SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking

Thibault Formal
Naver Labs Europe
Meylan, France
Sorbonne Université, LIP6
Paris, France
thibault.formal@naverlabs.com

Benjamin Piwowarski
Sorbonne Université, CNRS, LIP6
Paris, France
benjamin.piwowarski@lip6.fr

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

ABSTRACT

In neural Information Retrieval, ongoing research is directed towards improving the first retriever in ranking pipelines. Learning dense embeddings to conduct retrieval using efficient approximate nearest neighbors methods has proven to work well. Meanwhile, there has been a growing interest in learning *sparse* representations for documents and queries, that could inherit from the desirable properties of bag-of-words models such as the exact matching of terms and the efficiency of inverted indexes. In this work, we present a new first-stage ranker based on explicit sparsity regularization and a log-saturation effect on term weights, leading to highly sparse representations and competitive results with respect to state-of-the-art dense and sparse methods. Our approach is simple, trained end-to-end in a single stage. We also explore the trade-off between effectiveness and efficiency, by controlling the contribution of the sparsity regularization.

KEYWORDS

neural networks, indexing, sparse representations, regularization

ACM Reference Format:
Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The release of large pre-trained language models like BERT [7] has shaken-up Natural Language Processing and Information Retrieval. These models have shown a strong ability to adapt to various tasks by simple fine-tuning. At the beginning of 2019, *Nogueira and Cho* [17] achieved state-of-the-art results – by a large margin – on the MS MARCO passage re-ranking task, paving the way for LM-based neural ranking models. Because of strict efficiency requirements, these models have initially been used as re-rankers in a two-stage ranking pipeline, where first-stage retrieval – or candidate generation – is conducted with bag-of-words models (e.g. BM25) that rely on inverted indexes. While BOW models remain strong baselines [27], they suffer from the long standing vocabulary mismatch problem, where relevant documents might not contain terms that appear in the query. Thus, there have been attempts to substitute standard BOW approaches by learned (neural) rankers. Designing such models poses several challenges regarding efficiency and scalability: therefore there is a need for methods where most of the computation can be done offline and online inference is fast. Dense retrieval with approximate nearest neighbors search has shown impressive results [8, 15, 26], but is still combined with BOW models because of its inability to explicitly model term matching. Hence, there has recently been a growing interest in learning *sparse representations* for queries and documents [1, 4, 19, 28, 29]. By doing so, models can inherit from the desirable properties of BOW models like exact-match of (possibly latent) terms, efficiency of inverted indexes and interpretability. Additionally, by modeling implicit or explicit (latent, contextualized) *expansion* mechanisms – similarly to standard expansion models in IR – these models can reduce the vocabulary mismatch.

The contributions of this paper are threefold: (1) we build upon SparTerm [1], and show that a mild tuning of hyperparameters brings improvements that largely outperform the results reported in the original paper; (2) we propose the SParse Lexical AnD Expansion (SPLADE) model, based on a logarithmic activation and sparse regularization. SPLADE performs an efficient document expansion [1, 16], with competitive results with respect to complex training pipelines for dense models like ANCE [26]; (3) finally, we show how the sparsity regularization can be controlled to influence the trade-off between efficiency (in terms of the number of floating-point operations) and effectiveness.

2 RELATED WORKS

Dense retrieval based on BERT Siamese models [22] has become the standard approach for candidate generation in Question Answering and IR [8, 10, 12, 15, 25]. While the backbone of these models remains the same, recent works highlight the critical aspects of the training strategy to obtain state-of-the-art results, ranging from improved negative sampling [8, 25] to distillation [11, 15]. ColBERT [13] pushes things further: the postponed token-level interactions allow to efficiently apply the model for first-stage retrieval, benefiting of the effectiveness of modeling fine-grained interactions, at the cost of storing embeddings for each (sub)term – raising concerns about the actual scalability of the approach for large collections. To the best of our knowledge, very few studies have discussed the impact of using *approximate* nearest neighbors

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(ANN) search on IR metrics [2, 23]. Due to the moderate size of the MS MARCO collection, results are usually reported with an *exact*, brute-force search, therefore giving no indication on the effective computing cost.

An alternative to dense indexes is term-based ones. Building on standard BOW models, *Zamani et al.* first introduced SNRM [28]: the model embeds documents and queries in a sparse high-dimensional latent space by means of ℓ_1 regularization on representations. However, SNRM effectiveness remains limited and its efficiency has been questioned [20]. More recently, there have been attempts to transfer the knowledge from pre-trained LM to sparse approaches. Based on BERT, DeepCT [4–6] focused on learning contextualized term weights in the full vocabulary space – akin to BOW term weights. However, as the vocabulary associated with a document remains the same, this type of approach does not solve the vocabulary mismatch, as acknowledged by the use of query expansion for retrieval [4]. A first solution to this problem consists in expanding documents using generative approaches such as doc2query [19] and docTTTTTquery [18] to predict expansion words for documents. The document expansion adds new terms to documents – hence fighting the vocabulary mismatch – as well as repeats existing terms, implicitly performing re-weighting by boosting important terms. These methods are however limited by the way they are trained (predicting queries), which is indirect in nature and limit their progress. A second solution to this problem, that has been chosen by recent works such as [1, 16, 29], is to estimate the importance of each term of the vocabulary *implied by* each term of the document, i.e. to compute an interaction matrix between the document or query tokens and all the tokens from the vocabulary. This is followed by an aggregation mechanism (roughly sum for SparTerm [1], max for EPIC [16] and SPARTA [29]), that allows to compute an importance weight for each term of the vocabulary, for the full document or query. However, EPIC and SPARTA (document) representations are not sparse enough by construction – unless resorting on top-k pooling – contrary to SparTerm, for which fast retrieval is thus possible. Furthermore, the latter does not include (like SNRM) an *explicit* sparsity regularization, which hinders its performance. Our SPLADE model relies on such regularization, as well as other key changes, that boost both the efficiency and the effectiveness of this type of models.

3 SPARSE LEXICAL REPRESENTATIONS FOR FIRST-STAGE RANKING

In this section, we first describe in details the SparTerm model [1], before presenting our model named SPLADE.

3.1 SparTerm

SparTerm predicts term importance – in BERT WordPiece vocabulary ($|V| = 30522$) – based on the logits of the Masked Language Model (MLM) layer. More precisely, let us consider an input query or document sequence (after WordPiece tokenization) $t = (t_1, t_2, ..., t_N)$, and its corresponding BERT embeddings $(h_1, h_2, ..., h_N)$. We consider the importance w_{ij} of the token j (vocabulary) for a token i (of the input sequence):

$$w_{ij} = \text{transform}(h_i)^T E_j + b_j \quad j \in \{1, ..., |V|\} \tag{1}$$

where E_j denotes the BERT input embedding for token j, b_j is a token-level bias, and transform(.) is a linear layer with GeLU activation and LayerNorm. Note that Eq. 1 is equivalent to the MLM prediction, thus it can be also be initialized from a pre-trained MLM model. The final representation is then obtained by summing importance predictors over the input sequence tokens, after applying ReLU to ensure the positivity of term weights:

$$w_j = g_j \times \sum_{i \in t} \text{ReLU}(w_{ij}) \tag{2}$$

where g_j is a binary mask (gating) described latter. The above equation can be seen as a form of query/document *expansion*, as observed in [1, 16], since for each token of the vocabulary the model predicts a new weight w_j. SparTerm [1] introduces two sparsification schemes that turn off a large amount of dimensions in query and document representations, allowing to efficiently retrieve from an inverted index:

lexical-only is a BOW masking, i.e. $g_j = 1$ if token j appears in t, and 0 otherwise;

expansion-aware is a lexical/expansion-aware binary gating mechanism, where g_j is *learned*. To preserve the original input, it is forced to 1 if the token j appears in t.

Let $s(q, d)$ denote the ranking score obtained via dot product between q and d representations from Eq. (2). Given a query q_i, a positive document d_i^+ and a negative document d_i^-, SparTerm is trained by minimzing the following loss:

$$\mathcal{L}_{rank} = -\log \frac{e^{s(q_i, d_i^+)}}{e^{s(q_i, d_i^+)} + e^{s(q_i, d_i^-)}} \tag{3}$$

Limitations. SparTerm expansion-aware gating is somewhat intricate, and the model cannot be trained end-to-end: the gating mechanism is learned beforehand, and *fixed* while fine-tuning the matching model with $\mathcal{L}_{rank}$, therefore preventing the model to learn the optimal sparsification strategy for the ranking task. Moreover, the two lexical and expansion-aware strategies do perform almost equally well, questioning the actual benefits of expansion.

3.2 SPLADE: SParse Lexical AnD Expansion model

In the following, we propose slight, but essential changes to the SparTerm model that dramatically improve its performance.

Model. We introduce a minor change in the importance estimation from Eq. 2, by introducing a log-saturation effect which prevents some terms to dominate and naturally ensures sparsity in representations:

$$w_j = \sum_{i \in t} \log\left(1 + \text{ReLU}(w_{ij})\right) \tag{4}$$

While it is intuitive that using a log-saturation prevents some terms from dominating – drawing a parallel with axiomatic approaches in IR and log(tf) models [9] – the implied sparsity can seem surprising at first, but, according to our experiments, it obtains better experimental results and allows already to obtain sparse solutions *without any regularization.*

Ranking loss. Given a query q_i in a batch, a positive document d_i^+, a (hard) negative document d_i^- (e.g. coming from BM25 sampling), and a set of negative documents in the batch (positive documents from other queries) $\{d_{i,j}^-\}_j$, we consider the ranking loss from [8], which can be interpreted as the maximization of the probability of the document d_i^+ being relevant among the documents d_i^+, d_i^- and $\{d_{i,j}^-\}$:

$$\mathcal{L}_{rank-IBN} = -\log \frac{e^{s(q_i,d_i^+)}}{e^{s(q_i,d_i^+)} + e^{s(q_i,d_i^-)} + \sum_j e^{s(q_i,d_{i,j}^-)}} \quad (5)$$

The *in-batch negatives* (IBN) sampling strategy is widely used for training image retrieval models, and has shown to be effective in learning first-stage rankers [8, 12, 15].

Learning sparse representations. The idea of learning sparse representations for first-stage retrieval dates back to SNRM [28], via ℓ_1 regularization. Later, [20] pointed-out that minimizing the ℓ_1 norm of representations does not result in the most efficient index, as nothing ensures that posting lists are evenly distributed. Note that this is even more true for standard indexes due to the Zipfian nature of the term frequency distribution. To obtain a well-balanced index, *Paria et al.* [20] introduce the FLOPS regularizer, a smooth relaxation of the average number of floating-point operations necessary to compute the score of a document, and hence directly related to the retrieval time. It is defined using a_j as a continuous relaxation of the activation (i.e. the term has a non zero weight) probability p_j for token j, and estimated for documents d in a batch of size N by $\bar{a}_j = \frac{1}{N}\sum_{i=1}^{N} w_j^{(d_i)}$. This gives the following regularization loss

$$\ell_{\texttt{FLOPS}} = \sum_{j \in V} \bar{a}_j^2 = \sum_{j \in V} \left(\frac{1}{N} \sum_{i=1}^{N} w_j^{(d_i)} \right)^2$$

This differs from the ℓ_1 regularization used in SNRM [28] where the $\bar{a}_j$ are not squared: using $\ell_{\texttt{FLOPS}}$ thus pushes down high average term weight values, giving rise to a more balanced index.

Overall loss. We propose to combine the best of both worlds for end-to-end training of sparse, expansion-aware representations of documents and queries. Thus, we discard the binary gating in SparTerm, and instead learn our log-saturated model (Eq. 4) by jointly optimizing ranking and regularization losses:

$$\mathcal{L} = \mathcal{L}_{rank-IBN} + \lambda_q \mathcal{L}_{\texttt{reg}}^q + \lambda_d \mathcal{L}_{\texttt{reg}}^d \quad (6)$$

where $\mathcal{L}_{\texttt{reg}}$ is a sparse regularization (ℓ_1 or $\ell_{\texttt{FLOPS}}$). We use two distinct regularization weights (λ_d and λ_q) for queries and documents – allowing to put more pressure on the sparsity for queries, which is critical for fast retrieval.

4 EXPERIMENTAL SETTING AND RESULTS

We trained and evaluated our models on the MS MARCO passage ranking dataset[1] in the full ranking setting. The dataset contains approximately 8.8M passages, and hundreds of thousands training queries with shallow annotation ($\approx$ 1.1 relevant passages per query in average). The development set contains 6980 queries with similar labels, while the TREC DL 2019 evaluation set provides fine-grained annotations from human assessors for a set of 43 queries [3].

Training, indexing and retrieval. We initialized the models with the `BERT-base` checkpoint. Models are trained with the ADAM optimizer, using a learning rate of $2e^{-5}$ with linear scheduling and a warmup of 6000 steps, and a batch size of 124. We keep the best checkpoint using MRR@10 on a validation set of 500 queries, after training for 150k iterations (note that this is not optimal, as we validate on a re-ranking task). We consider a maximum length of 256 for input sequences. In order to mitigate the contribution of the regularizer at the early stages of training, we follow [20] and use a scheduler for λ, quadratically increasing λ at each training iteration, until a given step (50k in our case), from which it remains constant. Typical values for λ fall between $1e^{-1}$ and $1e^{-4}$. For storing the index, we use a custom implementation based on Python arrays, and we rely on Numba [14] to parallelize retrieval. Models[2] are trained using PyTorch [21] and HuggingFace transformers [24], on 4 Tesla $V100$ GPUs with 32GB memory.

Evaluation. We report Recall@1000 for both datasets, as well as the official metrics MRR@10 and NDCG@10 for MS MARCO dev set and TREC DL 2019 respectively. Since we are essentially interested in the first retrieval step, we do not consider re-rankers based on BERT, and we compare our approach to first stage rankers only – results reported on the MS MARCO leaderboard are thus not comparable to the results presented here. We compare to the following sparse approaches (1) BM25 (2) DeepCT [4] (3) doc2query-T5 [18] (4) and SparTerm [1], as well as state-of-the-art dense approaches ANCE [25] and TCT-ColBERT [15]. We report the results from the original papers. We include a pure lexical SparTerm trained with our ranking pipeline (ST lexical-only). To illustrate the benefits of the log-saturation, we add results for models trained using Eq. (2) instead of Eq. (4) (ST exp-ℓ_1 and ST exp-$\ell_{\texttt{FLOPS}}$). For sparse models, we indicate an estimation of the average number of floating-point operations between a query and a document in Table 1, when available, which is defined as the expectation $\mathbb{E}_{q,d}\left[\sum_{j \in V} p_j^{(q)} p_j^{(d)}\right]$ where p_j is the activation probability for token j in a document d or a query q. It is empirically estimated from a set of approximately 100k development queries, on the MS MARCO collection.

Results are given in Table 1. Overall, we observe that: (1) *our models outperform the other sparse retrieval methods by a large margin (except for recall@1000 on TREC DL);* (2) *the results are competitive with state-of-the-art dense retrieval methods.*

More specifically, our training method for ST lexical-only already outperforms the results of DeepCT as well as the results reported in the original SparTerm paper – including the model using expansion. Thanks to the additional sparse expansion mechanism, we are able to obtain results on par with state-of-the-art dense approaches on MS MARCO dev set (e.g. Recall@1000 close to 0.96 for ST exp-ℓ_1), but with a much bigger average number of FLOPS.

By adding a log-saturation effect to the expansion model, SPLADE greatly increases sparsity – reducing the FLOPS to similar levels than BOW approaches – at no cost on performance when compared to the best first-stage rankers. In addition, we observe the advantage

[1] https://github.com/microsoft/MSMARCO-Passage-Ranking

[2] We made the code public at https://github.com/naver/splade

Table 1: Evaluation on MS MARCO passage retrieval (dev set) and TREC DL 2019

model	MS MARCO dev		TREC DL 2019		FLOPS
	MRR@10	R@1000	NDCG@10	R@1000	
`Dense retrieval`					
Siamese (ours)	0.312	0.941	0.637	0.711	-
ANCE [25]	0.330	0.959	0.648	-	-
TCT-ColBERT [15]	0.335	0.964	0.670	0.720	-
`Sparse retrieval`					
BM25	0.184	0.853	0.506	0.745	0.13
DeepCT [4]	0.243	0.913	0.551	0.756	-
doc2query-T5 [18]	0.277	0.947	0.642	0.827	0.81
ST lexical-only [1]	0.275	0.912	-	-	-
ST expansion [1]	0.279	0.925	-	-	-
`Our methods`					
ST lexical-only	0.290	0.923	0.595	0.774	1.84
ST exp-ℓ_1	0.314	0.959	0.668	0.800	4.62
ST exp-$\ell_{\texttt{FLOPS}}$	0.312	0.954	0.671	0.813	2.83
SPLADE-ℓ_1	0.322	0.954	0.667	0.792	0.88
SPLADE-$\ell_{\texttt{FLOPS}}$	0.322	0.955	0.665	0.813	0.73

of the FLOPS regularization over ℓ_1 in order to decrease the computing cost. Note that in contrast to SparTerm, SPLADE is trained end-to end in a single step. It is also remarkably simple, compared to dense state-of-the-art baselines such as ANCE [25], and avoids resorting to approximate neighbors search, whose impact on IR metrics has not been fully evaluated yet.

Effectiveness-efficiency trade-off. Figure 1 illustrates the trade-off between effectiveness (MRR@10) and efficiency (FLOPS), when we vary λ_q and λ_d (varying both implies that plots are not smooth). We observe that ST exp-$\ell_{\texttt{FLOPS}}$ falls far behind BOW models and

Figure 1: Performance vs FLOPS for SPLADE models trained with different regularization strength λ on MS MARCO

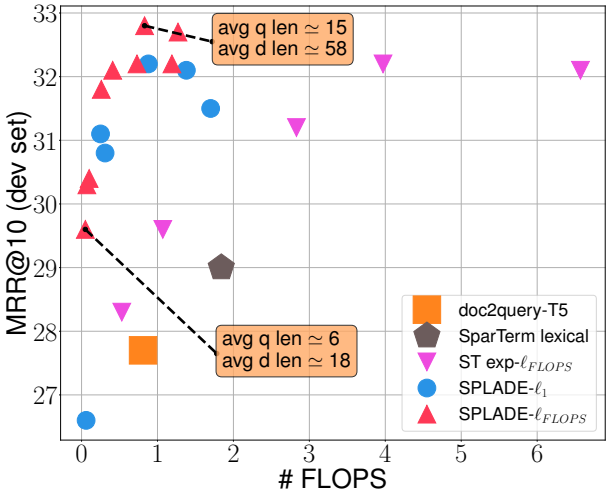

Table 2: Document and expansion terms: between parenthesis is the weight associated with the term – omitted for the second occurrence of the term in the document, and strike-through for zeros

original document (doc ID: 7131647)
if (1.2) bow (2.56) legs (1.18) ~~is~~ caused (1.29) by (0.47) ~~the~~ bone (1.2) alignment (1.88) issue (0.87) ~~than you may be~~ able (0.29) ~~to~~ correct (1.37) through (0.43) *bow legs* correction (1.05) ~~exercises.~~ ~~read more here..~~ *if bow legs is caused by the bone alignment issue than you may be able to correct through bow legs correction exercises.*
expansion terms
(leg, 1.62) (arrow, 0.7) (exercise, 0.64) (bones, 0.63) (problem, 0.41) (treatment, 0.35) (happen, 0.29) (create, 0.22) (can, 0.14) (worse, 0.14) (effect, 0.08) (teeth, 0.06) (remove, 0.03)

SPLADE in terms of efficiency. In the meantime, SPLADE reaches efficiency levels equivalent to sparse BOW models, while outperforming doc2query-T5. Interestingly, strongly regularized models still show competitive performance (e.g. FLOPS=0.05, MRR@10=0.296). Finally, the regularization effect brought by $\ell_{\texttt{FLOPS}}$ compared to ℓ_1 is clear: for the same level of efficiency, performance of the latter is always lower.

The role of expansion. Experiments show that the expansion brings improvements w.r.t. to the purely lexical approach by increasing recall. Additionally, representations obtained from expansion-regularized models are sparser: the models learn how to balance expansion and compression, by both turning-off irrelevant dimensions and activating useful ones. On a set of 10k documents, the SPLADE-$\ell_{\texttt{FLOPS}}$ from Table 1 drops in average 20 terms per document, while adding 32 expansion terms. For one of our most efficient model (FLOPS=0.05), 34 terms are dropped in average, for only 5 new expansion terms. In this case, representations are extremely sparse: documents and queries contain in average 18 and 6 non-zero values respectively, and we need less that 1.4 GB to store the index on disk. Table 2 shows an example where the model performs term re-weighting by emphasizing on *important* terms and discarding most of the terms without information content. Expansion allows to enrich documents, either by implicitly adding stemming effects (legs → leg) or by adding relevant topic words (e.g. treatment).

5 CONCLUSION

Recently, dense retrieval based on BERT has demonstrated its superiority for first-stage retrieval, questioning the competitiveness of traditional sparse models. In this work, we have proposed SPLADE, a sparse model revisiting query/document expansion. Our approach relies on in-batch negatives, logarithmic activation and FLOPS regularization to learn effective and efficient sparse representations. SPLADE is an appealing candidate for initial retrieval: it rivals the latest state-of-the-art dense retrieval models, its training procedure is straightforward, its sparsity/FLOPS can be controlled explicitly through the regularization, and it can operate on inverted indexes. In reason of its simplicity, SPLADE is a solid basis for further improvements in this line of research.

REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. arXiv:2010.00768 [cs.IR]

[2] Leonid Boytsov. 2018. *Efficient and Accurate Non-Metric k-NN Search with Applications to Text Matching*. Ph.D. Dissertation. Carnegie Mellon University.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[4] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. arXiv:1910.10687 [cs.IR]

[5] Zhuyun Dai and Jamie Callan. 2020. *Context-Aware Document Term Weighting for Ad-Hoc Search*. Association for Computing Machinery, New York, NY, USA, 1897–1907. https://doi.org/10.1145/3366423.3380258

[6] Zhuyun Dai and Jamie Callan. 2020. *Context-Aware Term Weighting For First Stage Passage Retrieval*. Association for Computing Machinery, New York, NY, USA, 1533–1536. https://doi.org/10.1145/3397271.3401204

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[8] Yingqi Qu Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2010.08191 [cs.CL]

[9] Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A Formal Study of Information Retrieval Heuristics. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Sheffield, United Kingdom) *(SIGIR '04)*. Association for Computing Machinery, New York, NY, USA, 49–56. https://doi.org/10.1145/1008992.1009004

[10] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. arXiv:2002.08909 [cs.CL]

[11] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv:2010.02666 [cs.IR]

[12] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906 [cs.CL]

[13] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/3397271.3401075

[14] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 1–6.

[15] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. arXiv:2010.11386 [cs.IR]

[16] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Jul 2020). https://doi.org/10.1145/3397271.3401262

[17] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:1901.04085 [cs.IR]

[18] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery.

[19] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. arXiv:1904.08375 [cs.IR]

[20] Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv:2004.05665 [cs.LG]

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library.. In *NeurIPS*.

[22] 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*. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[23] Zhengkai Tu, Wei Yang, Zihang Fu, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2020. Approximate Nearest Neighbor Search and Lightweight Dense Vector Reranking in Multi-Stage Retrieval Architectures. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 97–100.

[24] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[25] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. arXiv:2007.00808 [cs.IR]

[26] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwikj. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=zeFrfgyZln

[27] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype". *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Jul 2019). https://doi.org/10.1145/3331184.3331340

[28] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 497–506. https://doi.org/10.1145/3269206.3271800

[29] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2020. SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval. arXiv:2009.13013 [cs.CL]