SPLATE: Sparse Late Interaction Retrieval

Thibault Formal
Naver Labs Europe
Meylan, France
thibault.formal@naverlabs.com

Stéphane Clinchant
Naver Labs Europe
Meylan, France
stephane.clinchant@naverlabs.com

Hervé Déjean
Naver Labs Europe
Meylan, France
herve.dejean@naverlabs.com

Carlos Lassance*
Cohere
Toronto, Canada
cadurosar@gmail.com

ABSTRACT

The late interaction paradigm introduced with ColBERT stands out in the neural Information Retrieval space, offering a compelling effectiveness-efficiency trade-off across many benchmarks. Efficient late interaction retrieval is based on an optimized multi-step strategy, where an approximate search first identifies a set of candidate documents to re-rank exactly. In this work, we introduce SPLATE, a simple and lightweight adaptation of the ColBERTv2 model which learns an "MLM adapter", mapping its *frozen* token embeddings to a sparse vocabulary space with a partially learned SPLADE module. This allows us to perform the candidate generation step in late interaction pipelines with traditional sparse retrieval techniques, making it particularly appealing for running ColBERT in CPU environments. Our SPLATE ColBERTv2 pipeline achieves the same effectiveness as the PLAID ColBERTv2 engine by re-ranking 50 documents that can be retrieved under 10*ms*.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**.

KEYWORDS

Late Interaction, Sparse Retrieval, Hybrid Models

ACM Reference Format:
Thibault Formal, Stéphane Clinchant, Hervé Déjean, and Carlos Lassance. 2024. SPLATE: Sparse Late Interaction Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3626772.3657968

1 INTRODUCTION

In the landscape of neural retrieval models based on Pre-trained Language Models (PLMs), the late interaction paradigm – introduced with the ColBERT model [16] – delivers state-of-the-art results across many benchmarks. ColBERT – and its variants [11, 12, 21, 25, 33, 38, 45, 48] – enjoys many good properties, ranging from interpretability [8, 46] to robustness [10, 26, 47, 49]. The fine-grained *interaction* mechanism, based on a token-level dense vector representation of documents and queries, alleviates the inherent limitation of single-vector models such as DPR [15]. Due to its *MaxSim* formulation, late interaction retrieval requires a dedicated multi-step search pipeline. In the meantime, Learned Sparse Retrieval [30] has emerged as a new paradigm to reconcile the traditional search infrastructure with PLMs. In particular, SPLADE models [6, 7, 9] exhibit strong in-domain and zero-shot capabilities at a fraction of the cost of late interaction approaches – both in terms of memory footprint and search latency [18, 19, 34, 35].

In this work, we draw a parallel between these two lines of works, and show how we can simply "adapt" ColBERTv2 *frozen* representations with a light SPLADE module to effectively map queries and documents in a sparse vocabulary space. Based on this idea, we introduce SPLATE – for **SP**arse **LATE** interaction – as an alternative approximate scoring method for late interaction pipelines. Contrary to optimized engines like PLAID [37], our method relies on traditional sparse techniques, making it particularly appealing to run ColBERT in mono-CPU environments.

2 RELATED WORKS

Efficient Late Interaction Retrieval. Late interaction retrieval is a powerful paradigm, that requires complex engineering to scale up efficiently. Specifically, it resorts to a multi-step pipeline, where an initial set of candidate documents is retrieved based on approximate scores [16]. While it is akin to the traditional *retrieve-and-rank* pipeline in IR, it still fundamentally differs in that the same (PLM) model is used for both steps[1]. Late interaction models offer advantages over cross-encoders because they allow for pre-computation of document representations offline, thus improving efficiency in theory. However, this comes at the cost of storing large indexes of dense term representations. Various optimizations of the ColBERT engine have thus been introduced [5, 12, 20, 23, 27, 29, 33, 37, 38, 41, 43]. ColBERTv2 [38] refines the original ColBERT by introducing residual compression to reduce the space footprint of late interaction approaches. Yet, search speed remains a bottleneck, mostly due to the large number of candidates to re-rank exactly ($> 10k$) [27]. *Santhanam et al.* identify the major bottlenecks – in

*Work done while at Naver Labs Europe.

[1]On the contrary, a standard DPR [15] ≫ MonoBERT [31] pipeline would require feeding the query *twice* to a PLM at inference time.

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657968

terms of search speed – of the vanilla ColBERTv2 pipeline, and introduce PLAID [37], a new optimized late interaction pipeline that can largely reduce the number of candidate passages without impacting ColBERTv2's effectiveness. In particular, PLAID candidate generation is based on three steps that leverage centroid interaction and centroid pruning – emulating traditional Bag-of-Words (BoW) retrieval – as well as dedicated CUDA kernels. It reduces the large number of candidate documents to re-rank, greatly offloading subsequent steps (index lookup, decompression, and scoring).

Hybrid Models. Several works have identified similarities between the representations learned by different neural ranking models. For instance, UNIFIER [40] jointly learns dense and sparse single-vector bi-encoders by sharing intermediate transformer layers. Similarly, the BGE-M3 embedding model [3] can perform dense, multi-vector, and sparse retrieval indifferently. SparseEmbed [17] extends SPLADE with dense contextual embeddings – borrowing ideas from ColBERT and COIL [11]. SLIM [22] adapts ColBERT to perform late interaction on top of SPLADE-like representations – making it fully compatible with traditional search techniques. *Ram et al.* [36] show that mapping representations of a dense bi-encoder to the vocabulary space – via the Masked Language Modeling (MLM) head – can also be used for interpretation purposes.

3 METHOD

SPLATE is motivated by two core ideas: (1) PLAID [37] draws inspiration from traditional BoW retrieval to optimize the late interaction pipeline; (2) dense embeddings can seemingly be mapped to the vocabulary space [36]. Rather than proposing a new standalone model, we show how SPLATE can be used to approximate the candidate generation step in late interaction retrieval, by bridging the gap between sparse and dense models.

Adapting Representations. SPLATE builds on the similarities between the representations learned by sparse and dense IR models. For instance, *Ram et al.* [36] show that mapping representations of a dense bi-encoder with the MLM head can produce meaningful BoW. We take one step further and hypothesize that effective sparse models can be derived – or at least *adapted* – from *frozen* embeddings of dense IR models in a SPLADE-like fashion. We, therefore, propose to "branch" an MLM head on top of a *frozen* ColBERT model.

SPLATE. Given ColBERT's contextual embeddings $(h_i)_{i\in t}$ of an input query or document t, we can define a simple "adapted" MLM head, by linearly mapping *transformed* representations back to the vocabulary. Inspired by Adapter modules [14, 32], SPLATE thus simply adapts *frozen* representations $(h_i)_{i\in t}$ by learning a simple two-layer MLP, whose output is recombined in a residual fashion before "MLM" vocabulary projection:

$$w_{iv} = (h_i + MLP_{\boldsymbol{\theta}}(h_i))^T E_v + b_v \tag{1}$$

where w_i corresponds to an unnormalized log-probability distribution over the vocabulary $\mathcal{V}$ for the token t_i, E_v is the (Col)BERT input embedding for the token v and b_v is a token-level bias. The residual guarantees a near-identity initialization – making training stable [14]. We can then derive sparse SPLADE vectors as follows:

$$w_v = \max_{i\in t} \log\left(1 + \text{ReLU}(w_{iv})\right), \quad v \in \{1, ..., |\mathcal{V}|\} \tag{2}$$

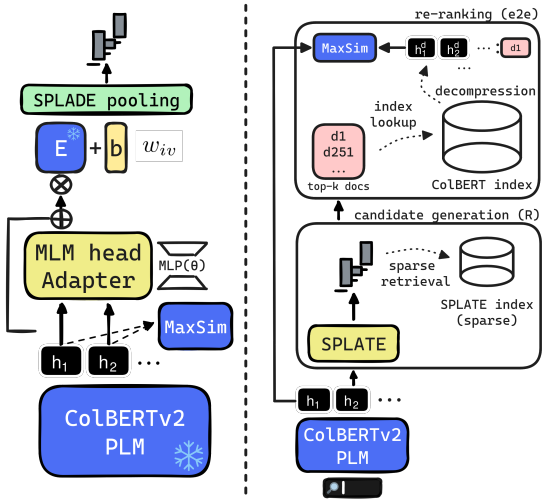

Figure 1: *(Left)* **SPLATE relies on the same representations $(h_i)_{i\in t}$ to learn sparse BoW with SPLADE (candidate generation) and to compute late interactions (re-ranking).** *(Right)* **Inference: SPLATE ColBERTv2 maps the representations of the query tokens to a sparse vector, which is used to retrieve k documents from a pre-computed sparse index (R setting). In the *e2e* setting, representations are gathered from the ColBERT index to re-rank the candidates exactly with *MaxSim*.**

We then train the parameters of the MLM head $(\boldsymbol{\theta}, \boldsymbol{b})$ with distillation based on the derived SPLADE vectors to reproduce ColBERT's scores – see Section 4. Our approach is very light, as the ColBERT backbone model is entirely frozen – including the (tied) projection layer E. In our default setting, the MLP first down-projects representations by a factor of two, then up-projects back to the original dimension. This corresponds to a latent dimension of $768/2 = 384$ – early experiments indicate that the choice of this hyperparameter is not critical – and amounts to roughly $0.6M$ trainable parameters only (yellow blocks in Figure 1, (*Left*)).

Efficient Candidate Generation for Late Interaction. By adapting ColBERT's frozen dense representations with a SPLADE module, SPLATE aims to approximate late interaction scoring with an efficient sparse dot product. Thus, *the same representations* $(h_i)_{i\in t}$ can function in both retrieval (SPLATE module) and re-ranking (ColBERT's *MaxSim*) scenarios – *requiring a single transformer inference step* on query and document sides. Thus, it becomes possible to replace the existing candidate generation step in late retrieval pipelines such as PLAID with traditional sparse retrieval to efficiently provide ColBERT with documents to re-rank. SPLATE is therefore not a model *per se*, but rather offers an alternative implementation to late-stage pipelines by bridging the gap between sparse and dense models. SPLATE however differs from PLAID in various aspects:

- While PLAID implicitly derives sparse BoW representations from ColBERTv2's centroid mapping, SPLATE explicitly learns

such representations by adapting a pseudo-MLM head to ColBERT frozen representations. The approximate step becomes supervised rather than (yet efficiently) "engineered".
- The candidate generation can benefit from the long-standing efficiency of inverted indexes and query processing techniques such as MaxScore [44] or WAND [2], making end-to-end ColBERT more "CPU-friendly" – see Table 1.
- It is more controllable and directly amenable to all sorts of recent optimizations for learned sparse models [18, 19].
- ColBERT's pipeline becomes even more interpretable, as SPLATE's candidate generation simply operates in the vocabulary space – rather than representing documents as a lightweight bag of centroids – see Table 3 for examples.

Nonetheless, SPLATE requires an additional – although light – training round for the parameters of the Adapter module. It also requires indexing SPLATE's sparse document vectors, therefore adding a small memory footprint overhead[2]. Also, note that hybrid approaches like BGE-M3 [3] – that can output sparse and multi-vector representations – could in theory be used in late interaction pipelines. However, SPLATE is directly optimized to approximate ColBERTv2, and we leave for future work the study of jointly training the candidate generation and re-ranking modules.

4 EXPERIMENTS

Setting. We initialize SPLATE with ColBERTv2 [38] weights which are kept *frozen*. We rely on top-$k_{q,d}$ pooling to obtain respectively query and document BoW SPLADE representations[3]. We train the MLM parameters $(\boldsymbol{\theta}, \boldsymbol{b})$ on the MS MARCO passage dataset [1], using both distillation and hard negative sampling. More specifically, we distill ColBERTv2's scores based on a weighted combination of marginMSE [13] and KLDiv [24] losses for 3 epochs. We set the batch size to 24, and select 20 hard negatives per query – coming from ColBERTv2's top-1000. By using ColBERTv2 as both the teacher and the source of hard negatives, SPLATE aims to approximate late interaction with sparse retrieval. SPLATE models are trained with the SPLADE codebase on 2 Tesla V100 GPUs with 32GB memory in less than two hours[4]. SPLATE can be evaluated as a standalone sparse retriever (R), but more interestingly in an end-to-end late interaction pipeline (e2e) where it provides ColBERTv2 with candidates to re-rank (see Figure 1, (*Right*))[5]. For the former, we rely on the PISA engine [28] to conduct sparse retrieval with block-max WAND and provide latency measurements as the Mean Response Time (MRT), i.e., the average search latency measured on the MS MARCO dataset using one core of an Intel(R) Xeon(R) Gold 6338 CPU @ 2.00GHz CPU. For the latter, we perform on-the-fly re-ranking with the ColBERT library[6]. Note that naive re-ranking with ColBERT is sub-optimal – compared to pipelines that precompute document term embeddings. We leave the end-to-end latency measurements for future work – but we believe the integration of SPLATE into ColBERT's pipelines such as PLAID should be seamless, as it would only require modifying the candidate generation step. We evaluate models on the MS MARCO dev set and the TREC DL19 queries [4] (in-domain), and provide out-of-domain evaluations on the 13 readily available BEIR datasets [42], as well as the test pooled Search dataset of the LoTTE benchmark [38].

The following experiments investigate three different Research Questions: (1) How does the sparsity of SPLATE vectors affect latency and re-ranking performance? (2) How accurate SPLATE candidate generation is compared to ColBERTv2? (3) How does it perform overall for in-domain and out-of-domain scenarios?

Latency Results. Table 1 reports in-domain results on MS MARCO, in both retrieval-only (R) and end-to-end (e2e) settings. Overall, the results show that it is possible to "convert" a frozen ColBERTv2 model to an effective SPLADE, with a lightweight residual adaptation of its token embeddings. We consider several SPLATE models trained with varying pooling sizes (k_q, k_d) – those parameters controlling the size of the query and document representations. We observe the standard effectiveness-efficiency trade-off for SPLADE, where pooling affects both the performance and average latency. These results indicate that one can easily control the latency of the candidate generation step by selecting appropriate pooling sizes. *However, after re-ranking with ColBERTv2, all the models perform comparably*, which is interesting from an efficiency perspective, as it becomes possible to use very lightweight models to cheaply provide candidates (e.g., as low as $2.9ms$ Mean Response Time), while achieving performance on par with the original ColBERTv2 (see Table 2). For comparison, the end-to-end latency reported in PLAID [37] (single CPU core, less conservative setting with $k = 10$) is around $186ms$ on MS MARCO. Given that candidate generation accounts for around two-thirds of the complete pipeline [37], SPLATE thus offers an interesting alternative for running ColBERT on mono-CPU environments.

Table 1: Retrieval latency (MRT), retrieval-only (R) and end-to-end (e2e, $k = 50$) MRR@10 on MS MARCO dev.

(k_q, k_d)	R		e2e ($k = 50$)
	MRT (ms)	MRR@10	MRR@10
$(5, 30)$	2.9	34.5	39.5
$(5, 50)$	4.3	35.5	39.7
$(5, 100)$	7.4	35.6	39.8
$(10, 100)$	24.0	36.7	40.0
$(20, 200)$	106.0	37.4	40.0

Approximation Quality. To assess the quality of SPLATE approximation, we compare the top-k passages retrieved by PLAID ColBERTv2 to the ones retrieved by SPLATE (R). We report in Figure 2 the average fraction $R(k)$ of documents in SPLATE's top-k' that also appear in the top-k documents retrieved by ColBERTv2 on MS MARCO, for $k \in \{10, 100\}$ and $k' = i \times k, i \in \{1, ..., 5\}$. When $k = 10$, SPLATE can retrieve more than 90% of ColBERTv2's documents in its top-50 ($i = 5$), for all levels of (k_q, k_d). This explains the ability of SPLATE to fully recover ColBERT's performance by

[2]Note however that this is negligible compared to ColBERT's index – for instance, the MS MARCO PISA index for the SPLATE model in Table 2 weighs around 2.2GB.
[3]While SPLADE is usually trained with sparse regularization, top-$k_{q,d}$ was shown to be almost as effective – while being much simpler [30].
[4]https://github.com/naver/splade
[5]Note that SPLATE (*e2e*) is an alternative implementation of ColBERTv2. We use SPLATE (resp. PLAID) or SPLATE ColBERTv2 (resp. PLAID ColBERTv2) indifferently.
[6]https://github.com/stanford-futuredata/ColBERT

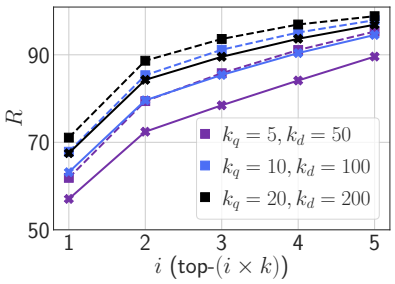

Figure 2: Candidate generation approximate accuracy on MS MARCO dev – SPLATE (R). Dotted lines (■) represent $R(10)$, solid lines represent (✖) $R(100)$.

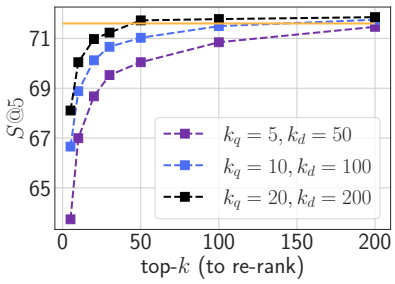

Figure 3: Impact of k and (k_q, k_d) on SPLATE (e2e) ouf-of-domain performance – *Success*@5 on LoTTE (test pooled Search). The orange line represents ColBERTv2.

re-ranking a handful of documents (e.g., 50 only). We additionally observe that the quality of approximation falls short for efficient models (i.e., lower (k_q, k_d)) when k is higher.

Figure 3 further reports the performance of SPLATE (e2e) on out-of-domain. We observe similar trends, where increasing both the number k of documents to re-rank and (k_q, k_d) leads to better generalization. Overall, re-ranking only 50 documents provides a good trade-off across all settings – echoing previous findings [27, 37]. Yet, the most efficient scenario ($(k_q, k_d) = (5, 50)$, $k = 10$) still leads to impressive results: 38.4 MRR@10 on MS MARCO dev (not shown), 70.0 S@5 on LoTTE (purple line on Figure 3).

Overall Results. Finally, Table 2 compares SPLATE ColBERTv2 with the reference points ColBERTv2 [38] and PLAID ColBERTv2 ($k = 1000$) [37] – in both R and e2e settings. We also include results from SPLADE++ [7], as well as the hybrid methods SparseEmbed [17] and SLIM++ [22] – even though they are not entirely comparable to SPLATE. While SparseEmbed and SLIM introduce new models, SPLATE rather proposes an alternative implementation to ColBERT's late retrieval pipeline. We further report the two baselines consisting of retrieving documents with BM25 (resp. SPLADE++) and re-ranking those with ColBERTv2 (BM25 ≫ C and S ≫ C respectively, with $k = 50$). Note that we expect SPLATE to perform in between, as BM25 ≫ C relies on a less effective retriever, while S ≫ C fundamentally differs from SPLATE, as it is based on two different models. Specifically, it requires feeding the query to a PLM *twice* at inference time. Overall, SPLATE (R) is effective as a standalone retriever (e.g., reaching almost 37 MRR@10 on MS MARCO dev). On the other hand, SPLATE (e2e) performs comparably to ColBERTv2 and PLAID on MS MARCO, BEIR, and LoTTE. Additionally, we conducted a meta-analysis against PLAID with RANGER [39] over the 13 BEIR datasets, and found no statistical differences on 10 datasets, and statistical improvement (resp. loss) on one (resp. two) dataset(s). Finally, we provide in Table 3 some examples of predicted BoW for queries in MS MARCO dev – highlighting the interpretable nature of the retrieval step in SPLATE-based ColBERT's pipeline.

Table 2: Evaluation of SPLATE with $(k_q, k_d) = (10, 100)$ and $k = 50$. ^{abcde} denote significant improvements over the corresponding rows, for a paired t-test with p-value=0.01 and Bonferroni correction (MS MARCO dev set and DL19). PLAID ColBERTv2 [37] ($k = 1000$) reports the *dev* LoTTE* S@5.

	MS MARCO	DL19		BEIR	LoTTE
	MRR@10	nDCG@10	R@1k	nDCG@10	S@5
► Sparse/Hybrid					
SPLADE++ [7]	38.0	73.2	87.5	50.7	-
SparseEmbed [17]	39.2	-	-	50.9	-
SLIM++ [22]	40.4	71.4	84.2	49.0	-
► References					
ColBERTv2 [38]	39.7	-	-	49.7	71.6
(a) PLAID ColBERTv2 [37]	39.8^{bd}	74.6	85.2^{b}	-	69.6^{*}
(b) BM25 ≫ C ($k = 50$)	34.3	68.7	73.9	49.0	62.8
(c) S ≫ C ($k = 50$)	40.4^{bd}	74.4	87.5^{b}	49.9	72.0
► SPLATE ColBERTv2 ($k = 50$)					
(d) SPLATE (R)	36.7^{b}	72.9	84.4^{b}	46.5	66.7
(e) SPLATE (e2e)	40.0^{bd}	74.2	84.4^{b}	49.6	71.0

Table 3: BoW SPLATE representations for queries in the MS MARCO dev set with $(k_q, k_d) = (10, 100)$ (model from Table 2).

SPLATE BoW
$Q \rightarrow$ "*what is the medium for an artisan*"
► (*medium*, 2.2), (*art*, 1.8), (*##isan*, 1.7), (*media*, 1.1), (*craftsman*, 0.9), (*arts*, 0.6), (*carpenter*, 0.6), (*artist*, 0.5), (*##vre*, 0.4), (*draper*, 0.3)
$Q \rightarrow$ "*treating tension headaches without medication*"
► (*headache*, 2.1), (*tension*, 1.8), (*without*, 1.6), (*treatment*, 1.5), (*treat*, 1.4), (*medication*, 1.3), (*drug*, 0.8), (*baker*, 0.7), (*no*, 0.6), (*stress*, 0.5)

5 CONCLUSION

We propose SPLATE, a new lightweight candidate generation technique simplifying ColBERTv2's candidate generation for late interaction retrieval. SPLATE adapts ColBERTv2's frozen embeddings to conduct efficient sparse retrieval with SPLADE. When evaluated end-to-end, the SPLATE implementation of ColBERTv2 performs comparably to ColBERTv2 and PLAID on several benchmarks, by re-ranking a handful of documents. The sparse term-based nature of the candidate generation step makes it particularly appealing in mono-CPU environments efficiency-wise. Beyond optimizing late interaction retrieval, our work opens the path to a deeper study of the link between the representations trained from different architectures.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *InCoCo@NIPS*.

[2] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient Query Evaluation Using a Two-Level Retrieval Process. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 426–434. https://doi.org/10.1145/956863.956944

[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL]

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen Voorhees. 2019. Overview of the TREC 2019 deep learning track. In *TREC 2019*.

[5] Joshua Engels, Benjamin Coleman, Vihan Lakshman, and Anshumali Shrivastava. 2023. DESSERT: An Efficient Algorithm for Vector Set Search with Vector Set Queries. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=kXfrlWXLwH

[6] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. arXiv:2109.10086 [cs.IR]

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2353–2359.

[8] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2020. A White Box Analysis of ColBERT. arXiv:2012.09650 [cs.IR]

[9] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proc. SIGIR*. 2288–2292.

[10] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2022. Match Your Words! A Study of Lexical Matching in Neural Information Retrieval. In *Advances in Information Retrieval*, Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty (Eds.). Springer International Publishing, Cham, 120–127.

[11] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proc. NAACL-HLT*. 3030–3042.

[12] Sebastian Hofstätter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing Neural Bag of Whole-Words with ColBERTer: Contextualized Late Interactions Using Enhanced Reduction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 737–747. https://doi.org/10.1145/3511808.3557367

[13] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2021. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv:2010.02666 [cs.IR]

[14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2790–2799. https://proceedings.mlr.press/v97/houlsby19a.html

[15] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[16] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proc. SIGIR*. 39–48.

[17] Weize Kong, Jeffrey M. Dudek, Cheng Li, Mingyang Zhang, and Mike Bendersky. 2023. SparseEmbed: Learning Sparse Lexical Representations with Contextual Embeddings for Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*.

[18] Carlos Lassance and Stéphane Clinchant. 2022. An Efficiency Study for SPLADE Models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2220–2226. https://doi.org/10.1145/3477495.3531833

[19] Carlos Lassance, Simon Lupart, Hervé Déjean, Stéphane Clinchant, and Nicola Tonellotto. 2023. A Static Pruning Study on Sparse Neural Retrievers. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1771–1775. https://doi.org/10.1145/3539618.3591941

[20] Carlos Lassance, Maroua Maachou, Joohee Park, and Stéphane Clinchant. 2022. Learned Token Pruning in Contextualized Late Interaction over BERT (ColBERT). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2232–2236. https://doi.org/10.1145/3477495.3531835

[21] Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y. Zhao. 2023. Rethinking the Role of Token Retrieval in Multi-Vector Retrieval. arXiv:2304.01982 [cs.CL]

[22] Minghan Li, Sheng-Chieh Lin, Xueguang Ma, and Jimmy Lin. 2023. SLIM: Sparsified Late Interaction for Multi-Vector Retrieval with Inverted Indexes. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. https://doi.org/10.1145/3539618.3591977

[23] Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 11891–11907. https://doi.org/10.18653/v1/2023.acl-long.663

[24] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Association for Computational Linguistics, Online, 163–173. https://doi.org/10.18653/v1/2021.repl4nlp-1.17

[25] Weizhe Lin, Jinghong Chen, Jingbiao Mei, Alexandru Coca, and Bill Byrne. 2023. Fine-grained Late-interaction Multi-modal Retrieval for Retrieval Augmented Visual Question Answering. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=IWWWulAX7g

[26] Simon Lupart, Thibault Formal, and Stéphane Clinchant. 2023. MS-Shift: An Analysis of MS MARCO Distribution Shifts on Neural Retrieval. In *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 636–652.

[27] Craig Macdonald and Nicola Tonellotto. 2021. On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieval. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 3318–3322. https://doi.org/10.1145/3459637.3482156

[28] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019*. 50–56. http://ceur-ws.org/Vol-2409/docker08.pdf

[29] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Multi-vector Dense Retrieval with Bit Vectors. In *Advances in Information Retrieval*, Nazli Goharian, Nicola Tonellotto, Yulan He, Aldo Lipani, Graham McDonald, Craig Macdonald, and Iadh Ounis (Eds.). Springer Nature Switzerland, Cham, 3–17.

[30] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *European Conference on Information Retrieval*. Springer, 101–116.

[31] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:Preprint: arXiv:1901.04085

[32] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 7654–7673. https://doi.org/10.18653/v1/2020.emnlp-main.617

[33] Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y. Zhao. 2022. Multi-Vector Retrieval as Sparse Alignment. arXiv:2211.01267 [cs.CL]

[34] Yifan Qiao, Yingrui Yang, Shanxiu He, and Tao Yang. 2023. Representation Sparsification with Hybrid Thresholding for Fast SPLADE-based Document Retrieval. *arXiv preprint arXiv:2306.11293* (2023).

[35] Yifan Qiao, Yingrui Yang, Haixin Lin, and Tao Yang. 2023. Optimizing Guided Traversal for Fast Learned Sparse Retrieval. In *Proceedings of the ACM Web Conference 2023*. 3375–3385.

[36] Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What Are You Token About? Dense Retrieval as Distributions Over the Vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 2481–2498. https://doi.org/10.18653/v1/2023.acl-long.140

[37] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022. PLAID: An Efficient Engine for Late Interaction Retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York,

NY, USA, 1747–1756. https://doi.org/10.1145/3511808.3557325

[38] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488* (2021).

[39] Mete Sertkan, Sophia Althammer, and Sebastian Hofstätter. 2023. Ranger: A Toolkit for Effect-Size Based Multi-Task Evaluation. *arXiv preprint arXiv:2305.15048* (2023).

[40] Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Kai Zhang, and Daxin Jiang. 2023. UnifieR: A Unified Retriever for Large-Scale Retrieval. arXiv:2205.11194 [cs.IR]

[41] Susav Shrestha, Narasimha Reddy, and Zongwang Li. 2023. ESPN: Memory-Efficient Multi-Vector Information Retrieval. arXiv:2312.05417 [cs.IR]

[42] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[43] Nicola Tonellotto and Craig Macdonald. 2021. Query Embedding Pruning for Dense Retrieval. In *Proc. CIKM*. 3453–3457.

[44] Howard Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. *Inf. Process. Manage.* 31, 6 (nov 1995), 831–850. https://doi.org/10.1016/0306-4573(95)00020-H

[45] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval* (Virtual Event, Canada) *(ICTIR '21)*. Association for Computing Machinery, New York, NY, USA, 297–306. https://doi.org/10.1145/3471158.3472250

[46] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2023. Reproducibility, Replicability, and Insights into Dense Multi-Representation Retrieval Models: From ColBERT to Col*. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 2552–2561. https://doi.org/10.1145/3539618.3591916

[47] Orion Weller, Dawn Lawrie, and Benjamin Van Durme. 2023. NevIR: Negation in Neural Information Retrieval. arXiv:2305.07614 [cs.IR]

[48] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. 2022. FILIP: Fine-grained Interactive Language-Image Pre-Training. In *International Conference on Learning Representations*. https://openreview.net/forum?id=cpDhcsEDC2

[49] Jingtao Zhan, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Evaluating Interpolation and Extrapolation Performance of Neural Retrieval Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 2486–2496. https://doi.org/10.1145/3511808.3557312