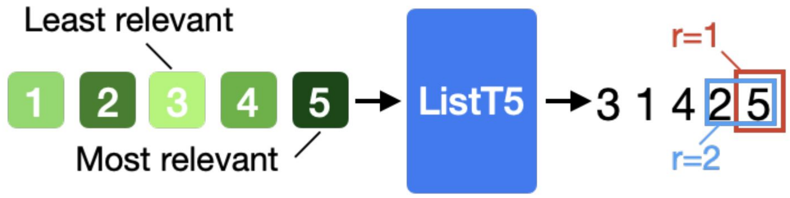
ListT5: Listwise Reranking with Fusion-in-Decoder Improves Zero-shot Retrieval

Soyoung Yoon[1*] Eunbi Choi[2] Jiyeon Kim[3] Yireun Kim[2]
Hyeongu Yun[2] Seung-won Hwang[1]
[1]Seoul National University [2]LG AI Research [3]KAIST AI

Overview



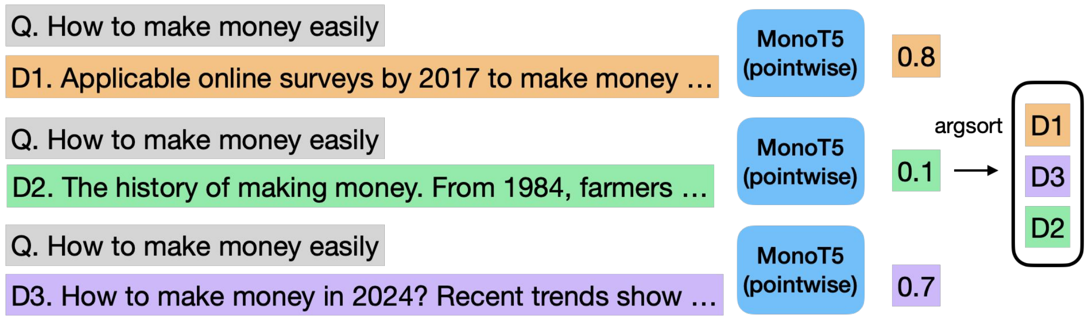
We introduce ListT5, FiD with tournament sort, that is..

1. Computationally efficient.
   a. Faster than pairwise or LLM + sliding window based listwise methods
   b. comparable with pointwise methods
2. Robust to positional bias.
   a. Overcomes the lost-in-the middle problem by FiD, with each passage encoded with identical positional encoding.
3. Shows great zero-shot performance.
   a. superior than any listwise, pointwise, pairwise models on BEIR benchmark, for T5-base and T5-3B with relatively small size.

Background

- Models still struggle on zero-shot retrieval
- Listwise reranking models are shown to be effective on zero-shot retrieval, but previous listwise reranking models had limitations
    - small-sized models only implement pairwise reranking with impractical efficiency (e.g., DuoT5)
    - large-sized models suffer from the lost-in-the middle problem due to its long input length.

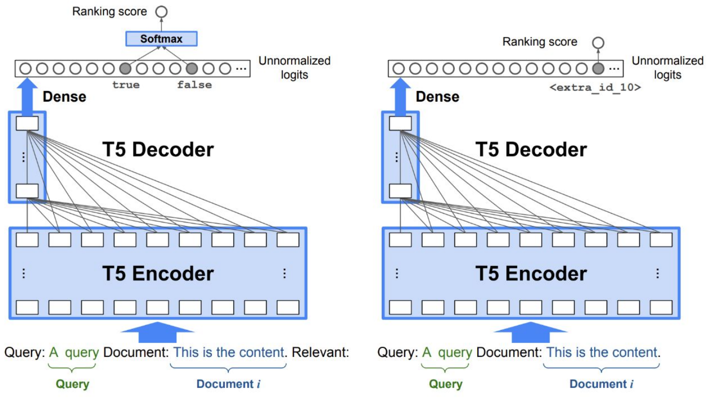
Pointwise v.s. Listwise Reranking



Pointwise reranking with MonoT5

Listwise reranking with ListT5

(pointwise rerankers)



(a) monoT5 (Nogueira et al., 2020) (b) RankT5 (Encoder-Decoder)

- Listwise rerankers can condition on and compare multiple passages to calibrate the relevance scores better, thus reducing the inaccuracy of predictions arising from domain shift.*

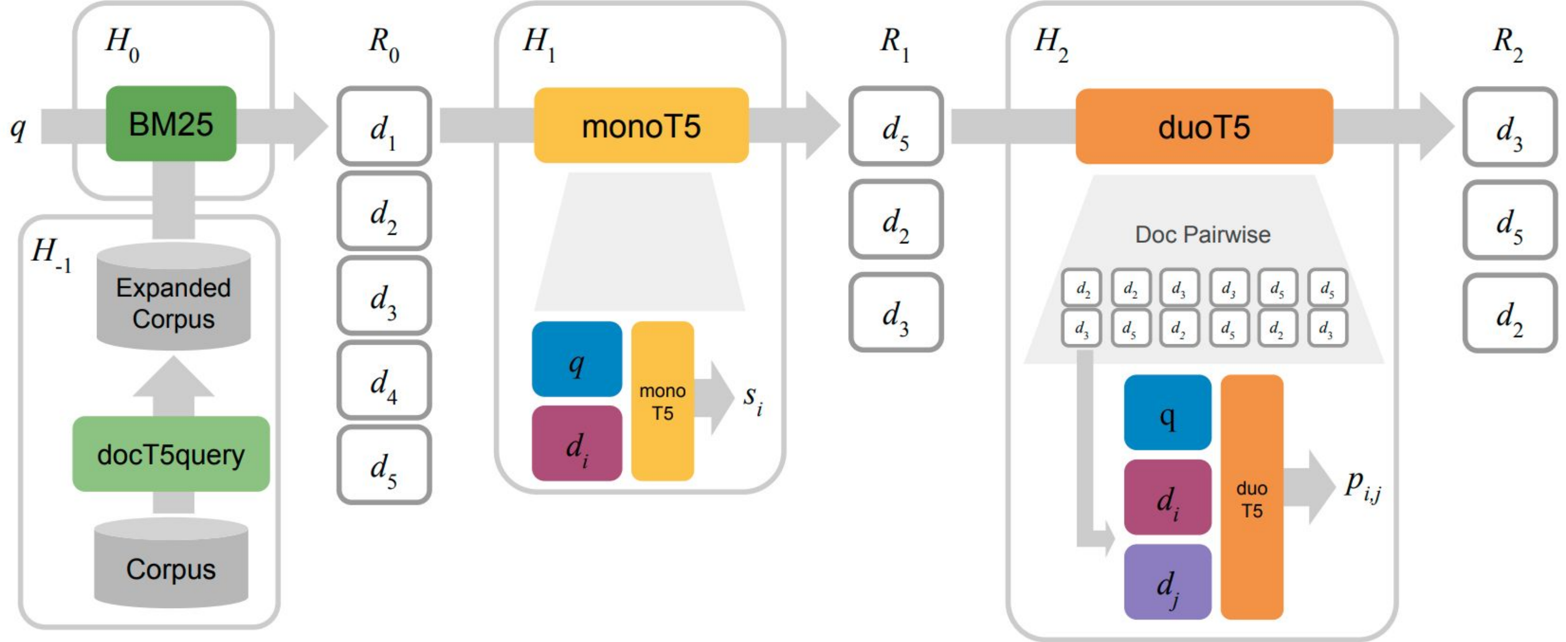
*Xian et al., Learning List-Level Domain-Invariant Representations for Ranking

monoT5 and RankT5 image borrowed from: Zhuang et al, RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses

Listwise reranking models: Baselines

- **Listwise Baseline models: DuoT5 for small models**

DuoT5: better performance than pointwise models, but n^2 time complexity!



Pradeep et al., The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models

Listwise reranking models: Baselines

- **Listwise Baseline models: RankVicuna, RankZephyr, RankGPT for Large Language Models**

```
USER: I will provide you with {num} passages, each
indicated by a numerical identifier []. Rank the
passages based on their relevance to the search
query: {query}.

[1] {passage 1}
[2] {passage 2}
...
[{num}] {passage {num}}

Search Query: {query}.

Rank the {num} passages above based on their
relevance to the search query. All the passages
should be included and listed using identifiers, in
descending order of relevance. The output format
should be [] > [], e.g., [4] > [2]. Only respond
with the ranking results, do not say any word
or explain.
```

Recently, methods to do listwise reranking with LLMs has been investigated

```
The following are passages related to query {{query}}
[1] {{passage_1}}
[2] {{passage_2}}
(more passages)
Rank these passages based on their relevance to the query.
```

```
[2] > [3] > [1] > [...]
```

(c) Permutation generation

But, inefficiency due to large parametric size of the model & **lost-in-the middle** problem occurs!

Sun et al., Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent

Listwise reranking models: Baselines

- Crucial problem in Listwise reranking with LLMs: **Lost in the middle problem**

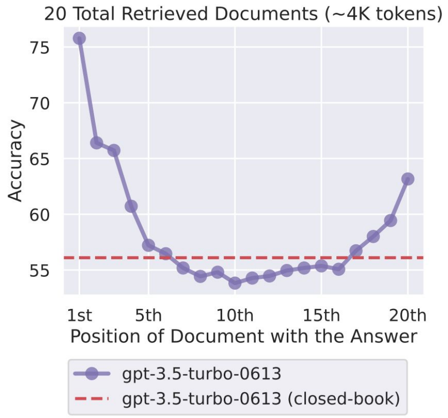
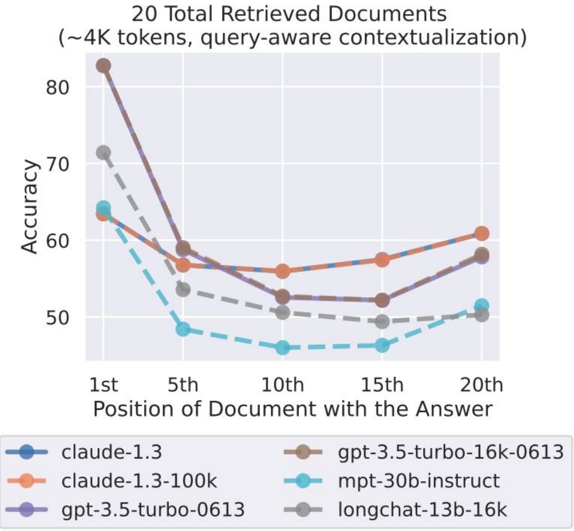
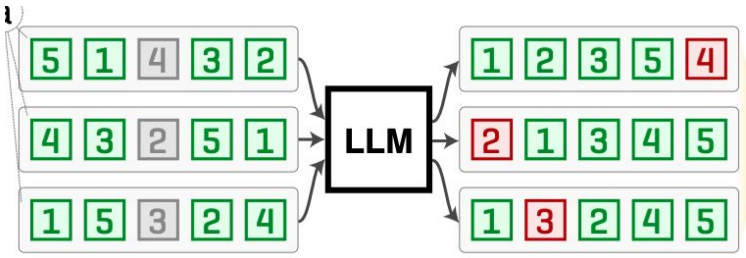


Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning (primacy bias) or end of its input context (recency bias), and performance degrades significantly when models must access and use information located in the middle of its input context.





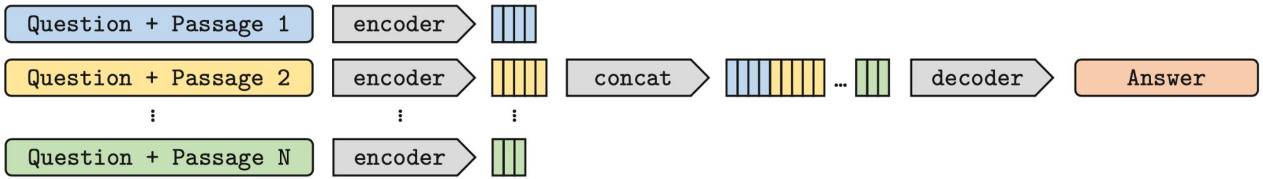
Liu et al., Lost in the Middle: How Language Models Use Long Contexts

Tang et al., Found in the Middle: Permutation Self-Consistency Improves Listwise Reranking in Language Models

Listwise reranking: Solutions

- listwise reranking is effective for zero-shot retrieval
- How to better utilize the autoregressive generation ability of reranking models?
- Small models can't see long context, pairwise models are impractical, efficiency hurts with lengthy inputs
- listwise reranking with LLMs has the lost-in-the-middle problem
- How can we train the model to efficiently see **multiple passages at once** and do **listwise ranking**, while being fairly efficient and exhibit less positional bias?

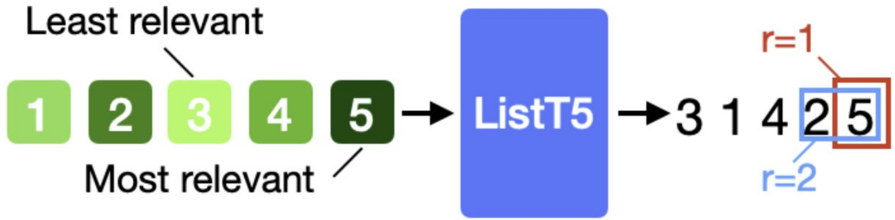
-> use FiD (Fusion-in-Decoder) architecture that outputs sorted passage index, and use (hierarchical) tournament sort to cache already-computed passages!

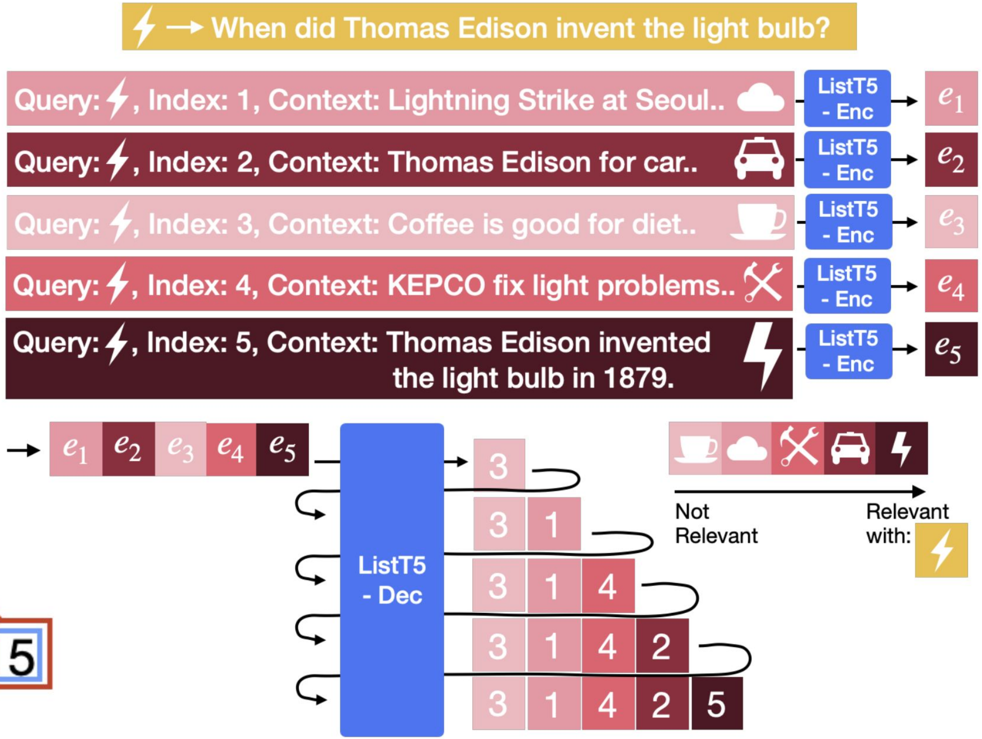


izacard et al., Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering

Proposed Method: ListT5 architecture

- Fusion-in-Decoder, that given k (=5) contexts, output sorted index, with relevant index coming at the last.
- Each passage is processed by the encoder with identical positional encodings, and the model identifies each passage with index number: CANNOT exploit positional bias.





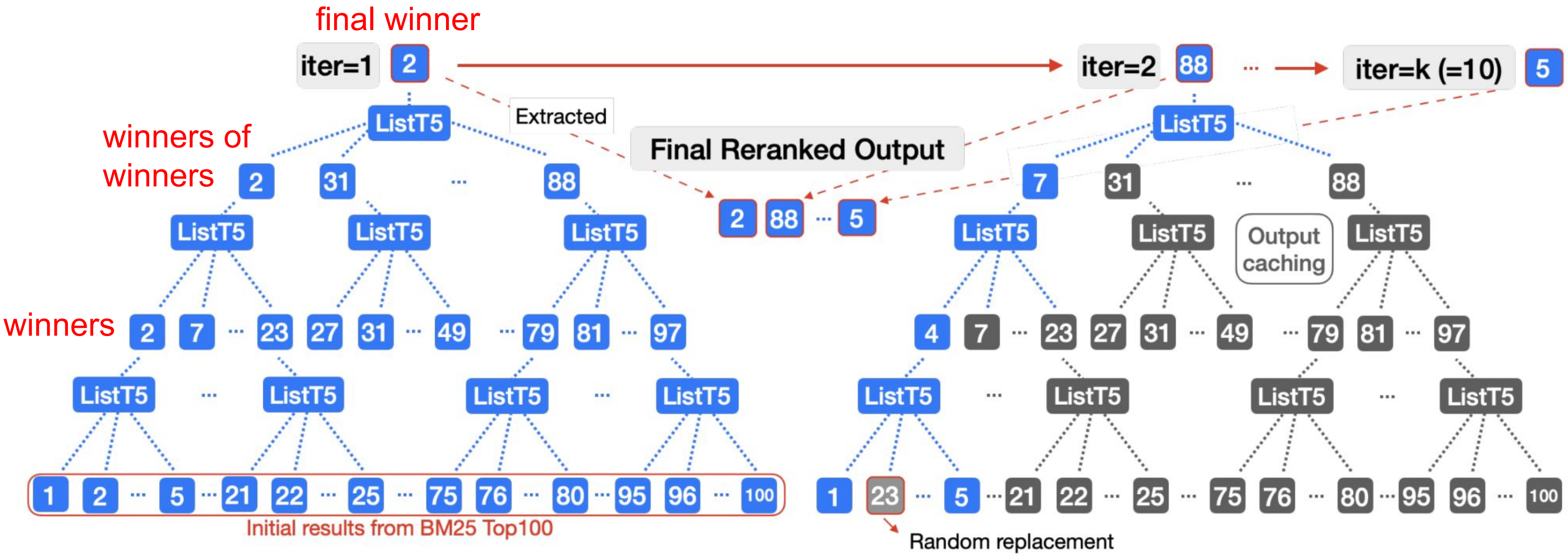
Making the train dataset

- Source: MSMARCO, with only positive/negative labels
- Labeling ordering between negatives
    - Used bi-encoder (coco-dr large: 340M, GTR-large for ablation), selected top-1000 out of 8.8M corpus, random selection of 4 negatives
    - labeled negative scoring by the dot product scores of the bi-encoder

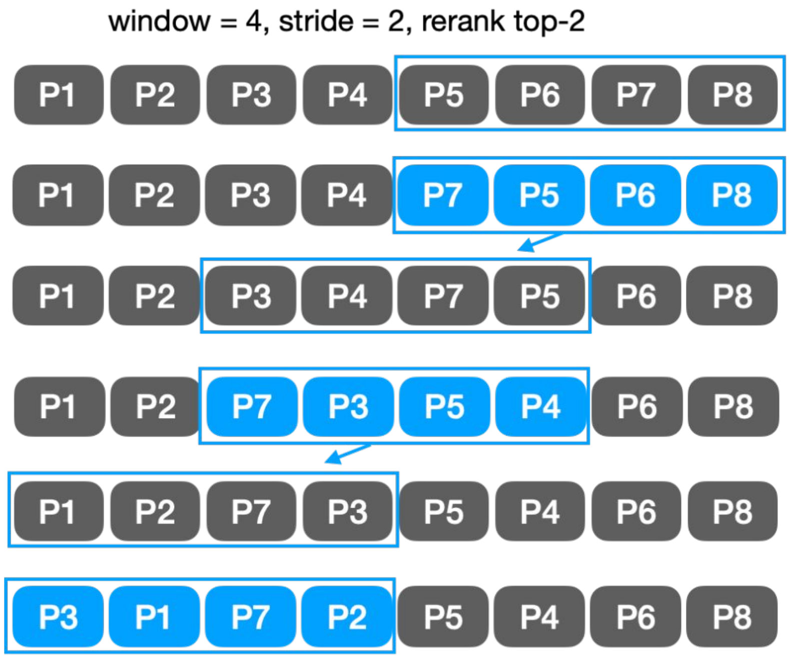
*training data open-sourced upon request at
https://huggingface.co/datasets/Soyoung97/ListT5-train-data

Sorting method: Tournament sort

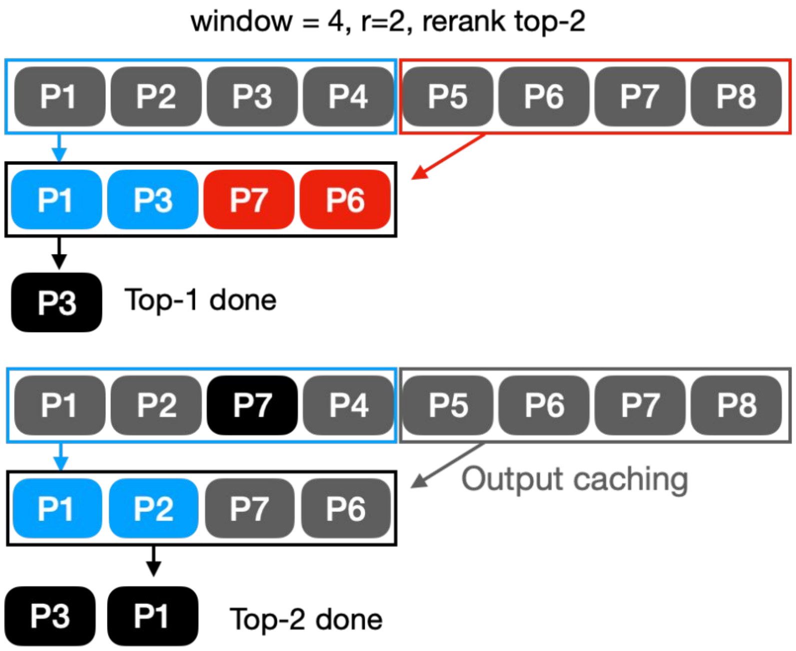
- 5-ary tournament tree, with output caching
- group passages to 5, ranks them hierarchically like tournament
- getting next top-1 only requires computing path for changed elements



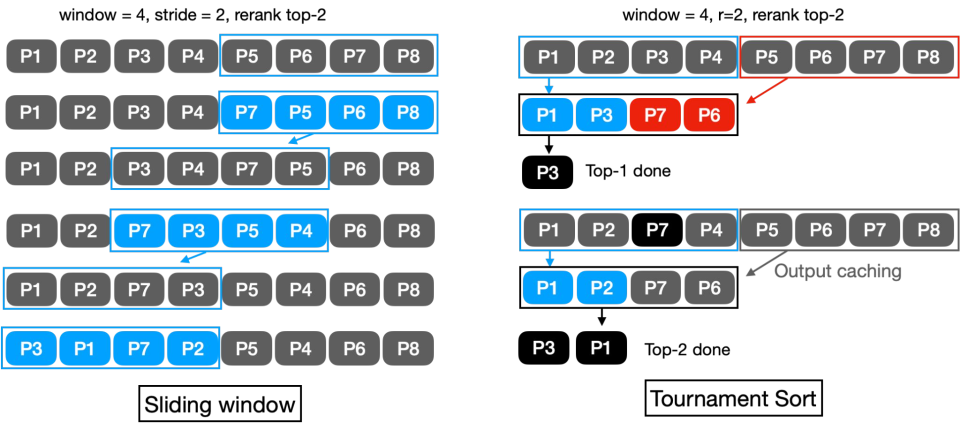
Tournament sort v.s. Sliding window



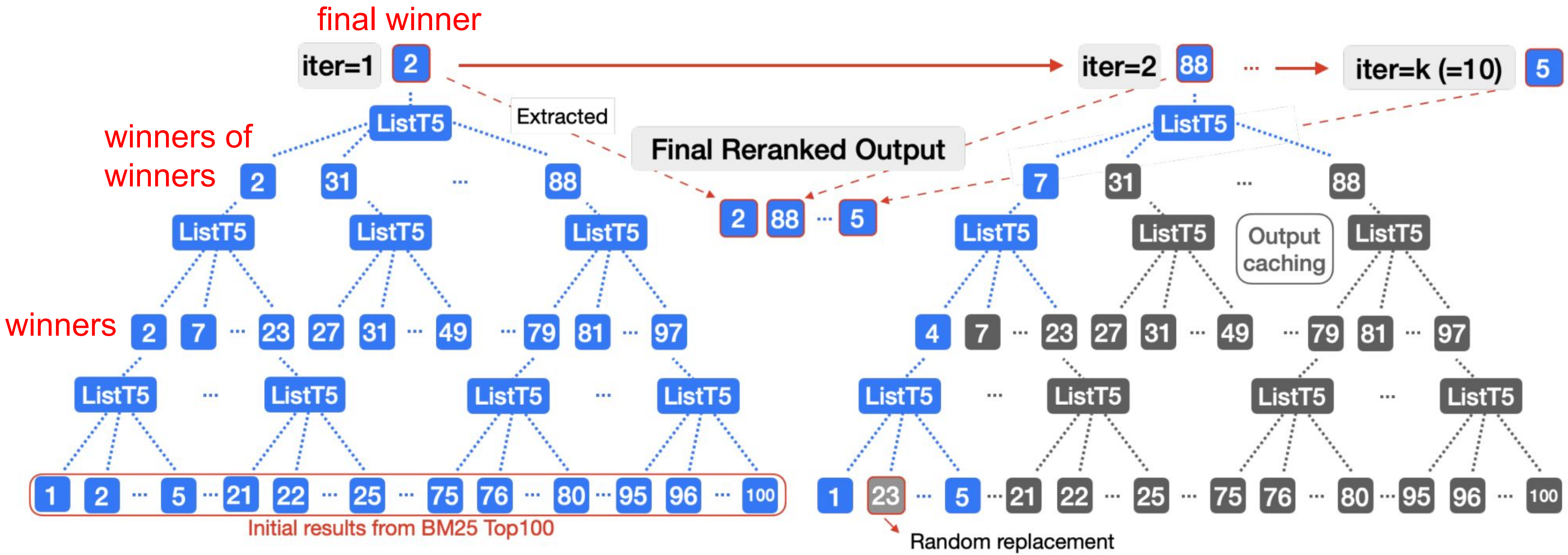
Sliding window



Tournament Sort



Method Name	Reranking Method	Complexity
MonoT5 (Nogueira et al., 2020), RankT5 (Zhuang et al., 2022)	Pointwise	$\mathcal{O}(n)$
DuoT5 (Pradeep et al., 2021)	Pairwise	$\mathcal{O}(n^2)$
ListT5 (Ours)	Listwise	$\mathcal{O}(n + k \log n)$



Zero-shot performance - pointwise baseline models

	BM25 Top-100							BM25 Top-1000		
	Initial	MonoT5 -base	RankT5 -base	ListT5 -base (r=2)	MonoT5 -3B	RankT5 -3B	ListT5 -3B (r=2)	MonoT5 -base	RankT5 -base	ListT5 -base (r=2)
TREC-COVID	59.5	**78.3**	77.7	**78.3**	79.8	81.7	**84.7**	78.3	79.1	**82.1**
NFCorpus	32.2	**35.7**	35.1	35.6	37.3	**37.4**	37.7	**36.1**	35.3	**36.1**
BioASQ	52.2	55.3	**58.2**	56.4	57.5	**58.3**	**58.3**	52.6	**57.6**	55.0
NQ	30.5	52.1	**53.2**	53.1	56.4	**57.8**	56.2	55.9	**57.6**	57.5
HotpotQA	63.3	71.2	**72.8**	72.6	74.3	74.8	**75.6**	70.9	**73.8**	73.6
FiQA-2018	23.6	39.2	39.2	**39.6**	**46.0**	45.2	45.1	41.2	41.1	**41.8**
Signal-1M (RT)	33.0	32.0	30.8	**33.5**	32.2	31.9	**33.8**	29.3	28.6	**30.9**
TREC-NEWS	39.5	48.0	45.4	48.5	48.3	49.5	**53.2**	47.8	45.9	**50.9**
Robust04	40.7	53.4	**54.3**	52.1	**58.5**	58.3	57.8	55.4	**57.2**	54.7
Arguana	40.8	34.4	35.5	**48.9**	46.8	37.4	**50.6**	24.2	26.6	**46.9**
Touche-2020	44.2	29.6	**37.1**	33.4	32.5	**38.8**	33.6	26.4	**37.0**	31.5
CQADupStack	30.0	38.6	37.0	**38.8**	41.3	40.3	**42.1**	40.1	38.1	**40.5**
Quora	78.9	84.6	83.3	**86.4**	84.0	83.6	**86.9**	84.2	82.9	**86.4**
DBPedia	31.8	42.8	43.7	43.7	44.8	**45.0**	46.2	43.1	**45.1**	44.9
SCIDOCS	14.9	16.7	16.8	**17.6**	19.0	18.9	**19.5**	17.0	17.1	**18.0**
FEVER	65.2	78.4	77.6	**79.8**	80.0	79.8	**82.0**	77.9	77.8	**81.0**
Climate-FEVER	16.5	23.1	21.2	**24.0**	**26.2**	24.5	24.8	23.3	20.6	**24.9**
SciFact	67.9	73.1	73.5	**74.1**	76.3	**77.1**	77.0	73.3	73.6	**74.9**
Average	42.5	49.3	49.6	**50.9**	52.3	52.2	**53.6**	48.7	49.7	**51.8**

	COCO-DR Large (Init.)	MonoT5	RankT5	ListT5 (r=1)	ListT5 (r=2)
MSMARCO Top-1000 (in-domain)	41.9	43.1	46.2	46.1	**46.3**
TREC-COVID	80.8	**83.5**	**83.5**	83.2	**83.5**
NFCorpus	35.5	35.6	35.5	**36.2**	**36.2**
NQ	54.3	57.9	59.6	59.7	**60.0**
HotpotQA	63.3	68.7	**71.1**	70.3	70.9
FiQA-2018	32.3	41.2	41.3	**41.7**	**41.7**
Arguana	46.9	33.0	34.8	49.0	**49.3**
Touche-2020	21.6	25.7	**35.7**	29.1	29.6
CQADupStack	37.3	40.5	38.7	40.7	**40.9**
Quora	**87.3**	84.0	83.0	86.2	86.3
DBPedia	40.7	44.4	**46.1**	45.6	45.4
SCIDOCS	17.3	17.5	17.5	17.7	**18.3**
FEVER	74.9	78.9	79.7	79.8	**81.4**
Climate-FEVER	23.1	24.2	22.9	23.9	**24.9**
SciFact	71.9	73.5	73.6	**74.4**	74.3
Avg. BEIR	49.1	50.6	51.6	52.7	**53.1**

Zero-shot performance - listwise baseline models

	TREC-DL19	TREC-DL20	TREC-COVID	NFC-orpus	Signal-1M (RT)	TREC-NEWS	Robu-st 04	Touche-2020	DBP-edia	Sci-Fact	Avg (In-domain)	Avg (BeIR)
DuoT5-base	71.4	67.4	**80.1**	35.0	31.4	**49.1**	49.6	31.8	**43.9**	69.6	69.4	48.8
LISTT5-base ($r = 2$)	**71.8**	**68.1**	78.3	**35.6**	**33.5**	48.5	**52.1**	**33.4**	43.7	**74.1**	**70.0**	**49.9**
RankGPT (GPT3.5)	65.8	62.9	76.7	35.6	32.1	48.9	50.6	**36.2**	44.5	70.4	64.4	49.4
RankVicuna-7b	68.9	66.1	80.5	33.2	**34.2**	46.9	48.9	33.0	44.4	70.8	67.5	49.0
RankZephyr-7b	**73.9**	**70.9**	84.0	36.7	31.8	52.6	54.3	33.8	44.6	74.9	**72.4**	51.6
LISTT5-3B ($r = 2$)	71.8	69.1	**84.7**	**37.7**	33.8	**53.2**	**57.8**	33.6	**46.2**	**77.0**	70.5	**53.0**

Efficiency

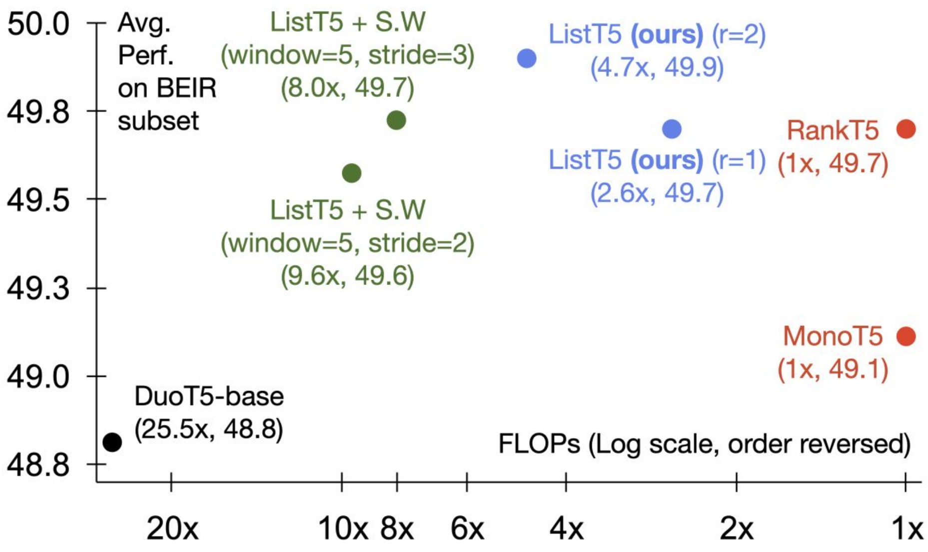
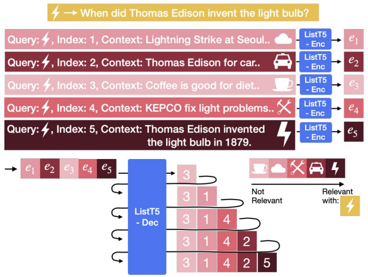


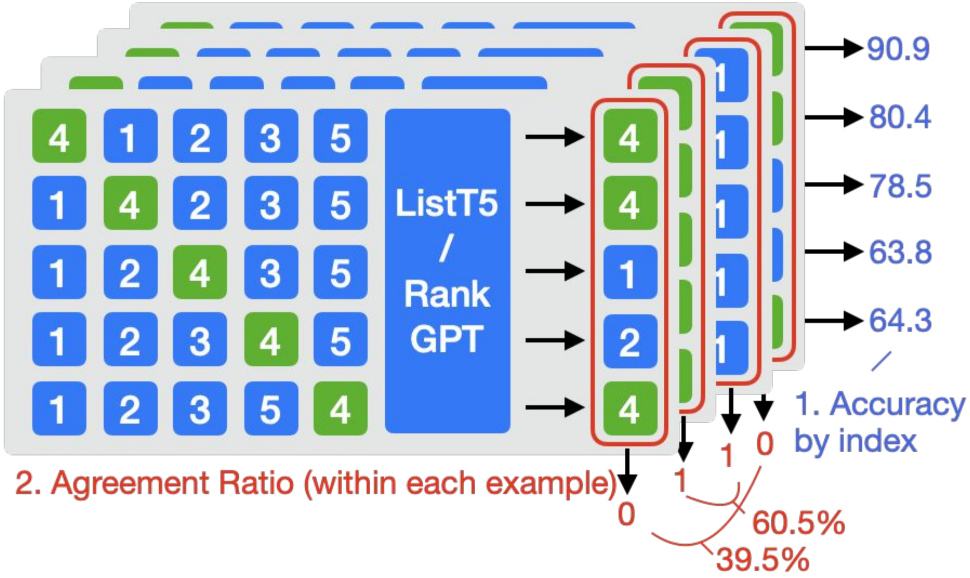
Figure 4: Real-time FLOPs comparison of the models on T5-base, including DuoT5 and the sliding window variants of LISTT5. The reported BEIR performance is averaged from a subset of BEIR, same as in Tab. 3.

Positional Invariance

```
I will provide you with 5 passages,
each indicated by numerical identifier [].
Rank the passages based on their
relevance to the search query: {query}.

[1] {passage_1}
[2] {passage_2}
[3] {passage_3}
[4] {passage_4}
[5] {passage_5}

Search Query: {query}

Rank the 5 passages above based on their
relevance to the search query.
All the passages should be included and
listed using identifiers, in descending
order of relevance. The output format
should be [] > [], e.g., [4] > [2].
Only respond with the ranking
results, do not say any word or explain.
```

v.s.





Positional Invariance - better than RankGPT4!

	TREC-COVID							FiQA						
	Accuracy when positive passage is at index # :						Aggrement ratio (↑)	Accuracy when positive passage is at index #:						Aggrement ratio (↑)
	1	2	3	4	5	Std. (↓)		1	2	3	4	5	Std. (↓)	
GPT-3.5	81.6	63.3	75.5	67.3	61.2	7.7	55.1	88.3	68.1	78.7	65.9	75.8	8.0	62.1
GPT-4	95.9	83.7	73.5	77.6	71.4	8.8	69.4	94.6	90.5	84.4	86.8	84.8	3.9	82.8
DuoT5	91.3	76.0	-	-	-	7.6	79.6	89.9	76.9	-	-	-	6.5	78.1
LISTT5	93.9	87.8	83.7	85.7	81.6	**4.2**	**83.7**	85.3	85.6	82.2	83.3	82.6	**1.4**	**90.4**

Table 4: Robustness to the position of the positive passage in the input, on TREC-COVID and FiQA. GPT-3.5, GPT-4, DuoT5, and LISTT5 stands for GPT-3.5-turbo-1106, GPT-4-0613, DuoT5-base, and LISTT5-base, respectively. Using the FiD structure effectively mitigates the problem of the positional bias of positive passages, showing lowest standard deviation and highest agreement ratio.

Positional Invariance - robustness to shuffling candidate passages

Initial ordering	DL19	DL20	TREC-COVID	TREC-NEWS	Touche -2020	Avg.
ListT5-base (tournament sort, r=2)						
No shuffle	71.8	68.1	78.3	48.5	33.4	60.0
Shuffle	71.2	68.1	77.2	48.9	32.8	59.6
Perf. drop						**-0.4**
ListT5-base (sliding windows, stride=3, iter=4)						
No shuffle	71.8	67.7	77.5	50.0	33.1	60.0
Shuffle	69.5	65.5	77.7	49.2	32.1	58.8
Perf. drop						**-1.2**
RankVicuna-7b (sliding windows)						
No shuffle	68.9	66.1	80.5	46.9	33.0	59.1
Shuffle	67.1	64.6	79.2	45.3	30.8	57.4
Perf. drop						**-1.7**
RankGPT-3.5 (sliding windows)						
No shuffle	68.4	64.9	72.6	46.5	38.2	58.1
Shuffle	62.5	57.0	66.1	38.3	22.8	49.3
Perf. drop						**-8.8**

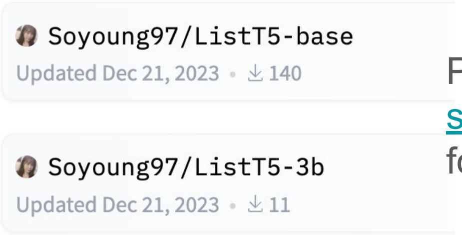
Summary

Paper

Code

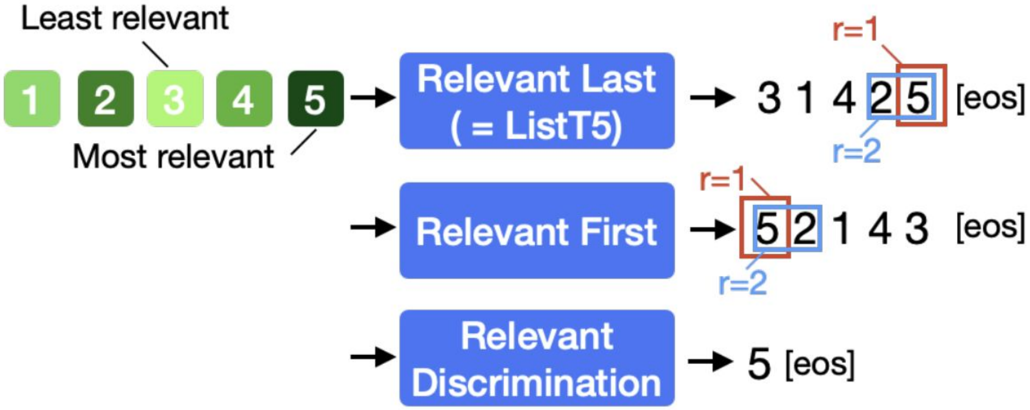
We introduce ListT5 with tournament sort, that is..

1. Computationally efficient.
    a. Lower than pairwise or LLM + sliding window based listwise methods
    b. comparable with pointwise methods
2. Robust to positional bias.
    a. Overcomes the lost-in-the-middle problem by FiD, with each passage encoded with identical positional encoding.
3. Shows great zero-shot performance.
    a. superior than any listwise, pointwise, pairwise models on BEIR benchmark, for T5-base and T5-3B with relatively small size



Please contact: soyoung.yoon@snu.ac.kr for any questions!

[Ablation] Model variants



- Most effective to generate most relevant index at the LAST!
- Sequential generation like reasoning chain!

Dataset	Relevant Discrimi-nation	Relevant First $(r=1)$	Relevant First $(r=2)$	Relevant Last (ListT5) $(r=1)$	Relevant Last (ListT5) $(r=2)$
In-domain					
MS MARCO	40.3	40.8	**40.9**	40.7	40.7
TREC-DL19	**72.5**	69.6	70.8	71.2	71.8
TREC-DL20	67.3	67.0	66.8	67.3	**68.1**
Avg (in-domain)	60.0	59.1	59.5	59.7	**60.2**
Out-domain (BEIR)					
TREC-COVID	74.0	74.9	75.9	76.7	**78.3**
NFCorpus	34.8	35.5	35.6	35.5	**35.6**
BioASQ	55.8	56.6	56.6	**57.2**	56.4
NQ	51.1	52.7	52.9	52.0	**53.1**
HotpotQA	70.9	72.5	72.6	72.1	**72.6**
FiQA-2018	38.1	39.3	39.0	39.5	**39.6**
Signal-1M (RT)	32.9	31.8	31.7	33.3	**33.5**
TREC-NEWS	43.9	46.6	47.3	47.9	**48.5**
Robust04	49.8	**52.3**	**52.3**	52.0	52.1
Arguana	26.1	32.8	34.6	**49.7**	48.9
Touche-2020	**34.2**	31.5	31.3	**34.2**	33.4
CQADupStack	**38.8**	38.3	38.4	38.4	**38.8**
Quora	81.9	84.4	84.8	86.1	**86.4**
DBPedia	42.4	43.4	43.6	**43.9**	43.7
SCIDOCS	16.3	17.3	17.3	17.2	**17.6**
FEVER	77.6	77.4	77.7	77.8	**79.8**
Climate-FEVER	20.7	22.8	23.0	22.8	**24.0**
SciFact	73.0	74.1	**74.2**	74.1	74.1
Avg (BEIR)	47.9	49.1	49.4	50.6	**50.9**

Appendix - design choice

	$m = 5$		$m = 10$	
	1/5 (ListT5, r=1)	2/5 (ListT5, r=2)	1/10 (r = 1)	4/10 (r = 4)
MS MARCO	**40.7**	**40.7**	40.5	**40.7**
+ Top-1000	44.7	44.9	44.6	**45.0**
TREC-DL19	71.2	**71.8**	70.1	70.5
TREC-DL20	67.3	**68.1**	66.9	67.2
TREC-COVID	76.7	**78.3**	76.2	77.9
NFCorpus	35.5	35.6	36.2	**36.6**
BioASQ	**57.2**	56.4	55.4	56.4
NQ	52.0	**53.1**	51.5	52.5
HotpotQA	72.1	**72.6**	71.4	71.9
FiQA-2018	39.5	**39.6**	39.0	38.9
Signal-1M (RT)	33.3	**33.5**	31.7	32.0
TREC-NEWS	47.9	**48.5**	47.1	47.8
Robust04	52.0	52.1	52.2	**53.1**
Arguana	**49.7**	48.9	38.6	46.6
Touche-2020	**34.2**	33.4	32.4	32.7
CQADupStack	38.4	**38.8**	38.2	28.8
Quora	86.1	86.4	85.5	**86.8**
DBPedia	43.9	**43.7**	42.7	43.6
SCIDOCS	17.2	17.6	17.2	**18.0**
FEVER	77.8	**79.8**	76.7	79.1
Climate-FEVER	22.8	**24.0**	22.7	23.9
SciFact	74.1	74.1	73.4	**74.2**
Avg(In-domain)	56.0	**56.4**	55.5	55.9
Avg(BeIR)	50.6	**50.9**	49.3	50.0

Appendix - LLM consistency

	GPT-3.5-turbo-1106				ListT5
	Trial 1	Trial 2	Trial 3	Avg.	-base
(1) Accuracy when the gold passage is at index #:					
1	81.6	79.6	81.6	**81.0**	**93.9**
2	63.3	63.3	61.2	**62.6**	**87.8**
3	75.5	75.5	75.5	**75.5**	**83.7**
4	67.3	63.3	67.3	**66.0**	**85.7**
5	61.2	63.3	65.3	**63.3**	**81.6**
std	7.68	7.1	7.4	**7.4**	**4.2**
(2) Agreement ratio (%) within index change of positive					
points to same passage	55.1	55.1	55.1	**55.1**	**83.7**
other	44.9	44.9	44.9	44.9	16.3

Table 10: Measuring the LLM consistency on TREC-COVID.

Appendix - Training Dataset

Model	RankT5	ListT5			
Training data	GTR	COCO-DR	GTR		
Learning Rate	1.00E-04	1.00E-04	1.00E-04	1.00E-04	1.00E-05
Steps	-	20k	20k	10k	30k
TREC-COVID	77.7	78.3	77.3	78.6	77.9
NFCorpus	35.1	35.6	35.4	36.2	35.9
BioASQ	58.2	56.4	54.9	55.1	56.8
NQ	53.2	53.1	52.7	52.8	53.2
HotpotQA	72.8	72.6	72.1	71.9	72.1
FiQA-2018	39.2	39.6	39.1	39.9	39.4
Signal-1M (RT)	30.8	33.5	34.1	32.9	30.9
TREC-NEWS	45.4	48.5	47.6	48.0	48.3
Robust04	54.3	52.1	52.9	52.7	53.6
Arguana	35.5	48.9	43.3	43.6	43.7
Touche-2020	37.1	33.4	31.5	32.7	32.5
CQADupStack	37.0	38.8	38.6	38.5	38.8
Quora	83.3	86.4	86.0	83.9	84.4
DBPedia	43.7	43.7	43.8	43.6	44.5
SCIDOCS	16.8	17.6	17.1	17.1	17.6
FEVER	77.6	79.8	78.9	79.4	79.7
Climate-FEVER	21.2	24.0	24.0	24.8	24.6
SciFact	73.5	74.1	74.0	73.1	74.4
Avg.	49.6	**50.9**	50.2	50.3	**50.5**

Table 7: Comparison of ListT5 models trained with GTR and COCO-DR. For ListT5, the reported scores are evaluated using the (r=2) variant.

Appendix : sliding window v.s. tournament sort

Sorting method	# required forward passes to rerank top1	# required forward passes to rerank top10
sliding window, stride=1	$1 + \lceil(100\text{-}5)/1\rceil = 96$	$96 \times \lceil 10/4 \rceil = 288$
sliding window, stride=2	$1 + \lceil(100\text{-}5)/2\rceil = 49$	$49 \times \lceil 10/3 \rceil = 196$
sliding window, stride=3	$1 + \lceil(100\text{-}5)/3\rceil = 33$	$33 \times \lceil 10/2 \rceil = 165$
sliding window, stride=4	$1 + \lceil(100\text{-}5)/4\rceil = \mathbf{25}$	$25 \times \lceil 10/1 \rceil = 250$
tournament sort, r=1	$(100/5) + (20/5) + 1 = \mathbf{25}$	$25 + 9$ x $(1{+}1{+}1) = \mathbf{52}$
tournament sort, r=2	$(100/5) + (40/5) + 2 + 1 = 31$	$31 + 9 \times (1{+}1{+}1{+}1) = 67$

Table 12: Number of forward passes to rerank top-k candidates from 100 candidate passages per one query, where window size w=5. In the case of reranking top-10 passages, tournament sort requires much more fewer number of forward passes.

Idx	Base Model	Sorting method	Name	FLOPs to rerank: Top-1	Top-10
0	T5-base	pointwise	MonoT5	1x	1x
1	T5-base	tournament	ListT5(r=1)	1.3x	2.6x
2	T5-base	tournament	ListT5(r=2)	1.8x	4.7x
3	T5-base	sliding w.(s=2)	T5(FiD)	2.5x	9.8x
4	T5-base	sliding w.(s=3)	T5(FiD)	1.7x	12.3x
5	T5-3b	tournament	ListT5(r=1)	17.6x	36.3x
6	T5-3b	tournament	ListT5(r=2)	24.6x	66.0x
7	T5-3b	sliding w.(s=2)	T5(FiD)	38.5x	154x
8	T5-3b	sliding w.(s=2)	T5(no FiD)	53.8x	215.1x
9	T5-3b	sliding w.(s=3)	T5(FiD)	25.6x	128x
10	T5-3b	sliding w.(s=3)	T5(no FiD)	35.1x	175.6x

Table 13: FLOPs (In a multiple of FLOPs of MonoT5-base) on the choice of architecture and method, on TREC-DL19. For the sliding window approach, we would need a total of 4 multiple passes for stride = 3 and 5 passes for stride = 2 (Explained at Tab. 12) to rerank Top-10 candidates.

Appendix : sliding window v.s. tournament sort

	Rerank Top-10 (NDCG@10)				Rerank Top-1 (NDCG@1)			
Sorting Method	T.S.		S.W		T.S.		S.W.	
Hyperparam.	r=1	r=2	s=2 (iter=5)	s=3 (iter=4)	r=1	r=2	s=2 (iter=1)	s=3 (iter=1)
FLOPS(DL19)	**1x**	1.8x	3.7x	3.1x	**1x**	1.4x	1.96x	1.32x
DL19	71.2	71.8	71.5	71.8	81.0	79.1	81.0	78.7
DL20	67.3	68.1	67.3	67.7	77.8	77.8	79.0	79.6
In-domain avg.	69.3	**70.0**	69.4	69.8	79.4	78.5	**80.0**	79.2
TREC-COVID	76.7	78.3	78.9	77.5	88.0	91.0	88.0	86.0
NFCorpus	35.5	35.6	35.3	35.5	47.8	49.2	48.6	48.6
BioASQ	57.2	56.4	54.5	54.9	59.2	58.4	55.8	57.2
NQ	52.0	53.1	52.7	52.8	36.0	37.6	36.4	36.6
HotpotQA	72.1	72.6	71.2	71.6	83.3	84.1	83.1	83.1
FiQA-2018	39.5	39.6	39.7	39.8	41.2	40.7	41.4	41.5
Signal-1M (RT)	33.3	33.5	32.4	33.2	43.3	41.8	42.3	41.8
TREC-NEWS	47.9	48.5	49.8	50.0	53.2	54.1	52.3	52.9
Robust04	52.0	52.1	51.3	51.7	65.1	66.3	67.1	65.9
Arguana	49.7	48.9	47.7	47.8	25.8	23.9	23.3	22.7
Touche-2020	34.2	33.4	32.7	33.1	34.7	31.6	36.7	36.7
CQADupStack	38.4	38.8	38.9	38.8	31.6	31.9	32.1	32.0
Quora	86.1	86.4	86.3	86.2	77.8	77.8	78.1	77.7
DBPedia	43.9	43.7	42.6	43.2	55.5	56.5	55.1	56.6
SCIDOCS	17.2	17.6	17.9	17.7	21.9	22.0	22.8	21.4
FEVER	77.8	79.8	79.3	79.3	69.4	72.4	70.2	70.4
Climate-FEVER	22.8	24.0	23.8	23.7	20.2	23.3	20.4	21.0
SciFact	74.1	74.1	73.6	73.5	65.0	65.3	65.3	65.7
BEIR avg.	50.6	**50.9**	50.5	50.6	51.1	**51.6**	51.1	51.0

Idx	Base Model	Sorting method	Name	FLOPs to rerank: Top-1	FLOPs to rerank: Top-10
0	T5-base	pointwise	MonoT5	1x	1x
1	T5-base	tournament	ListT5(r=1)	1.3x	2.6x
2	T5-base	tournament	ListT5(r=2)	1.8x	4.7x
3	T5-base	sliding w.(s=2)	T5(FiD)	2.5x	9.8x
4	T5-base	sliding w.(s=3)	T5(FiD)	1.7x	12.3x
5	T5-3b	tournament	ListT5(r=1)	17.6x	36.3x
6	T5-3b	tournament	ListT5(r=2)	24.6x	66.0x
7	T5-3b	sliding w.(s=2)	T5(FiD)	38.5x	154x
8	T5-3b	sliding w.(s=2)	T5(no FiD)	53.8x	215.1x
9	T5-3b	sliding w.(s=3)	T5(FiD)	25.6x	128x
10	T5-3b	sliding w.(s=3)	T5(no FiD)	35.1x	175.6x

Table 13: FLOPs (In a multiple of FLOPs of MonoT5-base) on the choice of architecture and method, on TREC-DL19. For the sliding window approach, we would need a total of 4 multiple passes for stride = 3 and 5 passes for stride = 2 (Explained at Tab. 12) to rerank Top-10 candidates.

Appendix: applying tournament sort on RankGPT

Method	dl19	dl20	trec-covid	news	touche
sliding	68.4 ± 0.4	64.9 ± 1.1	72.6 ± 1.4	46.5 ± 1.0	**38.2 ± 0.5**
tournament	67.4 ± 0.9	65.8 ± 0.6	**76.4 ± 0.4**	45.5 ± 1.0	33.1 ± 1.7

Table 18: NDCG@10 on the selected subset of BEIR, on RankGPT-3.5 with different sorting methods. For fair comparison, we used w = 20, s = 10 for the sliding approach, and m = 20, r = 10 for the tournament sort. To compensate for the instability of APIs, all results are run for 3 times. Except for trec-covid and touche, differences are statistically non-significant ($p > 0.1$). (Sec. K.2)

Appendix: Train Dataset Example

```
input: Query: did edison invent the car battery?, Index: 1, Context: Ransome Eli Olds. The first automobile to be mass produced in the United States was the 1901, Curved Dash Oldsmobile, built by the American car manufacturer Ransome Eli Olds (1864-1950). Olds invented the basic concept of the assembly line and started the Detroit area automobile industry.He first began making steam and gasoline engines with his father, Pliny Fisk Olds, in Lansing, Michigan in 1885. Olds designed his first steam-powered car in 1887.ansome Eli Olds. The first automobile to be mass produced in the United States was the 1901, Curved Dash Oldsmobile, built by the American car manufacturer Ransome Eli Olds (1864-1950). Olds invented the basic concept of the assembly line and started the Detroit area automobile industry.
>>> Query: did edison invent the car battery?, Index: 2, Context: Correction, Thomas Edison did not originally invented the phonograph, another one invented it before him. His name is Emile Berliner. Edison is known for his cunning ways, he is used to steal others work and make it his own so he got the credit. The same thing he did to the brilliant Nikola Tesla.
>>> Query: did edison invent the car battery?, Index: 3, Context: When Daimler-Benz (makers of Mercedes-Benz cars) says that the automobile was invented in 1886 by Karl Benz and Gottlieb Daimler, it's basing its claim on its own definition: a light carriage for personal transport with three or four wheels, powered by a liquid-fueled internal combustion engine.
>>> Query: did edison invent the car battery?, Index: 4, Context: 1898 â Conrad Hubert, known as the founder of the Eveready Battery Company, invented the electric hand torch, or flashlight â a dry cell battery, bulb and rough brass reflector inside a paper tube. - EvereadyÂ® introduced the D size battery for the first handheld flashlight.
>>> Query: did edison invent the car battery?, Index: 5, Context: Edison's Alkaline Battery. As with several of Thomas Edisonâs later projects, such as his effort to mine iron ore and his quest to create synthetic rubber, his attempts at improving the battery did not lead to the results he hoped for. Edison started his work on the battery in the 1890s, just after the automobile had been introduced.
output: 3 1 4 2 5 ({'id': '1885108', 'text': "Edison's Alkaline Battery. As with several of Thomas Edisonâ\x80\x99s later projects, such as his effort to mine iron ore and his quest to create synthetic rubber, his attempts at improving the battery did not lead to the results he hoped for. Edison started his work on the battery in the 1890s, just after the automobile had been introduced.",
```

format: [Query: did edison invent the car battery? Index: 1, Context: …]

3: Mercedes-Benz engine

1: first automobile

4: Conrad Hubert’s battery company invented flashlight.

2: Edison didn’t invent phonograph.

5: Edison’s car battery improvement didn’t come out as expected.

output: 3 1 4 2 5