Adapting Learned Sparse Retrieval for Long Documents

Thong Nguyen
t.nguyen2@uva.nl
University of Amsterdam
Amsterdam, Netherlands

Sean MacAvaney
sean.macavaney@glasgow.ac.uk
University of Glassgow
Glasgow, Scotland, UK

Andrew Yates
a.c.yates@uva.nl
University of Amsterdam
Amsterdam, Netherlands

ABSTRACT

Learned sparse retrieval (LSR) is a family of neural retrieval methods that transform queries and documents into sparse weight vectors aligned with a vocabulary. While LSR approaches like Splade work well for short passages, it is unclear how well they handle longer documents. We investigate existing aggregation approaches for adapting LSR to longer documents and find that proximal scoring is crucial for LSR to handle long documents. To leverage this property, we proposed two adaptations of the Sequential Dependence Model (SDM) to LSR: ExactSDM and SoftSDM. ExactSDM assumes only exact query term dependence, while SoftSDM uses potential functions that model the dependence of query terms and their expansion terms (i.e., terms identified using a transformer's masked language modeling head).

Experiments on the MSMARCO Document and TREC Robust04 datasets demonstrate that both ExactSDM and SoftSDM outperform existing LSR aggregation approaches for different document length constraints. Surprisingly, SoftSDM does not provide any performance benefits over ExactSDM. This suggests that soft proximity matching is not necessary for modeling term dependence in LSR. Overall, this study provides insights into handling long documents with LSR, proposing adaptations that improve its performance.

github.com/thongnt99/lsr-long

CCS CONCEPTS

• **Information systems → Information retrieval**.

KEYWORDS

learned sparse retrieval, term proximity, long documents

ACM Reference Format:
Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. Adapting Learned Sparse Retrieval for Long Documents. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591943

1 INTRODUCTION

Learned sparse retrieval (LSR) is a recent family of first-stage information retrieval that utilizes neural models, typically transformers, to encode queries and documents into sparse weight vectors aligned with a vocabulary. By operating in the lexical space, LSR provides several benefits over dense retrieval, including greater transparency while remaining comparably effective [4, 5, 15]. Moreover, LSR's compatibility with an inverted index allows for the reuse of techniques previously developed for lexical retrieval (e.g., BM25) [8].

Previous evaluations of LSR methods have focused on short texts due to transformers' input length limit, such as by focusing on the MSMARCO passage collection [16] or truncating texts to the transformer's maximum input length.

To handle long documents, a common practice is to split the input into multiple segments, encode these segments individually, and aggregate the output. Several works have investigated score and representation aggregation strategies for the Cross-Encoder architecture, finding that representation aggregation typically outperforms score aggregation (e.g., taking the max of segments' vector representations rather than taking the max segment score) [2, 7, 11, 21].

We hypothesize that these insights may change for LSR due to the inherent difference in the encoding mechanism between Cross-Encoders and LSR. While Cross-Encoders encode the query and document simultaneously, LSR encodes queries and documents separately, which makes it challenging to judge what from a document segment is not relevant to the information need. As a result, the aggregation operation may accumulate noise and make documents less separable. In addition, as the documents get longer, it is particularly difficult for LSR, which relies solely on term-based representations, to deal with scattered term matches. Intuitively, matches within neighbouring text should be a stronger relevance signal than isolated matches present in different segments [19].

In this work, we first study the effectiveness of existing aggregation approaches for adapting LSR to long documents. Specifically, we investigate three aggregation operators (*max, sum, mean*) on two output levels (*representation, score*). By applying these approaches to the state-of-the-art LSR method [5, 15] on two datasets (MSMARCO Document, TREC Robust04), we find that most of the approaches, except for the max score aggregation, are fragile with long documents; as can be seen in Figure 1, their performance drops severely

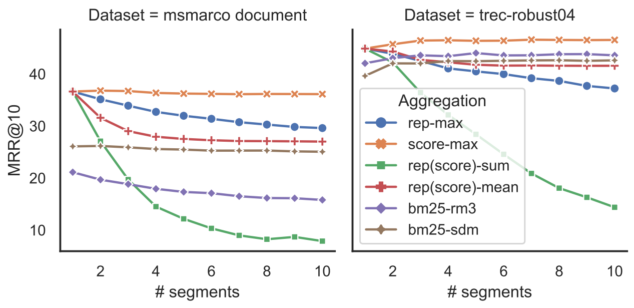

Figure 1: Performance of aggregation methods proposed for neural rankers with LSR and long documents.

as more document segments are added into the representations. Max score aggregation, however, avoids this downward trend with mostly stable performance over different numbers of segments. This suggests that the proximity matching that inherently happens with max score aggregation is crucial for LSR to handle long documents.

In order to better utilize local proximity, we propose two approaches, ExactSDM and SoftSDM, that adapt the SDM [14] for use with LSR. ExactSDM assumes a phrase or proximity match if the exact phrase or constituent terms appear in the document within a local document window. SoftSDM relaxes this assumption by allowing constituent terms to be softly matched with their expansion terms, as the Splade method does for unigrams.

We evaluate the performance of both ExactSDM and SoftSDM on the MSMARCO Document and TREC Robust04 datasets, finding that both approaches consistently outperform previous aggregation methods in the context of LSR. However, SoftSDM does not provide significant advantages over ExactSDM, making ExactSDM a more attractive option as it can be applied to a wider range of LSR methods, including both MLM-based architectures that leverage a masked language modeling (MLM) head to identify and score expansion terms [4, 5, 10] and MLP-based architectures that estimate term salience using a multilayer perceptron (MLP) with no soft matching via expansion terms [3, 9, 13].

Overall, our work sheds light on adapting LSR to long documents, with an approach adapted from SDM that significantly outperforms existing aggregation techniques for neural retrieval methods.

2 BACKGROUND

2.1 Learned sparse retrieval

Learned Sparse Retrieval (LSR) methods score a query-document pair by taking the dot product between their sparse vectors produced by a query encoder (f_Q) and a document encoder (f_D), i.e., $score(q,d) = f_Q(q) \cdot f_D(d) = w_q \cdot w_d = \sum_{i=1}^{|V|} w_q^i w_d^i$, where w_i represents the predicted weight of the i^{th} term in the vocabulary.

Compared to dense retrieval, LSR's vectors are high dimensional and sparse, with most of their elements being zero. The dimensions of these vectors are tied to a vocabulary, so LSR is closely connected to traditional sparse retrieval methods such as BM25. However, unlike BM25, LSR learns the term weights instead of obtaining them from corpus statistics like TF-IDF. This similarity with BM25 makes LSR compatible with many existing techniques such as the inverted index, which were previously built for purely lexical retrieval models like BM25 [8].

Splade is a state-of-the-art LSR method that uses a masked language modeling (MLM) head based on BERT to perform term expansion and term weighting end-to-end. Given an input query/document, Splade encodes it into a logit matrix $\mathbf{W}$, where $W_{i,j}$ represents the translation probability score from the i^{th} term in the input sequence to the j^{th} vocabulary item ($|V| \approx 30k$). Splade then outputs, for each term in the vocabulary, a non-negative log-scaled weight, which is the maximum logit value of the term across the sequence, i.e., $w_k = \max_i(\log(1 + \text{ReLU}(W_{i,k})))$. This max aggregation retains only the term weights and, like all LSR methods, it drops positional information. In other words, Splade uses term positional information when estimating query and document term weights, but is position-agnostic when scoring documents. Positional information is critical for inferring phrase semantics, however; *"MU defeated Arsenal"* has the opposite meaning of *"Arsenal defeated MU"*. Furthermore, IR Axioms suggest that documents that contain phrases from the query (or query terms in close proximity) are more relevant than those with query terms scatted throughout [19].

2.2 Term Dependence Model

In prior work, researchers introduced techniques that incorporate the dependence of two or more terms into traditional bag-of-words retrieval models [1, 6, 14]. Among these techniques, the Sequential Dependence Model [14] (SDM) is one of the most widely known and has been shown to be effective.

SDM assumes a dependence between neighboring query terms and models this dependence by a Markov Random Field (MRF). The MRF in SDM defines a joint probability over a graph G whose nodes are document random variable (**D**) and query terms $q_1, q_2, .., q_{|Q|}$. The edges between nodes represent the dependency between them, where a node is independent of all other nodes given its neighbors. The query document conditional relevance probability is defined via this joint probability factorized as follows:

$$P_\Lambda(D|Q) = \frac{P_\Lambda(Q,D)}{P_\Lambda(Q)} \overset{rank}{=} P_\Lambda(Q,D) \tag{1}$$

$$= \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \psi(c,\Lambda) \overset{rank}{=} \sum_{c \in C(G)} \log \psi(c,\Lambda) \tag{2}$$

Here, $C(G)$ is the set of cliques in the graph, $\psi(c,\Lambda)$ is a potential function parameterized by Λ, and Z_Λ is a normalization factor. SDM realizes the above formula by defining three potential functions:

- Exact individual term matching:
  $\psi_T(q_i, D) = \lambda_T log\left[(1-\alpha_D)\frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|}\right]$
- Exact n-gram/phrase matching:
  $\psi_O(q_i...q_{i+k}, D) = \lambda_O \log\left[(1-\alpha_D)\frac{tf_{\#1(q_i...q_{i+k}),D}}{|D|} + \alpha_D \frac{cf_{\#1(q_i...q_{i+k})}}{|C|}\right]$
- Exact multi-term proximity matching:
  (terms appear ordered/unordered within a window of size N)
  $\psi_U(q_i...q_j, D) = \lambda_U \log\left[(1-\alpha_D)\frac{tf_{\#uwN(q_i...q_j),D}}{|D|} + \alpha_D \frac{cf_{\#uwN(q_i...q_j)}}{|C|}\right]$
  (tf is term(s) frequency in D; cf is term(s) frequency in the corpus C; $\lambda_{(s)}$ are learnable weights; α_D is a smoothing factor)

Plugging the above potential functions into Equation 2, we get the following ranking function:

$$P_\Lambda(D|Q) \overset{rank}{=} \sum_{q_i \in Q} \psi_T(q_i, D) + \sum_{q_i...q_{i+k} \in Q} \psi_O(q_i...q_{i+k}, D) + \sum_{q_i...q_j \in Q} \psi_U(q_i...q_j, D) \tag{3}$$

3 SEQUENTIAL DEPENDENCE FOR LSR

In this section, we will introduce SoftSDM and ExactSDM, which are adapted from the original SDM, for performing phrase and proximity matching with LSR. We formulate these two adaptations on top of the state-of-the-art Splade LSR encoder that utilizes BERT's masked language modeling head. While SoftSDM's formulation is tied to the MLM head, ExactSDM can be generalized to other LSR architectures that produce term weights differently.

3.1 Query and document representations

In order to measure the dependence between terms, it is necessary to track both term positions and weights. We utilize a modified version of Splade [15] that enhances efficiency without compromising its effectiveness. It produces query term weights using an MLP layer (instead of MLM) on top of BERT's last hidden states, generating a sequence of term weights ($w_q = w_q^1, w_q^2..w_q^{|Q|}$) with w_q^i representing the weight of the i^{th} query token.

On the document side, the model's MLM document encoder produces for each document a logit matrix $\mathbf{W^D}$ where $W_{i,j}^D$ represents the translation score from the i^{th} document term to the j^{th} vocabulary item. This logit matrix provides access to term positions and preceding term-weight vectors. We sparsify $\mathbf{W^D}$ during training, enabling efficient storage via a sparse matrix format.

3.2 Soft Sequential Dependence Model

To adapt SoftSDM for use with the MLM head, we introduce three new functions derived from the query and document representations described previously. These potential functions are defined as follows (*with* $v[q_i]$ *used to denote the index of* q_i *in the vocabulary*):

- **Soft individual term matching**: This potential function is equivalent to the max-aggregation followed by a dot product used in LSR methods (e.g., Splade).

$$\psi_{ST}(q_i, D) = \lambda_{ST} \max_{1 \le r \le |D|} w_q^i W_{r,v[q_i]}^D \tag{4}$$

- **Soft n-gram/phrase matching**: Soft phrase similarity measures the likelihood of terms in one phrase translating to corresponding terms in another, considering the importance of each term. This function computes the maximum similarity between a query phrase and document phrases starting at every position r.

$$\psi_{SO}(q_i...q_{i+k}, D) = \lambda_{SO} \max_{1 \le r \le |D|-k} \sum_{l=0}^{k} w_q^{i+l} W_{r+l,v[q_{i+l}]}^D \tag{5}$$

- **Soft proximity matching**: This function approximates the maximum likelihood of translating terms within document windows of size p to a set of query terms regardless of the order, while also considering term importance.

$$\psi_{SU}(q_i...q_{i+k}, D) = \lambda_{SO} \max_{1 \le r \le |D|-p} \sum_{h=i}^{i+k} w_q^h \left[\max_{r \le l < r+p} W_{l,v[q_h]}^D \right] \tag{6}$$

3.2.1 Exact Sequential Dependence Model. ExactSDM uses the same potential function for individual term matching as SoftSDM (Equation 4), which has been shown to benefit LSR [4, 5]. However, with ExactSDM, we are interested in evaluating the impact of soft phrase/proximity matching. It accomplishes this by allowing a document term to only translate to itself in the output logits and disabling document expansion. This change modifies the potential function formulas in Equation 5 and 6 by modifying the logit matrix to only contain self-translations. ExactSDM is compatible with an MLM or MLP document encoder, making it applicable to additional LSR methods like uniCOIL [9], DeepCT [3], and DeepImpact [13].

4 EXPERIMENTS

4.1 Datasets

Our experiments use two long-document benchmarks: MSMARCO Document and TREC Robust04. The MSMARCO Document dataset [16] includes 3.2 million documents, 5.2K dev queries, and 367K queries in the train split. TREC Robust04 [20] consists of approximately 0.5 million news articles and 250 queries, with each query containing three fields: title, description, and narrative. In this work, we use only the description field, which contains a natural language version of the query. We access these datasets using ir_datasets [12].

4.2 Experimental settings

We utilize the Splade-based LSR architecture using DistilBERT [18] with a maximum input length of 512 tokens for all experiments. This architecture was trained on the MS-MARCO passage dataset using hard negatives and distillation from a Cross-Encoder [17]. To handle long documents, we divided them into sentences and grouped sentences into segments of up to 400 tokens, with longer sentences considered as a single segment. Document representations or scores were derived from the segments' representations/scores generated by the Splade model, which achieved a Passage MRR@10 of 38.51 when previously trained on the MS-MARCO passage dataset [16] (with no fine-tuning on the document dataset).

We explore various aggregation approaches, including Rep-max, Score-max, Rep(score)-sum, and Rep(score)-mean, that describe how segment-level term scores are treated to arrive at a document-level score. The representation aggregation methods (e.g., Rep-max) operate on the sparse vector corresponding to each segment, whereas the score aggregation methods (e.g., Score-max) operate on relevance scores assigned to each segment. With Rep-max, a document-level sparse vector is computed by taking the maximum weight for each vocabulary term across all document segments. This document-level vector is then used to compute the relevance score by taking the dot product with the query vector. Score-max calculates a relevance score between each segment and the query, and then returns the maximum segment relevance score as the document-level relevance score. Rep-sum and Rep-mean replace Rep-max's max pooling with sum pooling or mean pooling across segments. Due to the distributive property of dot products, sum and mean pooling return identical results regardless of whether they operate on sparse vectors (representations) or scores. Thus, we refer to these as Rep(score)-sum and Rep(score)-mean.

SoftSDM and ExactSDM were evaluated using the same query and document segment representations as the above approaches, but with additional positional information. Score-max can be viewed as a special case of SoftSDM with soft proximity matching within the segment windows. When using these approaches, we fine-tune the three weighting factors of the potential functions using the MSMARCO Document training triplets obtained from Boytsov et al. [2]. We also assess the generalizability of these weighting factors on TREC Robust04 without any further fine-tuning.

4.3 Results and Discussion

In our experiments, we compare existing aggregation methods with our proposed SDM for LSR variants as the number of segments in documents increases.

Table 1: The results of baselines and SDM variants on MSMARCO Document and TREC Robust04. MRR and NDCG are @10. Recall is @1000. A † indicates $p < 0.05$ (paired t-test between a SDM method and Score-max with Bonferroni correction).

	# segs	ExactSDM		SoftSDM		Score-max		Rep-max		Rep(sc.)-sum		Rep(sc.)-mean		BM25-SDM		BM25-RM3	
		MRR	R	MRR	R	MRR	R	MRR	R	MRR	R	MRR	R	MRR	R	MRR	R
MSDoc Dev	1	**37.08**	95.49	36.98	95.49	36.63	95.49	36.63	95.49	36.63	95.49	36.63	95.49	26.09	89.91	21.13	91.39
	2	†37.45	96.51	†**37.53**	96.51	36.80	96.51	35.14	96.40	27.09	88.81	31.58	95.69	26.17	90.43	19.69	91.39
	3	†37.36	96.76	†**37.41**	96.76	36.72	96.76	33.91	96.34	19.71	75.24	29.02	95.13	25.91	90.45	18.83	90.78
	4	†**37.03**	96.71	36.80	96.71	32.72	96.71	14.52	96.30	36.36	65.51	27.94	94.43	25.58	90.33	17.94	90.20
	5	†**36.95**	96.61	†36.79	96.61	36.24	96.61	31.97	96.15	12.17	55.83	27.52	93.95	25.48	90.12	17.34	89.60
		NDCG	R	NDCG	R	NDCG	R	NDCG	R	NDCG	R	NDCG	R	NDCG	R	NDCG	R
Robust04	1	†46.61	59.26	†**46.65**	59.26	44.88	59.26	44.88	59.26	44.88	59.26	44.88	59.26	39.63	54.84	42.05	61.76
	2	†**47.98**	64.28	†47.94	64.28	45.71	64.28	43.88	63.37	42.23	51.90	44.32	59.88	42.04	60.79	43.04	68.21
	3	†**48.59**	66.75	†48.28	66.75	46.42	66.75	42.51	64.35	36.43	44.25	42.72	59.04	42.02	63.14	43.59	70.34
	4	†**48.87**	68.01	†48.56	68.01	46.48	68.01	41.07	64.78	32.14	39.47	42.25	58.25	42.49	64.37	43.42	71.35
	5	†**49.04**	68.61	†48.65	68.61	46.37	68.61	40.50	64.86	28.41	36.02	41.77	58.10	42.46	64.63	44.02	70.97

We first look at the performance of existing long-document aggregation methods evaluated with LSR. Figure 1 illustrates the results of segment aggregation techniques with respect to the number of segments, with a detailed breakdown in Table 1. We observe a strong downward trend with existing approaches as more document segments are considered, with the exception of Score-max on both the MSMARCO Document and Robust04 datasets.

The approach of summing segment scores or representations, Rep(Score)-sum, suffers from the most severe decrease in performance on the MRR@10 and NDCG@10 measures. The MRR@10/NDCG@10 on MSMARCO Document/Robust04 drops significantly from 36.63/44.88 when using the first segment to worse than BM25 SDM/RM3 when 10 segments are used. The gap becomes even larger (not shown in the Table) when all segments are used, while BM25's scores only slightly decrease. This issue can be attributed to the longer and noisier texts resulting from using more segments, which leads to the addition of more non-relevant terms to the document's representation. Moreover, sum aggregation is inherently biased towards longer documents; less relevant documents may receive a higher accumulated score than shorter but more relevant ones.

The Rep(score)-mean aggregation method corrects for this bias by dividing the sum by the number of segments, thus exhibiting a clear recovery on both datasets. We also note the competitiveness between Rep(score)-mean and Rep-max. On MSMARCO Document, Rep-max consistently outperforms the mean aggregation, but the opposite is true for Robust04. In contrast with prior results on Cross-Encoders [7], where representation aggregation methods are preferable, the Score-max approach outperforms all other aggregation methods and is more robust than the max representation pooling. The MRR@10 of Score-max slightly drops on MSMARCO Document, while its NDCG@10 on Robust04 goes up initially, up to the third segment, and becomes stable after that. The local scoring capability of Score-max likely enables it to avoid noise accumulation. This discrepancy between performance with LSR and with Cross-Encoders may be due to the fact that Cross-Encoders can effectively ignore non-relevant segments, because they know both the query and the document at the time of encoding. LSR methods must encode documents without any knowledge of the query.

The performance of Score-max in the face of a downtrend suggests that proximity is crucial for effectively scoring long documents using LSR. This observation leads to the ExactSDM and SoftSDM models, both of which are adapted from the well-known SDM model from the pre-neural era. As shown in the two leftmost blocks of Table 1, both ExactSDM and SoftSDM consistently outperform previous aggregation approaches and BM25 RM3/SDM on both datasets, regardless of varying document lengths. Overall, ExactSDM and SoftSDM perform competitively on the two datasets.

On MSMARCO Document, the MRR@10 of both SoftSDM and ExactSDM slightly increases on the first two or three segments, but starts to fluctuate after that. In comparison with the previous best pooling method, Score-max, both SDM variants achieve better MRR@10, with the improvement ranging from 1.0% to 2.0%. On Robust04 (zero-shot), the improvement over Score-max is even greater, ranging from 3.9% to 5.8%. Additionally, the improvement slightly increases when more segments are used, demonstrating the ability of SDM variants to exploit long context.

Comparing SoftSDM and ExactSDM, we see that ExactSDM slightly outperforms SoftSDM, making it a great choice for modeling term dependence with LSR. ExactSDM does not necessarily rely on the MLM's logits, which means it is compatible with other LSR methods not using the MLM encoders.

Regarding the optimal phrase and proximity window size, we found that using two-term phrases (bi-grams) and a proximity window of size 8 often returns higher results on MSMARCO Document, which is consistent with the recommendations in [1, 14]. While this setting generalizes well to TREC Robust04 with ExactSDM, SoftSDM needs longer phrases (5 grams) and a longer proximity window (size=10) for better zero-shot performance.

5 CONCLUSION

Our work explores techniques for adapting learned sparse retrieval to long documents. We find that the max-score aggregation approach is robust to varying document lengths, leading to our SoftSDM and Exact SDM adaptations of the SDM model that outperform existing approaches. Surprisingly, SoftSDM's soft-matching ability does not outperform ExactSDM, indicating that it may not be necessary for modeling term dependence.

REFERENCES

[1] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM, 31–40. https://doi.org/10.1145/1718487.1718492

[2] Leonid Boytsov, Tianyi Lin, Fangwei Gao, Yutian Zhao, Jeffrey Huang, and Eric Nyberg. 2022. Understanding Performance of Long-Document Ranking Models through Comprehensive Evaluation and Leaderboarding. *CoRR* abs/2207.01262 (2022). https://doi.org/10.48550/arXiv.2207.01262 arXiv:2207.01262

[3] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *CoRR* abs/1910.10687 (2019). arXiv:1910.10687 http://arxiv.org/abs/1910.10687

[4] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (Eds.). ACM, 2353–2359. https://doi.org/10.1145/3477495.3531857

[5] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2288–2292. https://doi.org/10.1145/3404835.3463098

[6] Samuel J. Huston and W. Bruce Croft. 2014. A Comparison of Retrieval Models using Term Dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang (Eds.). ACM, 111–120. https://doi.org/10.1145/2661829.2661894

[7] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *CoRR* abs/2008.09093 (2020). arXiv:2008.09093 https://arxiv.org/abs/2008.09093

[8] Jimmy Lin. 2021. A proposed conceptual framework for a representational approach to information retrieval. *SIGIR Forum* 55, 2, 4:1–4:29. https://doi.org/10.1145/3527546.3527552

[9] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *CoRR* abs/2106.14807 (2021). arXiv:2106.14807 https://arxiv.org/abs/2106.14807

[10] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1573–1576. https://doi.org/10.1145/3397271.3401262

[11] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 1101–1104. https://doi.org/10.1145/3331184.3331317

[12] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2429–2436. https://doi.org/10.1145/3404835.3463254

[13] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 1723–1727. https://doi.org/10.1145/3404835.3463030

[14] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait (Eds.). ACM, 472–479. https://doi.org/10.1145/1076034.1076115

[15] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 13982)*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer, 101–116. https://doi.org/10.1007/978-3-031-28241-6_7

[16] 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 the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings, Vol. 1773)*, Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[17] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990. https://doi.org/10.18653/v1/D19-1410

[18] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[19] Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando (Eds.). ACM, 295–302. https://doi.org/10.1145/1277741.1277794

[20] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Proceedings of The Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA (NIST Special Publication)*, Ellen M. Voorhees and Lori P. Buckland (Eds.). National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec13/papers/ROBUST.OVERVIEW.pdf

[21] Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2021. Comparing Score Aggregation Approaches for Document Retrieval with Pretrained Transformers. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12657)*, Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani (Eds.). Springer, 150–163. https://doi.org/10.1007/978-3-030-72240-1_11