Sparse Pairwise Re-ranking with Pre-trained Transformers

ICTIR 2022

Lukas Gienapp\textsuperscript{1} \hspace{1cm} Maik Fröbe\textsuperscript{2} \hspace{1cm} Matthias Hagen\textsuperscript{2} \hspace{1cm} Martin Potthast\textsuperscript{1}

\textsuperscript{1}Universität Leipzig \hspace{1cm} \textsuperscript{2}Martin-Luther-Universität Halle-Wittenberg
Pairwise ranking models are slow.
Problem Description

Pairwise ranking models are slow.

Can we make them faster?
Background

Evolution of feature-based learning to rank models

- Pointwise LTR $\Rightarrow$ Pairwise LTR $\Rightarrow$ Listwise LTR

From pointwise to pairwise transformers [Nogueira et. al 2020, Pradeep et. al 2021]:

- Pointwise retrieval with monoT5:
  **Input:** Query $q$, Document $d$
  **Output:** Probability that $d$ is relevant to $q$

- Pairwise retrieval with duoT5:
  **Input:** Query $q$, Document $d_a$, Document $d_b$
  **Output:** Pairwise preference (probability that $d_a$ is more relevant to $q$ than $d_b$)
Background

Evolution of feature-based learning to rank models

- Pointwise LTR ⇒ Pairwise LTR ⇒ Listwise LTR

From pointwise to pairwise transformers [Nogueira et. al 2020, Pradeep et. al 2021]:

- **Pointwise retrieval with monoT5:**
  - **Input:** Query $q$, Document $d$
  - **Output:** Probability that $d$ is relevant to $q$

- **Pairwise retrieval with duoT5:**
  - **Input:** Query $q$, Document $d_a$, Document $d_b$
  - **Output:** Pairwise preference (probability that $d_a$ is more relevant to $q$ than $d_b$)

<table>
<thead>
<tr>
<th>Ranker</th>
<th>No. Inferences</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>monoT5 (k=1000)</td>
<td>1000</td>
<td>0.50</td>
</tr>
<tr>
<td>+ duoT5 (k=50)</td>
<td>1000 + 2450</td>
<td>0.67</td>
</tr>
</tbody>
</table>

For $k$ documents, duoT5 makes $k^2 - k$ pairwise comparisons.
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)

2. Pointwise re-ranking (top 1000)
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)
2. Pointwise re-ranking (top 1000)
3. Pairwise re-ranking (top 50)
   - assemble document pairs
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)

2. Pointwise re-ranking (top 1000)

3. Pairwise re-ranking (top 50)
   - assemble document pairs
   - pairwise inference
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. **BM25 ranking (whole corpus)**

2. **Pointwise re-ranking (top 1000)**

3. **Pairwise re-ranking (top 50)**
   - assemble document pairs
   - pairwise inference
   - score aggregation

4. **Score aggregation**
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)
2. Pointwise re-ranking (top 1000)
3. Pairwise re-ranking (top 50)
   - assemble document pairs
   - pairwise inference
   - score aggregation
4. Rank by aggregated score
Mono-Duo Pairwise Reranking [Pradeep et. al 2021]

Pipeline Overview

Four steps:

1. BM25 ranking (whole corpus)
2. Pointwise re-ranking (top 1000)
3. Pairwise re-ranking (top 50)
   - assemble document pairs
   - pairwise inference
   - score aggregation
4. Rank by aggregated score
Contributions

Key improvements in the pairwise step:

1. Efficiency
   - quadratic comparison amount when doing all doc-doc pairs is problematic
   - sparse comparison set for efficiency
   - But: requires good sampling approach
Contributions

Key improvements in the pairwise step:

1. Efficiency
   - quadratic comparison amount when doing all doc-doc pairs is problematic
   - sparse comparison set for efficiency
   - But: requires good sampling approach

2. Effectiveness
   - choice of aggregation method has direct impact on effectiveness
   - little attention in previous work
   - we investigate several aggregation methods with and without sampling
Sorting as Aggregation

**Sorting**: The most efficient solution we can hope for

- Kwiksort: “Quicksort” for pairwise preferences
- Complexity: $O(n \log n)$ instead of $O(n^2)$
**Sorting as Aggregation**

**Sorting**: The most efficient solution we can hope for

- Kwiksort: “Quicksort” for pairwise preferences
- Complexity: $O(n \log n)$ instead of $O(n^2)$

**But**: requires total order between predictions

- **consistency**: score of document pair $(d_a, d_b)$ should be the inverse of $(d_b, d_a)$
- **transitivity**: predictions for three documents should be transitive

<table>
<thead>
<tr>
<th>Property</th>
<th>Average Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>0.498</td>
</tr>
<tr>
<td>Transitivity</td>
<td>0.693</td>
</tr>
</tbody>
</table>

*dupT5 on MS MARCO*

Average over all document pairs of 50 topics at depth 50.
Sorting as Aggregation

**Sorting:** The most efficient solution we can hope for

- Kwiksort: “Quicksort” for pairwise preferences
- Complexity: $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^2)$

**But:** requires total order between predictions

- **consistency:** score of document pair $(d_a, d_b)$ should be the inverse of $(d_b, d_a)$
- **transitivity:** predictions for three documents should be transitive

<table>
<thead>
<tr>
<th>duoT5 on MS MARCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td>Consistency</td>
</tr>
<tr>
<td>Transitivity</td>
</tr>
</tbody>
</table>

Average over all document pairs of 50 topics at depth 50.

<table>
<thead>
<tr>
<th>pipeline</th>
<th>No. Comp.</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>monoT5</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td>+ duoT5</td>
<td>2450</td>
<td>0.67</td>
</tr>
<tr>
<td>+ duoT5 with Kwiksort</td>
<td>85</td>
<td>0.42</td>
</tr>
</tbody>
</table>

MS MARCO (Passage; DL 19/20; k=50 documents).

Pairwise model output contains too many individual errors to sort!
Sampling Methods
Random Sampling

- **Motivation**: baseline method

- **Method**:
  - randomly sample a fraction $f$ of possible comparisons
  - sampling is separate per doc.

- **Upside**: parameter-free

- **Downside**: not deterministic, pointwise ranking is not used
Sampling Methods
Neighbor Window Sampling

- **Motivation**: deterministic method

- **Method**:
  - based on pointwise reranking
  - compares a doc. to its $m$ successors
  - wraps around to compare last to first

- **Upside**: parameter-free, incorporates pointwise ranking context locally

- **Downside**: global context lost, cannot stray far from pointwise ranking
Sampling Methods
Skip Window Sampling

- **Motivation**: deterministic + global method
- **Method**:
  - like exhaustive window sampling
  - skips with steps size $\lambda$
- **Upside**: incorporates pointwise ranking context globally
- **Downside**: parametric, $\lambda$ has to be tuned
Aggregation Methods

Four different aggregation methods, each from a different aggregation paradigm.

Additive Aggregation

- baseline [Pradeep et. al 2021]
- symmetric sum of preference scores
Aggregation Methods

Four different aggregation methods, each from a different aggregation paradigm.

Additive Aggregation
- baseline [Pradeep et. al 2021]
- symmetric sum of preference scores

Bradley-Terry Aggregation
- maximum-likelihood logistic regression
- optimizes to fit pairwise preferences
Aggregation Methods

Four different aggregation methods, each from a different aggregation paradigm.

<table>
<thead>
<tr>
<th>Additive Aggregation</th>
<th>Bradley-Terry Aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline [Pradeep et. al 2021]</td>
<td>maximum-likelihood logistic regression</td>
</tr>
<tr>
<td>symmetric sum of preference scores</td>
<td>optimizes to fit pairwise preferences</td>
</tr>
</tbody>
</table>

Greedy Aggregation

- similar to additive
- identify best doc., then recursively apply to remaining
## Aggregation Methods

Four different aggregation methods, each from a different aggregation paradigm.

<table>
<thead>
<tr>
<th>Additive Aggregation</th>
<th>Bradley-Terry Aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>❑ baseline [Pradeep et. al 2021]</td>
<td>❑ maximum-likelihood logistic regression</td>
</tr>
<tr>
<td>❑ symmetric sum of preference scores</td>
<td>❑ optimizes to fit pairwise preferences</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Greedy Aggregation</th>
<th>PageRank Aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>❑ similar to additive</td>
<td>❑ graph-based aggregation</td>
</tr>
<tr>
<td>❑ identify best doc., then recursively apply to remaining</td>
<td>❑ docs. are nodes, comparisons are weighted edges</td>
</tr>
</tbody>
</table>
Evaluation
Experimental Setup

- **Collection**: MS MARCO

- **Ranking Pipeline**:
  1. BM25 with default parameters
  2. Top 1000 reranking with monoT5
  3. Top 50 reranking with duoT5

- **Measure**: nDCG@10 with qrels from TREC-DL passage ranking

- **Parameters**: grid search was carried out to find optimal $\lambda$-value for S-Window sampling
Evaluation

nDCG@10 on MS MARCO

Additive Aggregation

Bradley-Terry Aggregation

Greedy Aggregation

PageRank Aggregation

nDCG@10 on MS MARCO

- G-Random
- N-Window
- S-Window
- Pointwise
- Unsampled
Greedy aggregation is best under no sampling.
Greedy aggregation is best across all sampling methods.
Global sampling context seems more important than local sampling context.
Evaluation

nDCG@10 on MS MARCO

S-Window sampling is best across all aggregation methods.
Evaluation

Best setup matches effectiveness down to 30% of the comparisons.
Best setup is competitive down to 10% of the comparisons. ($\Delta = 0.04$)
Conclusion

Findings:

- Sparse comparison sets are highly effective at increasing the efficiency of pairwise retrieval
- Effectiveness can be increased with better aggregation approaches
- Up to 90% cost savings are possible
Conclusion

Findings:

- Sparse comparison sets are highly effective at increasing the efficiency of pairwise retrieval
- Effectiveness can be increased with better aggregation approaches
- Up to 90% cost savings are possible

What's more in the paper?

- Replication of experiments on the ClueWebs, corroborating results
- More in-depth evaluation of comparison properties
- Code: github.com/webis-de/ICTIR-22
Conclusion

Findings:
- Sparse comparison sets are highly effective at increasing the efficiency of pairwise retrieval
- Effectiveness can be increased with better aggregation approaches
- Up to 90% cost savings are possible

What's more in the paper?
- Replication of experiments on the ClueWebs, corroborating results
- More in-depth evaluation of comparison properties
- Code: github.com/webis-de/ICTIR-22

What's more in the future?
- Instead of lower budget at same depth, increase depth at same budget
- Promising for high-recall search applications
- Model adaptions for more consistent predictions
- Dynamic sampling approaches

Thank You!