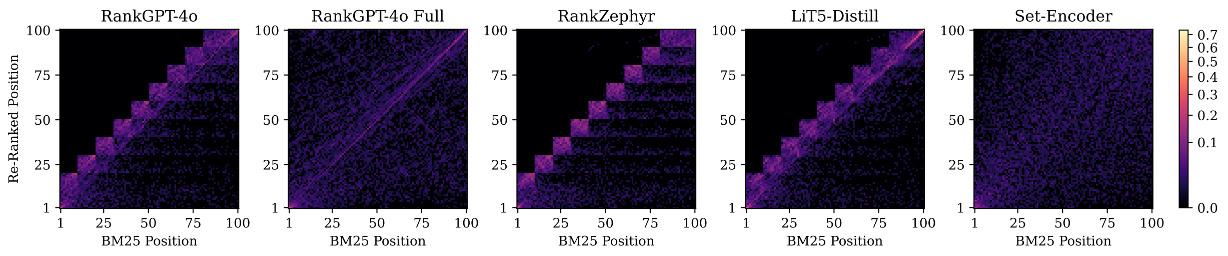

Figure 2: Proportional rank changes of various cross-encoders for re-ranking BM25 averaged across all queries from the TREC Deep Learning 2019 and 2020 tracks.

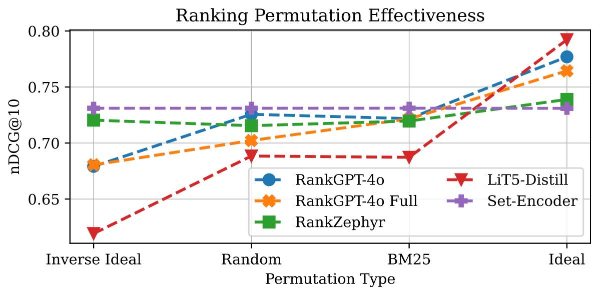

Figure 3: Effectiveness in terms of nDCG@10 for different permutations of BM25 rankings.

sition as in the original ranking. The Set-Encoder does not exhibit any immediately obvious positional biases; only a clustering of lighter pixels in the lower left corner and darker areas and the upper left and lower right quadrants show the Set-Encoder's ranking correlates to some degree with the original BM25 ranking.

Ranking Perturbations To test how these positional biases affect a cross-encoder's ranking effectiveness, we generate several "corrupted" perturbations of the top 100 passages retrieved by BM25 on the TREC DL 2019 and 2020 tracks: a random permutation, an ideal permutation, and a reverse-ideal permutation. Ideal and reverse-ideal permutations are generated by reordering the passages according to their relevance judgments. Figure 3 visualizes nDCG@10 of various LLM-based cross-encoders and our Set-Encoder on the different permutations.

The Set-Encoder is robust to the order of input passages and achieves the same effectiveness irrespective of the permutation. All other listwise cross-encoders are affected by the ranking permutations. RankGPT-4o, RankGPT-4o Full, and LiT5-Distill lose a substantial portion of their effectiveness when re-ranking the inverse ideal ranking and improve with increasingly better initial rankings. Only RankZephyr is comparatively robust to perturbations because it is fine-tuned to counteract positional biases, but it nonetheless fluctuates slightly.

We additionally emphasize that the Set-Encoder reaches a higher nDCG@10 than all LLM-based cross-encoders on the inverse ideal, random, and the original BM25 rankings. The LLM-based cross-encoders only achieve higher effectiveness when re-ranking the ideal permutation. Due to the positional biases, most LLM-based cross-encoders require a good initial ranking to be effective.

This result also explains how the Set-Encoder and monoELECTRA are more effective than their RankZephyr teacher model on most corpora in TIREx. In the distillation dataset, RankZephyr is applied to ColBERTv2, an already effective first-stage retriever. The distilled models learn from these very effective rankings and are then more effective than the teacher model when re-ranking lower quality initial rankings.

5 Conclusion

In this paper, we have introduced the Set-Encoder, a new cross-encoder architecture that models passage interactions in a permutation-invariant way. Contrasting previous work, our empirical results show that passage interactions do not necessarily improve a cross-encoder's re-ranking effectiveness: a pointwise model can be as effective as the Set-Encoder or other state-of-the-art LLM-based cross-encoders when only relevance is measured.

To demonstrate the benefits of passage interactions, we also evaluate the Set-Encoder on a novelty ranking task and show it is more effective in this more complex setting than its pointwise counterpart and previous listwise cross-encoders.

We also show that permutation invariance is crucial for effective listwise cross-encoders. Explicit and implicit positional biases in previous listwise cross-encoders lead to inconsistent and suboptimal effectiveness across ranking settings, while the Set-Encoder with no positional biases by design is on par or more effective than previous listwise cross-encoders, irrespective of the ranking setting.

6 Limitations

In this paper, we have evaluated the Set-Encoder with our new inter-passage attention pattern against previous pointwise and listwise cross-encoders. To our knowledge, our new pattern is the first to allow direct permutation-invariant interactions between input passages. Other patterns, like the Fusion-in-Encoder (Kedia et al., 2022) for question answering, model passage interactions indirectly using global tokens. As we were unable to find published models or code for the Fusion-in-Encoder approach, reproduction was difficult, so we instead opted to compare against state-of-the-art LLMs.

To demonstrate the advantages of listwise re-rankers over pointwise re-rankers, we introduced a new semi-synthetic novelty-ranking task. We are aware that our task is not fully representative of a real-world novelty ranking task and, additionally, it could be addressed by simply ranking passages according to relevance and then removing near-duplicate passages in a second step. Still, our goal was not to create a realistic ranking task but rather, due to a lack of suitable large-scale realistic novelty ranking datasets, we created a semi-synthetic task to demonstrate the potential benefits of passage interactions.

Finally, the LLM-based cross-encoder's effectiveness could be improved by tuning the prompt. Additional more thorough experiments are needed to further analyze this hypothesis.

7 Ethical Considerations

Fine-tuning and running large transformer-based language models requires considerable amounts of energy and thus may contribute to climate change. In our research, we have tried to minimize the environmental impact by fine-tuning models with (comparatively) few parameters.

Additionally, while our work derives from publicly available and widely used datasets and models, these may contain biases. We release our data, code, and models to the public but caution that they may contain biases that could be harmful if used in production systems.

References

Alexander Bondarenko, Maik Fröbe, Johannes Kiesel, Shahbaz Syed, Timon Gurcke, Meriem Beloucif, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2022. Overview of Touché 2022: Argument Retrieval. In *Proceedings of CLEF 2022*, pages 311–336.

Alexander Bondarenko, Lukas Gienapp, Maik Fröbe, Meriem Beloucif, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2021. Overview of Touché 2021: Argument Retrieval. In *Proceedings of CLEF 2021*, pages 450–467.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *Proceedings of ECIR 2016*, pages 716–722.

Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report MSR-TR-2010-82, Microsoft Research, Redmond, WA.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of ICML 2005*, pages 89–96.

Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff. 2006. The TREC 2006 Terabyte Track. In *Proceedings of TREC 2006*.

Maarten Buyl, Paul Missault, and Pierre-Antoine Sondag. 2023. RankFormer: Listwise Learning-to-Rank Using Listwide Labels. In *Proceedings of KDD 2023*, pages 3762–3773.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of ICLR 2020*.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 Terabyte Track. In *Proceedings of TREC 2004*.

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

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

Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. 2011. Overview of the TREC 2011 Web Track. In *Proceedings of TREC 2011*.

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

Charles L. A. Clarke, Falk Scholer, and Ian Soboroff. 2005. The TREC 2005 Terabyte Track. In *Proceedings of TREC 2005*.

Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and Diversity in Information Retrieval Evaluation. In *Proceedings of SIGIR 2008*, pages 659–666.

Cyril W. Cleverdon. 1991. The Significance of the Cranfield Tests on Index Languages. In *Proceedings of SIGIR 1991*, pages 3–12.

Kevyn Collins-Thompson, Paul N. Bennett, Fernando Diaz, Charlie Clarke, and Ellen M. Voorhees. 2013. TREC 2013 Web Track Overview. In *Proceedings of TREC 2013*.

Kevyn Collins-Thompson, Craig Macdonald, Paul N. Bennett, Fernando Diaz, and Ellen M. Voorhees. 2014. TREC 2014 Web Track Overview. In *Proceedings of TREC 2014*.

Nick Craswell and David Hawking. 2002. Overview of the TREC-2002 Web Track. In *Proceedings of TREC 2002*.

Nick Craswell and David Hawking. 2004. Overview of the TREC 2004 Web Track. In *Proceedings of TREC 2004*.

Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. 2003. Overview of the TREC 2003 Web Track. In *Proceedings of TREC 2003*, pages 78–92.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of TREC 2020*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of TREC 2019*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL 2019*, pages 4171–4186.

William Falcon and The PyTorch Lightning team. 2023. PyTorch Lightning.

Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Information Retrieval Experiment Platform. In *Proceedings of SIGIR 2023*, pages 2826–2836.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink Training of BERT Rerankers in Multi-stage Retrieval Pipeline. In *Proceedings of ECIR 2021*, pages 280–286.

Lukas Gienapp, Maik Fröbe, Matthias Hagen, and Martin Potthast. 2022. Sparse Pairwise Re-ranking with Pre-trained Transformers. In *Proceedings of SIGIR 2022*, pages 72–80.

Ralf Gommers, Pauli Virtanen, Matt Haberland, Evgeni Burovski, Warren Weckesser, Tyler Reddy, Travis E. Oliphant, David Cournapeau, Andrew Nelson, alexbrc, Pamphile Roy, Pearu Peterson, Ilhan Polat, Josh Wilson, endolith, Nikolay Mayorov, Stefan van der Walt, Matthew Brett, Denis Laxalde, Eric Larson, Atsushi Sakai, Jarrod Millman, Lars, peterbell10, C. J. Carey, Paul van Mulbregt, eric-jones, Kai, Nicholas McKibben, and Lucas Colley. 2024. scipy/scipy: SciPy 1.14.0rc1. Zenodo.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array Programming with NumPy. *Nature*, 585(7825):357–362.

Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. 2020. ANTIQUE: A Non-factoid Question Answering Benchmark. In *Proceedings of ECIR 2020*, pages 166–173.

William R. Hersh, Ravi Teja Bhupatiraju, L. Ross, Aaron M. Cohen, Dale Kraemer, and Phoebe Johnson. 2004. TREC 2004 Genomics Track Overview. In *Proceedings of TREC 2004*.

William R. Hersh, Aaron M. Cohen, Jianji Yang, Ravi Teja Bhupatiraju, Phoebe M. Roberts, and Marti A. Hearst. 2005. TREC 2005 Genomics Track Overview. In *Proceedings of TREC 2005*.

John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(03):90–95.

Akhil Kedia, Mohd Abbas Zaidi, and Haejun Lee. 2022. FiE: Building a Global Probability Space by Leveraging Early Fusion in Encoder for Open-Domain Question Answering. In *Proceedings of EMNLP 2022*, pages 4246–4260.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. 2016. Jupyter Notebooks – A Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of ICML 2019*, pages 3744–3753.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained Transformers for Text Ranking: BERT and*

Beyond. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Proceedings of ICLR 2019*.

Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Streamlining Evaluation with `ir-measures`. In *Proceedings of ECIR 2022*, pages 305–310.

Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with `ir_datasets`. In *Proceedings of SIGIR 2021*, pages 2429–2436.

Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proceedings of CIKM 2021*, pages 4526–4533.

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. In *Proceedings of COCO@NeurIPS 2016*.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. arXiv:1901.04085[v5].

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP 2020*, pages 708–718.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. arXiv:1910.14424.

The pandas development team. 2024. Pandas-dev/pandas: Pandas. Zenodo.

Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of SIGIR 2020*, pages 499–508.

Rama Kumar Pasumarthi, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Permutation Equivariant Document Interaction Network for Neural Learning to Rank. In *Proceedings of ICTIR 2020*, pages 145–148.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of NeurIPS 2019*, pages 8024–8035.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Przemysław Pobrotyn, Tomasz Bartczak, Mikołaj Synowiec, Radosław Białobrzeski, and Jarosław Bojar. 2020. Context-Aware Learning to Rank with Self-Attention. In *Proceedings of eCom@SIGIR 2020*.

Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. 2022. Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. In *Proceedings of ECIR 2022*, pages 655–670.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv:2101.05667.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. arXiv:2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv:2312.02724.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3980–3990.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, and Alexander J. Lazar. 2018. Overview of the TREC 2018 Precision Medicine Track. In *Proceedings of TREC 2018*.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, Alexander J. Lazar, and Shubham Pant. 2017. Overview of the TREC 2017 Precision Medicine Track. In *Proceedings of TREC 2017*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of TREC 1994*, pages 109–126.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. arXiv:2112.01488.

Ferdinand Schlatt, Maik Fröbe, Harrisen Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. 2024. A Systematic Investigation of Distilling Large Language Models into Cross-Encoders for Passage Re-ranking. arXiv:2405.07920.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of EMNLP 2023*, pages 14918–14937.

Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. arXiv:2312.16098.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2023. Found in the Middle: Permutation Self-Consistency Improves Listwise Ranking in Large Language Models. arXiv:2310.07712.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NeurIPS 2017*, pages 5998–6008.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3):261–272.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. arXiv:2005.04474.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Procedings of TREC 2004*.

Ellen M. Voorhees and Donna Harman. 1998. Overview of the Seventh Text REtrieval Conference (TREC-7). In *Proceedings of TREC 1998*.

Ellen M. Voorhees and Donna Harman. 1999. Overview of the Eigth Text REtrieval Conference (TREC-8). In *Proceedings of TREC 1999*.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The COVID-19 Open Research Dataset. arXiv:2004.10706.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. arXiv:2210.10634.

A Software

This work made use of the following software packages: HuggingFace Transformers (Wolf et al., 2020), ir_datasets (MacAvaney et al., 2021), ir_measures (MacAvaney et al., 2022), Jupyter (Kluyver et al., 2016), Lightning (Falcon and The PyTorch Lightning team, 2023), matplotlib (Hunter, 2007), NumPy (Harris et al., 2020), pandas (pandas development team, 2024), PyTerrier (Macdonald et al., 2021), PyTorch (Paszke et al., 2019), scikit-learn (Pedregosa et al., 2011), and SciPy (Virtanen et al., 2020; Gommers et al., 2024),

B Fine-Tuning Settings

We mostly follow Schlatt et al. (2024) for fine-tuning the Set-Encoder and monoELECTRA models. We use ELECTRA$_{\text{BASE}}$[2] and ELECTRA$_{\text{LARGE}}$[3] (Clark et al., 2020) checkpoints from HuggingFace (Wolf et al., 2020) as starting points for fine-tuning. We use the relevance labels

[2]`google/electra-base-discriminator`
[3]`google/electra-large-discriminator`

from the MS MARCO passage dataset (Nguyen et al., 2016) and the ColBERTv2 (Santhanam et al., 2022) hard negatives from Rank-DistiLLM (Schlatt et al., 2024) for first-stage fine-tuning. We truncate queries to at most 32 tokens and passages to 256 tokens and fine-tune monoELECTRA and Set-Encoder models for 20k steps with a batch size of 32 using LCE loss (Gao et al., 2021) and 7 negative examples per query. We also fine-tune the Set-Encoder using our newly proposed duplicate-aware LCE for at least 20k steps but at most 60k steps or until the duplicate-detection cross-entropy loss is smaller than 0.05 for at least 100 steps. The models are then further fine-tuned for 3 epochs on the ColBERTv2-then-RankZephyr distillation data also from Rank-DistiLLM using the RankNet loss function (Burges, 2010). For the novelty ranking task, we fine-tune models for 10 epochs, using α-nDCG@10 on TREC DL 2021 as a validation set. We use the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 10^{-5}. All models were trained on a single NVIDIA A100 40GB GPU.

C MS MARCO Duplicates

Figure 5 shows the top 8 passages from the Rank-DistiLLM dataset (Schlatt et al., 2024) for the query "which organelle contains the majority of the cell's genetic materials in an animal cell?". The passages are clustered based on their lexical similarity. If the word-based Jaccard similarity between two passages is greater than 0.5, we consider them as near-duplicates.

D TIREx

Table 4 provides an overview of the 31 retrieval tasks over 14 corpora contained in TIREx (Fröbe et al., 2023) used for evaluation. Citations for each corpus (except for Vaswani and WaPo, which do not have specific references) are provided below:

- Antique — QA Benchmark (Hashemi et al., 2020)
- Args.me — Touché (Bondarenko et al., 2021, 2022)
- ClueWeb09 — TREC Web Tracks (Clarke et al., 2009, 2010, 2011, 2012)
- ClueWeb12 — TREC Web Tracks, Touché (Collins-Thompson et al., 2013, 2014; Bondarenko et al., 2021, 2022)
- CORD-19 — TREC-COVID (Voorhees et al., 2020; Wang et al., 2020)
- Cranfield — Fully Judged Corpus (Cleverdon, 1991)
- Disks4+5 — TREC-7/8, Robust04 (Voorhees and Harman, 1998, 1999; Voorhees, 2004)

Table 4: Overview of the retrieval tasks in TIREx (Fröbe et al., 2023) used for evaluation. The number of tasks per corpus, number of queries, average judgments per query, and average document length are provided.

Corpus	**Tasks**		**Queries**		**Docs.**
	Details	#	Judg.	#	Length
Antique	QA Benchmark	1	32.9	200	49.9
Args.me	Touché 2020–2021	2	60.7	99	435.5
ClueWeb09	Web Tracks 2009–2012	4	421.8	200	1132.6
ClueWeb12	Web Tracks, Touché	4	163.8	200	5641.7
CORD-19	TREC-COVID	1	1386.4	50	3647.7
Cranfield	Fully Judged Corpus	1	8.2	225	234.8
Disks4+5	TREC-7/8, Robust04	3	1367.4	350	749.3
GOV	Web tracks 2002–2004	3	603.9	325	2700.5
GOV2	TREC TB 2004–2006	3	902.3	150	2410.3
MEDLINE	Genomics, PM	4	518.3	180	309.1
MS MARCO	DL 2019–2020	2	212.8	97	77.1
NFCorpus	Medical LTR Benchmark	1	48.7	325	364.6
Vaswani	Scientific Abstracts	1	22.4	93	51.3
WaPo	Core 2018	1	524.7	50	713.0

- GOV — TREC Web Tracks (Craswell and Hawking, 2002; Craswell et al., 2003; Craswell and Hawking, 2004)
- GOV2 — TREC TB (Clarke et al., 2004, 2005; Büttcher et al., 2006)
- MEDLINE — TERC Genomics, TREC Precision Medicine (Hersh et al., 2004, 2005; Roberts et al., 2017, 2018)
- MS MARCO — TREC Deep Learning (Craswell et al., 2019, 2020)
- NFCorpus — Medical LTR Benchmark (Boteva et al., 2016)
- Vaswani — Scientific Abstracts
- WaPo — Core '18

E Novelty Prompt

We add additional instructions to the original RankGPT prompt (Sun et al., 2023) (which is also used by RankZephyr (Pradeep et al., 2023b)) to also take novelty into account. The novelty prompt is shown in Figure 4.

F Relationship between Passage Length and Near-Duplicates

MS MARCO contains many instances of near-duplicates where two (or more) passages are identical, but another passage simply contains one or two additional sentences appended at the end. See Figure 5 for examples. Even without inter-passage information, a pointwise model can then learn to rank short passages at the top of the ranking and long passages at the bottom to avoid ranking near-duplicates at the top.

System: You are RankGPT, an intelligent assistant that can rank passages based on their relevancy **and novelty** to the query.

User: I will provide you with {num} passages, each indicated by number identifier [].
Rank the passages based on their relevance **and novelty** to the query: {query}.

Assistant: Okay, please provide the passages.

...

User: Search Query: {query}. Rank the {num} passages above based on their relevance to the search query. The passages should be listed in descending order using identifiers. The most relevant passages should be listed first. **For near-duplicate passages, put all but one of the passages at the bottom of the ranking.** The output format should be [] > [], e.g., [1] > [2]. Only respond with the ranking results, do not say any word or explain.

Figure 4: Novelty prompt for RankGPT and RankZephyr.

We see this behavior for the monoELECTRA model. Pearson's correlation coefficient between passage rank and length in tokens rises from a moderate 0.28 when fine-tuning with RankNet to a strong 0.73 when fine-tuning with novelty-aware RankNet (both statistically significant $p < 0.05$). Pearson's correlation coefficient also rises for the Set-Encoder, from 0.36 to 0.54 (both statistically significant $p < 0.05$), but is substantially less severe.

Query: which organelle contains the majority of the cell's genetic materials in an animal cell?
Passages:

1. Cluster
   - The nucleus is an organelle found in eukaryotic cells. Inside its fully enclosed nuclear membrane, it contains the majority of the cell's genetic material. This material is organized as DNA molecules, along with a variety of proteins, to form chromosomes.
2. Cluster
   - The cell nucleus contains the majority of the cell's genetic material in the form of multiple linear DNA molecules organized into structures called chromosomes. Each human cell contains roughly two meters of DNA.
   - The cell nucleus contains the majority of the cell's genetic material in the form of multiple linear DNA molecules organized into structures called chromosomes. Each human cell contains roughly two meters of DNA. The cell nucleus contains the majority of the cell's genetic material in the form of multiple linear DNA molecules organized into structures called chromosomes.
3. Cluster
   - The cell nucleus contains the majority of the cell's genetic material in the form of multiple linear DNA molecules organized into structures called chromosomes.Each human cell contains roughly two meters of DNA.n cell biology, the nucleus (pl. nuclei; from Latin nucleus or nuculeus, meaning kernel) is a membrane-enclosed organelle found in eukaryotic cells. Eukaryotes usually have a single nucleus, but a few cell types have no nuclei, and a few others have many.
4. Cluster
   - Nucleus. The nucleus is one of the most important organelles in a cell. It is often the largest organelle in animal cells, but this is not always the case. Nuclei contain the genetic material called DNA that is responsible for controlling and directing all cell activities. Note-although this organelle is also found animal cells, the vacuoles from plant cells are drastically bigger and play a much more important role in the processes of the cell.
   - The nucleus is one of the most important organelles in a cell. It is often the largest organelle in animal cells, but this is not always the case. Nuclei contain the genetic material called DNA that is responsible for controlling and directing all cell activities.
   - Nucleus. The nucleus is one of the most important organelles in a cell. It is often the largest organelle in animal cells, but this is not always the case. Nuclei contain the genetic material called DNA that is responsible for controlling and directing all cell activities.
   - The nucleus is one of the most important organelles in a cell. It is often the largest organelle in animal cells, but this is not always the case. Nuclei contain the genetic material called DNA that is responsible for controlling and directing all cell activities. This is a very important organelle given its vital function.

Figure 5: Example of duplicate passages in the MS MARCO dataset.