WebConf23

05/04/2023

Optimizing Guided Traversal for Fast Learned Sparse Retrieval

Yifan Qiao, Yingrui Yang, Haixin Lin, Tao Yang

UC SANTA BARBARA

Motivation

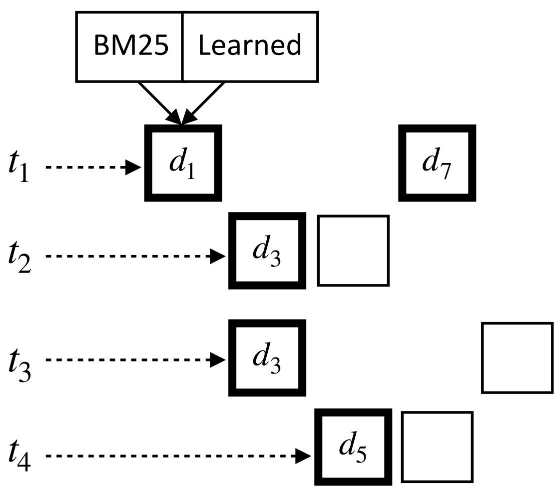
Problem

Fast top *k* document retrieval with an inverted index using a learned sparse representation: e.g. SPLADE [Formal et al. SIGIR'21 and 22], uniCOIL, DeepImpact

Standard retrieval with dynamic index pruning: MaxScore or VBMW

Prior work: GTI [Mallia et al. SIGIR'22]

- Store both BM25 and learned weights of a document in an inverted index
- Uses **BM25** based scoring to **skip documents** while final ranking uses a linear combination of learned neural weights and BM25 weights



$$\sum_{t \in q} w_t \cdot w_{Learned}(t, d) < \theta_{Learned}$$

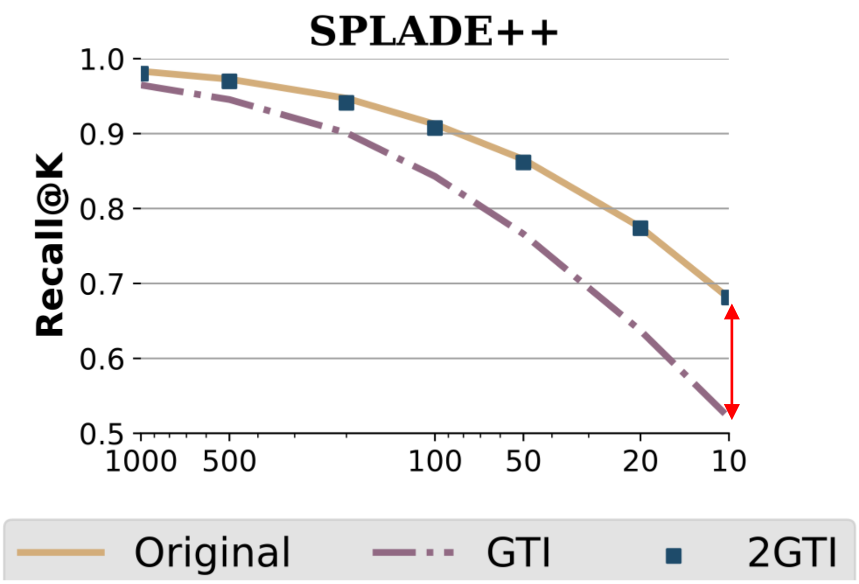
$$\downarrow$$

$$\sum_{t \in q} w_t \cdot w_{BM25}(t, d) < \theta_{BM25}$$

Motivation

Weakness addressed

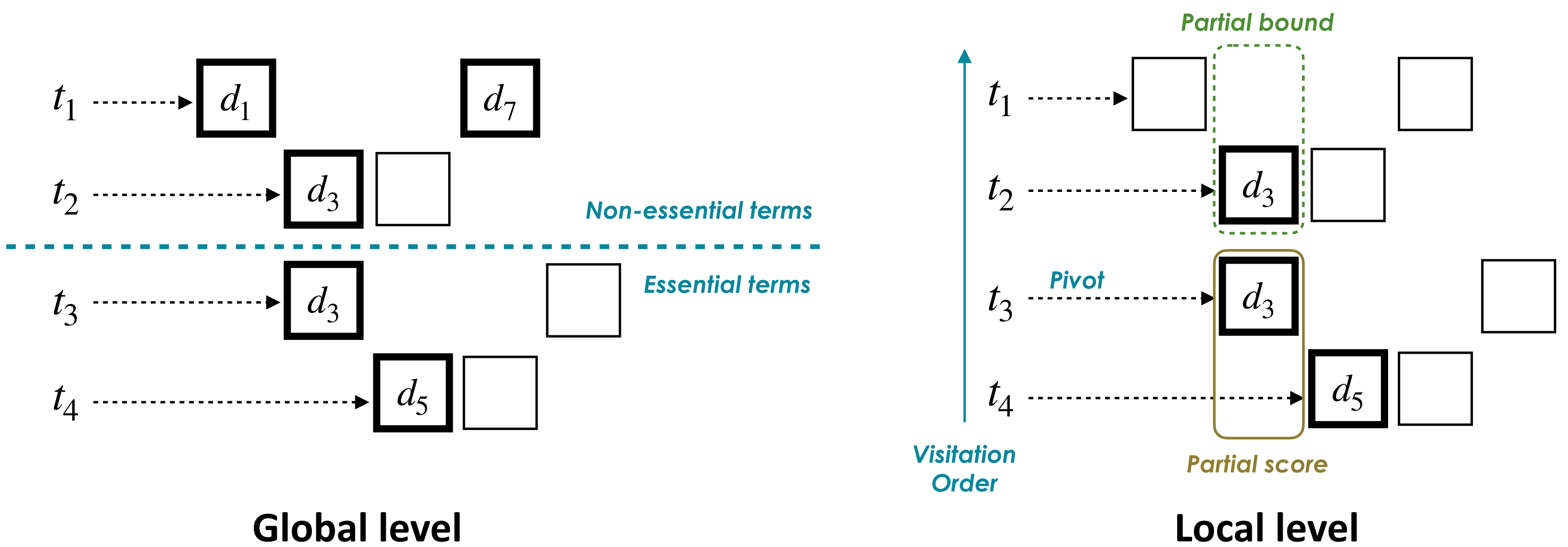
- When k becomes relatively small, the relevance drops significantly, indicating BM25 based guidance for pruning is too aggressive.
- Token inconsistency in BM25 model and a learned neural model creates un-smoothed weighting and results in significant relevance drop.



Proposed Solution: 2GTI

Two level pruning guidance with different scoring and thresholding

- View pruning in standard **MaxScore** retrieval algorithm in two levels
  - **Global level**: partitioning of the non-essential and essential terms
  - **Local level**: skipping a document selected during possible deep visitation
- Allow different scoring/thresholding at these two levels and at final ranking



- 2GTI on VBMW is similar

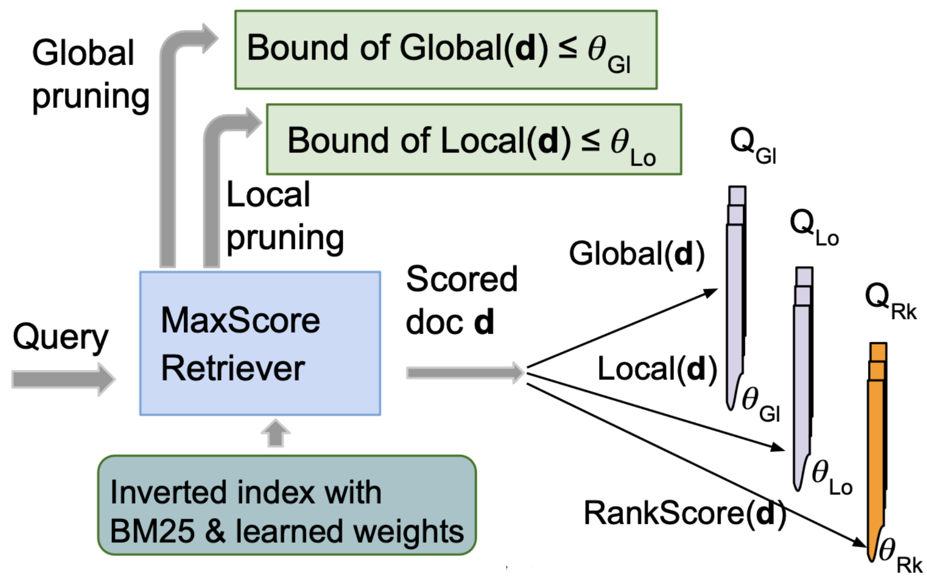
Proposed Solution: 2GTI

Maintain 3 top *k* queues with different rankings

- Q_{Gl} uses ranking R_α for global pruning: $Global(d) = \alpha \cdot w_{BM25} + (1-\alpha) \cdot w_{learned}$
- Q_{Lo} uses ranking R_β for local pruning: $Local(d) = \beta \cdot w_{BM25} + (1-\beta) \cdot w_{learned}$
- Q_{Rk} uses ranking R_γ for final ranking: $RankScore(d) = \gamma \cdot w_{BM25} + (1-\gamma) \cdot w_{learned}$

Maintain 3 thresholds for dynamic index pruning

- θ_{Gl} for essential term partitioning based on R_α
- θ_{Lo} for minimum top *k* score based on R_β
- θ_{Rk} for top k thresholding based on final ranking R_γ.

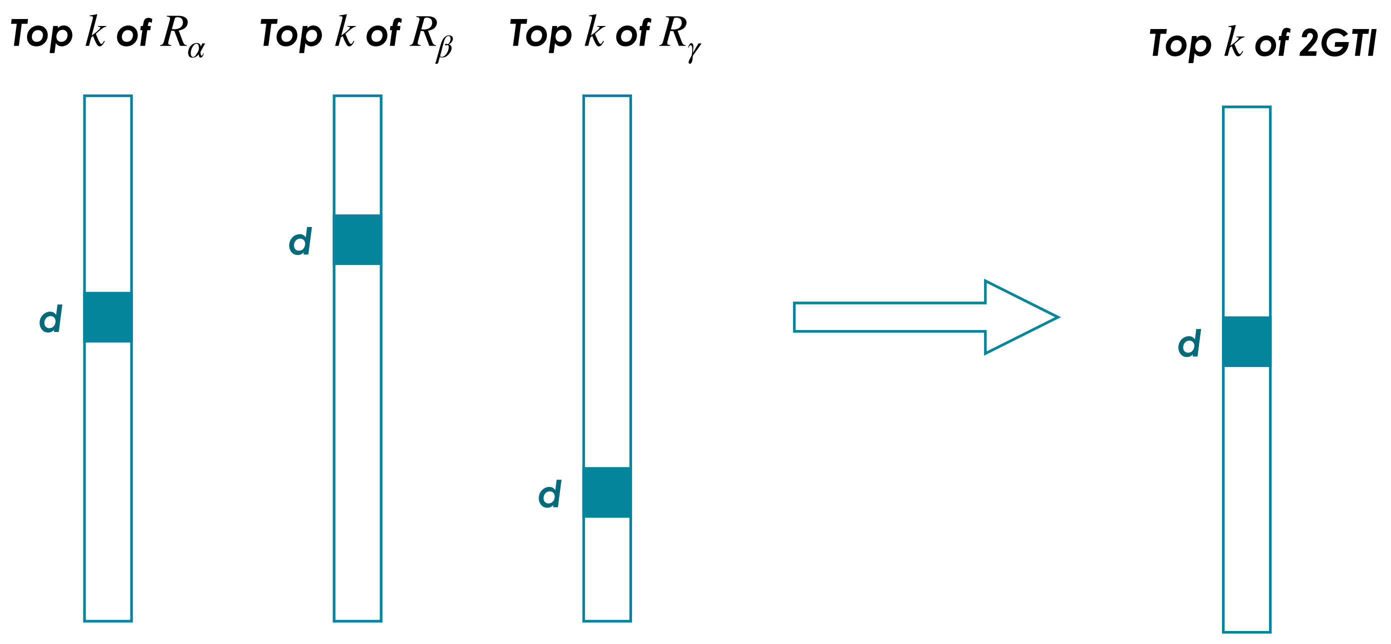


Relevance Properties of 2GTI

Objective: Analyze relevance behavior of 2GTI formally and its competitiveness

(GTI is a special case of 2GTI with $\alpha = \beta = 1$)

#1: Top documents agreed by top k of each ranking R_α, R_β, and R_γ are kept on the top k by 2GTI.



Relevance Properties of 2GTI

#1: Top documents agreed by top k of each ranking R_α, R_β, and R_γ are kept on the top k by 2GTI.

#2: Properly configured 2GTI can outperform the two-stage algorithm R_2: retrieval with R_α and re-ranking with R_γ.

(1) When $\alpha = \beta$ or $\beta = \gamma$, the average rank score of the top k positions produced by 2GTI is equal or higher than this two-stage algorithm R_2.

$$\sum_{d \in 2GTI} RankScore_{(\gamma)}(d) \geq \sum_{d \in R_2} RankScore_{(\gamma)}(d)$$

(2) When R_γ outmatches R_β which outmatches R_α, 2GTI retrieves equal or more relevant results at top k positions than R_2.

$$Recall@k(2GTI) \geq Recall@k(R_2)$$

Evaluation

MS MARCO Passage

MS MARCO Passage	$k = 10$		$k = 1000$	
Dev	**MRR@10**	**MRT (P_{99})**	**MRR@10**	**MRT (P_{99})**
SPLADE++-Orignal	0.3937	121 (483)	0.3937	278 (819)
-GTI	0.2687	118 (440)	0.2961	332 (1059)
-2GTI-Accurate	0.3939	31.1 (171)	0.3946	109 (478)
-2GTI-Fast	0.3934	22.7 (116)	0.3937	43.1 (144)

2.6x faster

6.5x faster

milliseconds

- On MS MARCO Passage Dev, $k = 1000$
  - 2GTI-Accurate produces slightly higher MRR@10 (due to BM25 interpolation) than the original SPLADE while being 2.6x faster
  - 2GTI-Fast has similar MRR score while being 6.5x faster
- Similar trend observed in TREC DL'19 and DL'20

Evaluation

Token and weight alignment between BM25 index and learned index

For those missing weights in the BM25 model

- /0: do nothing
- /1: fill with 1
- /s: fill with learned scores scaled by ratio of mean values of non-zero weights

SPLADE++. $k = 10$	**MRR@10**	**Recall@10**	**MRT**	P_{99}
Weight alignment for GTI ($\alpha = 1, \beta = 1, \gamma = 0.05$)				
GTI/0	0.2687	0.5209	118	440
GTI/1	0.3036	0.5544	26.7	114
GTI/s	0.3468	0.5774	9.1	36.1
Weight alignment for 2GTI-Accurate ($\alpha = 1, \beta = 0, \gamma = 0.05$)				
2GTI/0	0.3933	0.6799	328	1262
2GTI/1	0.3933	0.6818	89.3	393
2GTI/s	0.3939	0.6812	31.1	171

Faster & more accurate

10.5x faster

Evaluation

Zero-shot performance (13 BEIR datasets)

	$k = 10$		$k = 1000$	
BEIR	**nDCG@10**	**Avg. Speedup**	**nDCG@10**	**Avg. Speedup**
SPLADE++-Orignal	0.500	-	0.500	-
-GTI/s	0.430	6.1x	0.496	2.1x
-2GTI/s-Fast	0.499	2.0x	0.501	2.5x

Efficiency-driven SPLADE

- Apply 2GTI on the efficiency-driven SPLADE model [Lassance et al. SIGIR'22] with a relevance tradeoff ($k = 10$)

BT-SPLADE-L	**MRR@10**	**Recall@10**	**MRT**
Original MaxScore	0.3799	0.6626	17.4
2GTI/s (α=1, β=0.3, γ=0.05)	0.3772	0.6584	8.0
GTI/s ($\alpha = \beta = 1, \gamma = 0.05$)	0.3284	0.5520	6.6

2.2x faster

Conclusions

- 2GTI retrieval manages 3 top *k* queues with 3 linear combinations of neural and BM25 weights to rank/skip docs
  - Pruning decision is more accurate than GTI
  - Can outperform a two-stage retrieval algorithm at least
- Sample configurations for SPLADE++:
  - R_α: $\alpha \cdot w_{BM25} + (1-\alpha) \cdot w_{learned}$, with $\alpha = 1$
  - R_β: $\beta \cdot w_{BM25} + (1-\beta) \cdot w_{learned}$, with $\beta = 0$ or 0.3
  - R_γ: $\gamma \cdot w_{BM25} + (1-\gamma) \cdot w_{learned}$, with $\gamma = 0.05$
- Smooth weight alignment is necessary to address token inconsistency between BM25 and neural models
- For MS MARCO passages with SPLADE++, 5x to 7x faster than original MaxScore and GTI

Thanks and Q/A?