A Reproducibility Study of PLAID

Sean MacAvaney
University of Glasgow
Glasgow, United Kingdom
sean.macavaney@glasgow.ac.uk

Nicola Tonellotto
University of Pisa
Pisa, Italy
nicola.tonellotto@unipi.it

ABSTRACT

The PLAID (Performance-optimized Late Interaction Driver) algorithm for ColBERTv2 uses clustered term representations to retrieve and progressively prune documents for final (exact) document scoring. In this paper, we reproduce and fill in missing gaps from the original work. By studying the parameters PLAID introduces, we find that its Pareto frontier is formed of a careful balance among its three parameters; deviations beyond the suggested settings can substantially increase latency without necessarily improving its effectiveness. We then compare PLAID with an important baseline missing from the paper: re-ranking a lexical system. We find that applying ColBERTv2 as a re-ranker atop an initial pool of BM25 results provides better efficiency-effectiveness trade-offs in low-latency settings. However, re-ranking cannot reach peak effectiveness at higher latency settings due to limitations in recall of lexical matching and provides a poor approximation of an exhaustive ColBERTv2 search. We find that recently proposed modifications to re-ranking that pull in the neighbors of top-scoring documents overcome this limitation, providing a Pareto frontier across all operational points for ColBERTv2 when evaluated using a well-annotated dataset. Curious about why re-ranking methods are highly competitive with PLAID, we analyze the token representation clusters PLAID uses for retrieval and find that most clusters are predominantly aligned with a single token and vice versa. Given the competitive trade-offs that re-ranking baselines exhibit, this work highlights the importance of carefully selecting pertinent baselines when evaluating the efficiency of retrieval engines.

https://github.com/seanmacavaney/plaidrepro

CCS CONCEPTS

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

KEYWORDS

Late Interaction, Efficiency, Reproducibility

ACM Reference Format:
Sean MacAvaney and Nicola Tonellotto. 2024. A Reproducibility Study of PLAID. 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, 9 pages. https://doi.org/10.1145/3626772.3657856

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

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657856

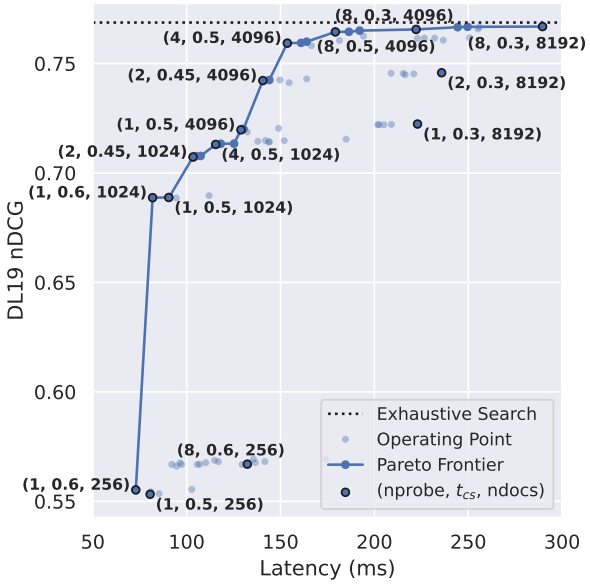

Figure 1: The Pareto frontier of PLAID for ColBERTv2 on TREC DL 2019 over the three parameters it introduces (`nprobe`, t_{cs}, and `ndocs`). Several operational points are labeled to highlight the interdependence of PLAID's parameters.

1 INTRODUCTION

Relevance ranking is a central task in information retrieval. Numerous classes of models exist for the task, including lexical [21], dense [10], learned sparse [18], and late interaction [11]. While efficient exact top-k retrieval algorithms exist for lexical and learned sparse retrieval systems (e.g., BlockMaxWAND [6]), dense and late interaction methods either need to perform expensive exhaustive scoring over the entire collection or resort to an approximation of top-k retrieval. A myriad of approximate k-nearest-neighbor approaches are available for (single-representation) dense models (e.g., HNSW [16]). However, these approaches generally do not apply directly to late interaction scoring mechanisms, so bespoke retrieval algorithms for late interaction models have been proposed.

PLAID (Performance-optimized Late Interaction Driver) [22] is one such retrieval algorithm. It is designed to efficiently retrieve and score documents for ColBERTv2 [23], a prominent late interaction model. PLAID first performs coarse-grained retrieval by matching the closest ColBERTv2 centroids (used for compressing term embeddings) to the query term embeddings. It then progressively filters the candidate documents by performing finer-grained estimations of a document's final relevance score. These filtering steps are controlled by three new parameters, which are discussed in more detail in Section 2.

The original PLAID paper answered several important research questions related to the overall effectiveness and efficiency compared to ColBERTv2's default retriever, the effect of the individual filtering stages, and its transferability. However, it also left several essential questions unanswered. This reproducibility paper aims to reproduce the core results of the paper and answer several additional questions. First, we explore the effect of PLAID's new parameters to better understand the practical decisions one must make when deploying a PLAID engine. Then we explore and report an important missing baseline (re-ranking a lexical retrieval system), which has been shown to be a highly-competitive approach for dense systems [12, 14]. Throughout our exploration, we also answer questions about how well PLAID applies to a dataset with many known relevant documents and how well it approximates an exhaustive ColBERTv2 search.

We find that PLAID's parameters need to be carefully set in conjunction with one another to avoid sub-optimal tradeoffs between effectiveness and efficiency. As shown in Figure 1, PLAID's Pareto frontier is a patchwork of parameter settings; changing one parameter without corresponding changes to the others can result in slower retrieval with no change to effectiveness. Further, we find that re-ranking lexical search results provides better efficiency-effectiveness trade-offs than PLAID in low-latency settings. For instance, competitive results can be achieved in a single-threaded setup in as low as 7ms/query with re-ranking, compared to 73ms/query for PLAID. Through an analysis of the token clusters, we confirm that a large proportion of tokens predominantly perform lexical matching, explaining why a lexical first stage is so competitive. We feel that our study provides important operational recommendations for those looking to use ColBERTv2 or similar models, both with and without the PLAID algorithm.

2 BACKGROUND AND PRELIMINARIES

The late interaction class of ranking models applies a lightweight query-token to document-token "interaction" operator atop a contextualized text encoder to estimate relevance between a query and a document. Perhaps most well-known is the ColBERT model [11], which applies a maximum-similarity operator over a pretrained transformer-based language model—though other late interaction operators (e.g., [15, 27]) and contextualization strategies (e.g., [5, 9]) have also been proposed. Due to the nature of their scoring mechanism, late interaction models cannot efficiently identify the exact top-k search results without an exhaustive scan over all documents,[1] nor can they directly use established approximate nearest neighbor algorithms.[2] Early work in late interaction approaches (e.g., [5, 9, 15]) overcame this limitation through a re-ranking strategy, wherein a candidate set of documents are identified using an efficient first-stage lexical retriever such as BM25 [21]. Khattab and Zaharia [11] identified that this re-ranking strategy may be sub-optimal, since the (lexically-matched) first stage results may not be aligned with those that the model will rank highly. Therefore, they proposed using approximate k-nearest neighbour search over the token representations to identify documents to score instead.

[1] In contrast, learned sparse models can be stored in inverted indices and retrieved using algorithms such as BlockMax-WAND [6].

[2] In contrast, single-representation dense retrieval models can be indexed and retrieved using algorithms such as HNSW [16].

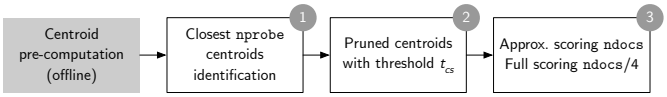

Figure 2: The logical phases composing the candidate documents identification procedure in PLAID.

To deal with the considerable space requirements to store the pre-computed representations of document tokens, ColBERTv2 [23] implements a clustering solution to identify document token centroids that can be used to decompose a document token representation as a sum of a centroid and a quantized residual vector, reducing the storage requirements by one order of magnitude w.r.t. the original ColBERT. These cluster centroids can serve as proxies of document tokens [22, 24, 25].

PLAID [22] further builds upon the centroids of ColBERTv2 to improve retrieval efficiency. PLAID selects and then progressively filters out candidate documents through three distinct phases, as illustrated in Figure 2. Firstly, given an encoded query token representation, its closest document token centroids are computed. The corresponding document identifiers are retrieved and merged together into a candidate set. The number of closest centroids to match per query token is a hyperparameter called `nprobe`. Naturally, the initial pool of documents increases in size as `nprobe` increases. Secondly, the set of candidate centroids is pruned by removing all centroids whose maximum similarity w.r.t. all query tokens is smaller than a threshold parameter t_{cs}. Next, the pruned set of centroids is further pruned by selecting the top `ndocs` documents based on relevance scores computed with the late interactions mechanism on the unpruned centroids. Then, the top `ndocs/4` approximately-scored documents are fully scored by decompressing the token representations and computing the exact ColBERTv2 relevance scores. Note that PLAID introduces a total of three hyperparameters, namely `nprobe`, t_{cs}, and `ndocs`. Although three suggested configurations of these settings were provided by the original PLAID paper, it does not explore the effects and inter-dependencies between them.

3 CORE RESULT REPRODUCTION

We begin by reproducing the core results of PLAID. Specifically, we test that retrieving using PLAID's recommended operational points provides the absolute effectiveness and relative efficiency presented in the original paper. Given the limitation that the original paper experimented with sparsely-labeled evaluation sets, we test one sparsely-labeled dataset from the original paper and one dataset with more complete relevance assessments. We also add a new measurement that wasn't explored in the original work—the Rank-Biased Overlap (RBO) [26] with an exhaustive ColBERTv2 search—to test how good of an approximation PLAID is with respect to a complete search.

Our experimental setup, detailed in the following section, includes both elements of both reproducibility and replicability per ACM's definitions,[3] since we are a different team using some of the same artifacts (code, model, datasets, etc.), while also introducing other changes to the experimental setup (added an evaluation dataset, new measures, etc.).

[3] https://www.acm.org/publications/policies/artifact-review-and-badging-current

Table 1: Suggested operational points from the original PLAID paper. We refer to them as (a), (b), and (c).

System	nprobe	t_{cs}	ndocs
PLAID (a)	1	0.50	256
PLAID (b)	2	0.45	1024
PLAID (c)	4	0.40	4096

3.1 Experimental Setup

Model and Code. We reproduce PLAID starting form the released ColBERTv2 checkpoint[4] and the PLAID authors' released codebase.[5] We release our modified version of the code and scripts to run our new experiments.

Parameters. We use PLAID's recommended settings for the nprobe, t_{cs}, and ndocs parameters, as shown in Table 1. We refer to these operational settings as (a), (b), and (c) for simplicity, where each setting progressively filters *fewer* documents. PLAID performs a final top k selection at the end of the process (i.e., after fully scoring and sorting the filtered documents). We recognize that this step is unnecessary and only limits the *apparent* result set size. Therefore, in line with typical IR experimental procedures, we wet $k = 1000$ across all settings. We also use the suggested settings of nbits=2 and nclusters=2^{18}.

Baselines. We compare directly against the results reported by the original PLAID paper for our experimental settings (Table 3 in their paper). We further conducted an exhaustive search over ColBERTv2[6] to better contextualize the results and support the measurement of rank-biased overlap (described below).

Datasets. We evaluate on the MS MARCO v1 passage development dataset [3, 19], which consists of 6,980 queries with sparse relevance assessments (1.1 per query). To make up for the limitations of these assessments, we also evaluate using the more comprehensive TREC DL 2019 dataset [4], which consists of 43 queries with 215 assessments per query. In line with the official task guidelines and the original PLAID paper, we do not augment the MS MARCO passage collection with titles [13].

Measures. For MS MARCO Dev, we evaluate using the official evaluation measure of mean Reciprocal Rank at depth 10 (RR@10), using MS MARCO's provided evaluation script. To understand the overall system's ability to retire the relevant passage, we measure the recall at depth 1000 (R@1k), which is also frequently used for the evaluation of Dev. To test how well PLAID approximates an exhaustive search, we measure Rank Biased Overlap (RBO) [26], with a persistence of 0.99. We measure efficiency via the mean response time using a single CPU thread over the Dev set in milliseconds per query (ms/q). In line with the original paper, we only measure the time for retrieval, ignoring the time it takes to encode the query (which is identical across all approaches). For TREC DL 2019, we evaluate the official measure of nDCG@10, alongside nDCG@1k to test the quality of deeper rankings and R@1k to test the ability of the algorithm to identify all known relevant passages to a given topic. Following standard conventions on TREC DL 2019, we use a minimum relevance score of 2 when computing recall. We use pytrec_eval [8] to compute these measurements.

[4]https://downloads.cs.stanford.edu/nlp/data/colbert/colbertv2/colbertv2.0.tar.gz
[5]https://github.com/stanford-futuredata/ColBERT
[6]I.e., fully scoring all documents. We modified the codebase to support this option.

Table 2: Core reproduction results. NR: Value was Not Reported in the original paper. N/A: The latency for an exhaustive search over ColBERTv2 is excessive and not applicable to this study.

	MS MARCO Dev				TREC DL 2019		
	RR@10	R@1k	RBO	ms/q	nDCG@10	nDCG@1k	R@1k
PLAID Reproduction							
(a)	0.394	0.833	0.612	80.5	0.739	0.553	0.555
(b)	0.397	0.933	0.890	103.4	0.745	0.707	0.786
(c)	0.397	0.975	0.983	163.9	0.745	0.760	0.871
Original PLAID Results							
(a)	0.394	NR	NR	185.5	NR	NR	NR
(b)	0.398	NR	NR	222.3	NR	NR	NR
(c)	0.398	0.975	NR	352.3	NR	NR	NR
Exhaustive ColBERTv2 Search							
-	0.397	0.984	1.000	N/A	0.745	0.769	0.894

Hardware. The original PLAID paper evaluated multiple hardware configurations, including single-CPU, multi-CPU, and GPU settings. Given the algorithm's focus on efficiency, we exclusively use a single-threaded setting, recognizing that most parts of the algorithm can be trivially parallelized on either CPU or GPU. Also as was done in the original work, we load all embeddings into memory, eliminating the overheads of reading from disk. We conducted our experiments using a machine equipped with a 3.4GHz AMD Ryzen 9 5950X processor. (The original paper used a 2.6 GHz Intel Xeon Gold 6132 processor.)

3.2 Results

Table 2 presents the results of our core reproduction study. We start by considering the effectiveness reported on MS MARCO Dev. We see virtually no difference across all three operational points in terms of the precision-oriented RR@10 measure.[7] In terms of efficiency, our absolute latency measurements are lower, though this is not surprising given that we are using a faster CPU. The approximate relative differences between each of the operational points are similar, however, e.g., operational point (b) provides a 37% speedup over (c) in both the original paper and our reproduction. Regarding R@1k and RBO, we see similar trends to that of RR@10: as the operational settings collectively consider more documents for final scoring, the measures improve. These results demonstrate that PLAID is working as expected: when more documents are considered, PLAID identifies a larger number of relevant documents (R@1k increases) and also produces a better approximation of an exhaustive ColBERTv2 search (RBO increases).

When considering the results on TREC DL 2019, we observe similar trends to the Dev results. The precision-focused nDCG@10 measure improves slightly from (a) to (b), while nDCG@1k and R@1k exhibit larger improvements across the settings due to the improved recall of the system. These results help further demonstrate PLAID's robustness in different evaluation settings.

[7]We note that tools such as repro_eval [1] are available to provide more fine-grained comparisons of individual rankings. In our study, we are primarily interested in the overall effectiveness, rather than the precise ordering of the results used to achieve these scores.

In summary, we are able to reproduce PLAID's core results (in terms of precision and efficiency) successfully on a single CPU setting. We further validate that the trends hold when measuring PLAID with recall-oriented measures and when evaluating PLAID on a dataset with more complete relevance assessments. However, there are still several limitations with the original evaluation. Although we know three settings in which PLAID's parameters can work together to deliver efficient retrieval, we do not understand the effect of each individually. Further, although PLAID retrieval is quite fast in an absolute sense (down to around 80ms/query on a single CPU core), we do not know how well this compares to highly-competitive re-ranking systems. These limitations are addressed in the following sections.

4 PARAMETER STUDY

Recall that PLAID introduces three new parameters: `nprobe` (the number of clusters retrieved for each token), t_{cs} (the centroid pruning threshold), and `ndocs` (the maximum number of documents returned after centroid interaction pruning). Although the original paper suggested three settings for these parameters (see Table 1), it did not explain how these parameters were selected or how each parameter ultimately affects retrieval effectiveness or efficiency. In this section, we fill this gap.

4.1 Experimental Setup

We extend the experimental setup from our core reproduction study presented in Section 3.1. We then performed a grid search over the following parameter settings: `nprobe` $\in \{1, 2, 4, 8\}$, $t_{cs} \in \{0.3, 0.4, 0.45, 0.5, 0.6\}$, and `ndocs` $\in \{256, 1024, 4096, 8192\}$.

This set of parameters was initially seeded by performing a grid search over the suggested parameter settings. Given that `nprobe` already includes the minimum value of 1, we extended it to 8 to check if introducing even more candidate documents from the first stage helps. For t_{cs}, we extended the parameter search in both directions: down to 0.3 (filtering out fewer documents based on the centroid scores) and up to 0.6 (filtering out more documents). Finally, we extended `ndocs` up to 8192, based on our observations that low values of this parameter (e.g., 256) substantially harm effectiveness.

We also asked the PLAID authors about anything else to tweak with PLAID to maximize effectiveness or efficiency. They told us that these three parameters have the most substantial effect. Meanwhile, the indexing-time parameters of `nbits` and `nclusters` can also affect the final performance. However, see these two indexing parameters as settings of the ColBERTv2 *model* rather than the PLAID *retriever*, so in the interest of keeping the number of combinations manageable, we focus on the retriever's parameters.

4.2 Results

Figure 3 presents the results of our parameter study. The figure breaks down the effect of each parameter when balancing retrieval latency (ms/q) and either MS MARCO Dev RR@10, Dev RBO, or DL19 nDCG@1k. Each evaluation covers a different possible goal of PLAID: finding a single relevant passage, mimicking an exhaustive search, and ranking all known relevant documents. To help visually isolate the effect of each parameter, lines connect the points that keep the other two parameters constant.

From examining the figure, it is clear that `ndocs` consistently has the most substantial effect on both effectiveness and efficiency. Selecting too few documents to score (`ndocs=256`) consistently reduces effectiveness while only saving minimal latency (around 10ms/q compared to `ndocs=1024`). Meanwhile, increasing `ndocs` further to 4096 does not benefit the quality of the top 10 results (RR@10). However, the change plays a consistent and important role in improving the quality of the results further down in the ranking (RBO and nDCG@1k). Finally, increasing `ndocs=8192` provides no additional benefits regarding search result quality or the faithfulness of the approximation to an exhaustive search, while increasing latency substantially. Based on these observations, we recommend setting `ndocs` $\in [1024, 4098]$, since the benefits of the values outside this range are minimal.

The next most influential parameter is `nprobe`. As expected, increasing the number of clusters matched for each token consistently increases the latency since more candidate documents are produced and processed throughout the pipeline. Setting the value too low (`nprobe=1` and sometimes `nprobe=2`) can often substantially reduce effectiveness, however, since documents filtered out at this stage will have no chance to be retrieved. This is especially apparent in Dev RR@10. Meanwhile, setting this value too high can reduce efficiency without yielding any gains in effectiveness.

Finally, t_{cs} has the smallest effect on retrieval effectiveness, with changes to this parameter typically only adjusting the retrieval latency. This can be see by the roughly horizontal lines in Figure 3. However, as this threshold gets too high, it can have variable effects on both effectiveness and efficiency. For instance, with Dev RR@10, setting $t_{cs} = 0.6$ sometimes reduces effectiveness and increases latency. Therefore, we recommend using $t_{cs} \in [0.4, 0.5]$ — and preferably towards the higher end of the range to limit the effect on latency.

We now consider the effect of all three parameters together. Achieving the Pareto frontier for PLAID involves tuning all three parameters in concert. For instance, the lowest retrieval latency requires a very low value of `ndocs`. However, lowering `ndocs` to 256 from 1024 without corresponding changes to the other parameters could simply yield worse effectiveness without making much of a dent in latency. Meanwhile, boosting `ndocs` without also adjusting `nprobe` will increase latency without improving effectiveness. Figure 1 (on Page 1) perhaps shows the effect of this patchwork of parameters most clearly, with the Pareto frontier formed of various combinations of `nprobe` $\in \{1, 2, 4, 8\}$, $t_{cs} \in \{0.3, 0.45, 0.5, 0.6\}$, and `ndocs` $\in \{256, 1024, 4096, 8192\}$.

In summary, each of PLAID's parameters plays a role in the final efficiency-effectiveness trade-offs of the algorithm. While `ndocs` plays the most important role, properly setting `nprobe` (and to a lesser extent, t_{cs}) is also necessary to achieve a good balance. In some ways, the importance of `ndocs` is unsurprising since the more documents you score precisely, the higher effectiveness one can expect (up to a point). But this begs some important questions. What is the impact of the source of the pool of documents for exact scoring? Is PLAID's progressive filtering process worth the computational cost compared to simpler and faster candidate generation processes? We answer these questions by exploring re-ranking baselines in the following section.

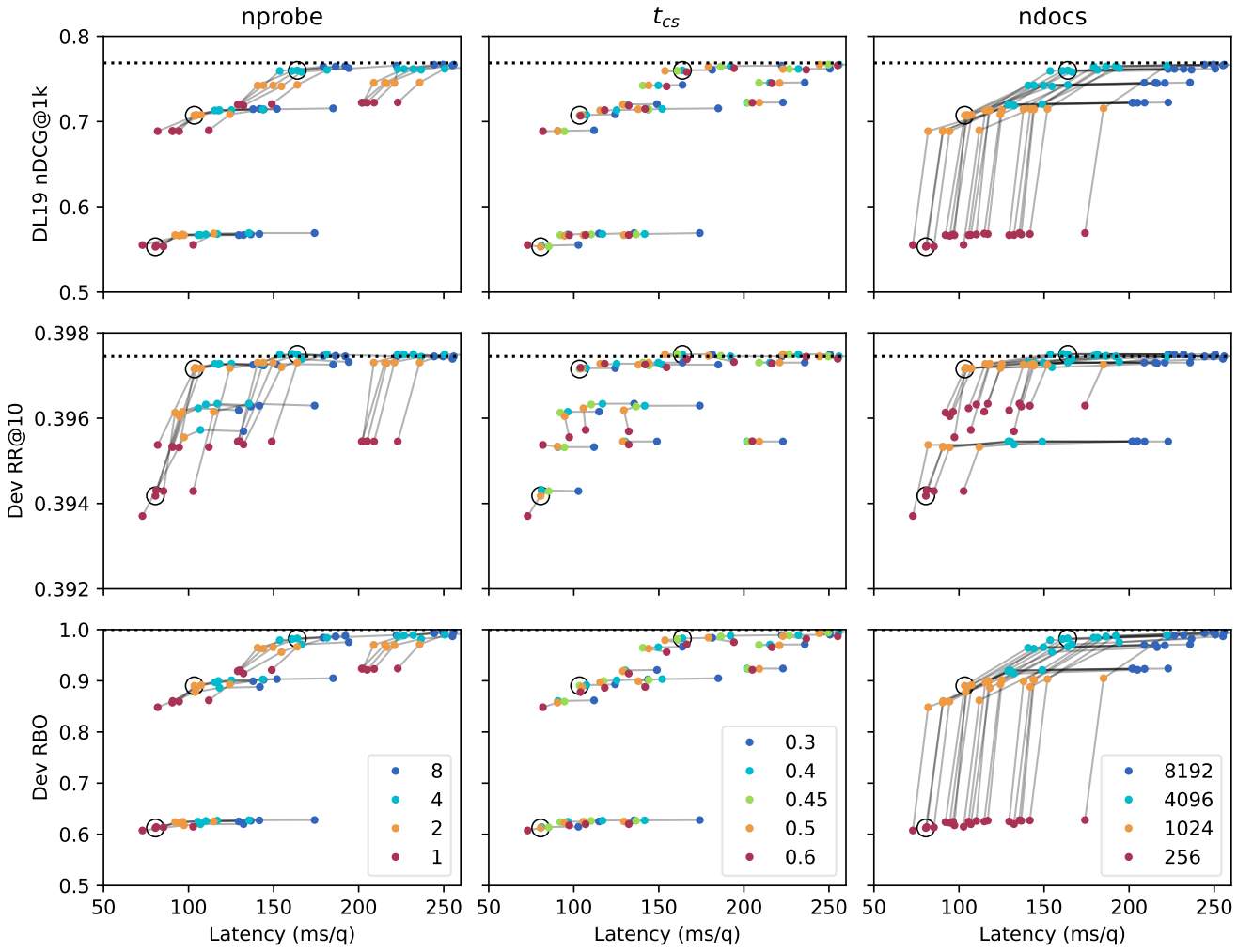

Figure 3: Results of our study of PLAID's parameters `nprobe`, t_{cs}, and `ndocs`. Each row plots the same data points, with the colors representing each parameter value and the lines between them showing the effect with the other two parameters held constant. The dotted line shows the results of an exhaustive search, and the circled points highlight the three recommended settings from the original paper.

5 BASELINE STUDY

The original paper compared PLAID's efficiency to three baselines: (1) Vanilla ColBERT(v2), which uses IVF indexes for each token for retrieval, in line with the method used by original ColBERT(v1) [11]; (2) SPLADEv2 [7], which is a learned sparse retriever [18]; and (3) BM25 [21], which is a traditional lexical retrieval model. Among these baselines, only Vanilla ColBERT(v2) represents an alternative retrieval engine; SPLADEv2 use other scoring mechanisms and act as points of reference. Curiously, the evaluation omitted the common approach of just re-ranking the results from an efficient-but-imprecise model like BM25. In this section, we compare PLAID with this baseline. Further, we compare both approaches with Lexically Accelerated Dense Retrieval (LADR) [12], which modifies the re-ranking algorithm to also consider the nearest neighbors of the top-scoring results encountered when re-ranking.

5.1 Experimental Setup

We use PLAID's experimental results from Section 4 as a starting point for our baseline study. We further modify the PLAID source code to support two more approaches: re-ranking and LADR.

Re-Ranking. We use the efficient PISA engine [17] for BM25 retrieval, using default parameters and a BlockMaxWAND [6] index structure. We then re-rank those results using ColBERTv2's decompression and scoring function. Given that we found the number of candidate documents for scoring to be the most important parameter for PLAID, we vary the number of retrieved results from BM25 as each of the following values: $n \in \{200, 500, 1000, 2000, 5000, 10000\}$. Note that due to the dynamic index pruning applied, performing initial retrieval is considerably faster for low values of n than for higher ones—in addition to the cost of ColBERTv2 decompression and scoring.

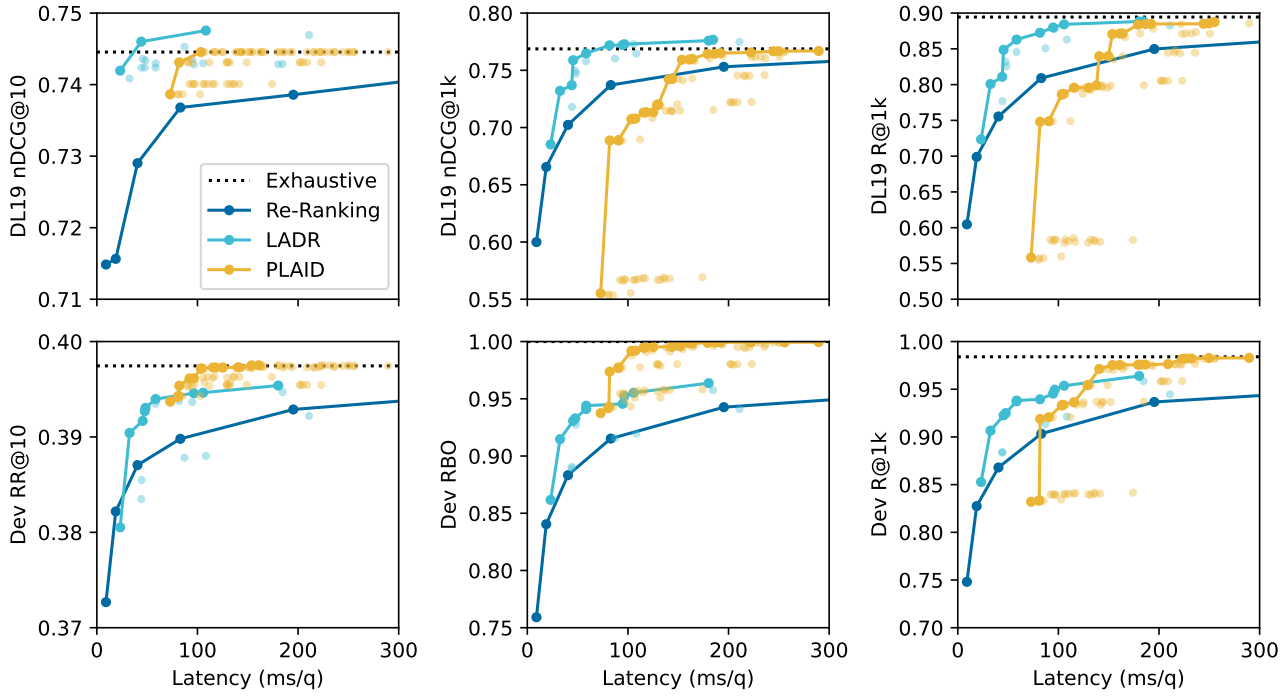

Figure 4: Results of our baseline study. The lines connecting points for each approach represent its Pareto frontier.

LADR. We further build upon the re-ranker pipeline using LADR. This approach incorporates a nearest neighbor lookup for top-scoring ColBERTv2 results to overcome possible lexical mismatch from the first stage retrieval. In line with the procedure for PLAID, we perform a grid search over the number of initial BM25 candidates $n \in \{100, 500, 1000\}$ and the number of nearest neighbors to lookup $k \in \{64, 128\}$. We use the precomputed nearest neighbor graph based on BM25 from the original LADR paper. By using the *adaptive* variant of LADR, we iteratively score the neighbors of the top $c \in \{10, 20, 50\}$ results until they converge.

Evaluation. We use the same datasets and evaluation measures as in Section 3.1. In line with this setting, we include the single-threaded first-stage retrieval latency from PISA for both additional baselines. In a multi-threaded or GPU environment, we note that this first-stage retrieval could be done in parallel with the ColBERTv2 query encoding process, further reducing the cost of these baselines. However, given the single-threaded nature of our evaluation, we treat this as additional latency.

5.2 Results

Figure 4 presents the results from our baseline study. We begin by focusing on the BM25 re-ranking pipeline. We observe that this pipeline can retrieve substantially faster than the fastest PLAID pipeline (as low as 9ms/q at $n = 200$, compared to 73ms/q for the fastest PLAID pipeline). Although this setting typically reduces the quality of results compared to the fastest PLAID pipelines (Dev RR@10, RBO, R@1k, and DL19 nDCG@10), it is still remarkably strong in terms of absolute effectiveness. For instance, its Dev RR@10 is 0.373 which is stronger than early BERT-based cross-encoders [20] and more recent learned sparse retrievers [7].

As the BM25 re-ranking pipeline considers more documents, the effectiveness gradually improves. In most cases, however, it continues to under-perform PLAID. For instance, when considering the top-10 documents via DL19 nDCG@10 and Dev RR@10, the Pareto frontier of the re-ranking pipeline always under-performs that of PLAID. Nevertheless, the low up-front dcost of performing lexical retrieval methods makes re-ranking an appealing choice when latency or computational cost are critical.

Re-ranking is inherently limited by the recall of the first stage, however, and when the first stage only enables lexical matches, this can substantially limit the potential downstream effectiveness. We observe that LADR, as an efficient pseudo-relevance feedback to a re-ranking pipeline, can largely overcome this limitation. On DL19, LADR's Pareto frontier completely eclipses PLAID's, both in terms of nDCG and recall. (LADR's non-optimal operational points are also consistently competitive.) Meanwhile, on Dev, LADR provides competitive—albeit not always optimal—effectiveness. Given that Dev has sparse assessments and DL19 has dense ones, we know that LADR is selecting suitable relevant documents as candidates, even though they are not necessarily the ones ColBERTv2 would have identified through an exhaustive search. The RBO results on Dev further reinforce this: while PLAID can achieve a nearly perfect RBO compared with an exhaustive search, LADR maxes out at around 0.96.

In summary, re-ranking and its variant LADR are highly competitive baselines compared to PLAID, especially at the low-latency settings that PLAID targets. Although they do not necessarily identify the same documents that an exhaustive ColBERTv2 search would provide, the baselines typically provide alternative documents of high relevance.

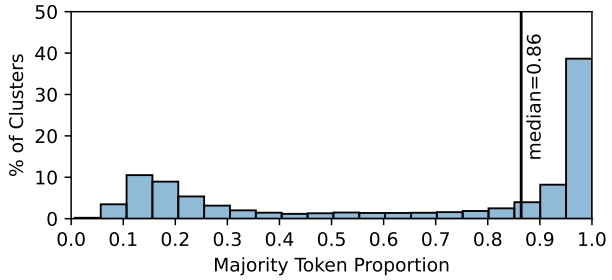

Figure 5: The distribution of Majority Token Proportions among clusters for ColBERTv2.

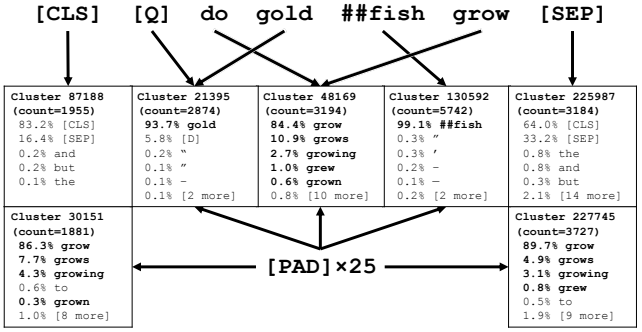

Figure 6: Example query, its corresponding clusters retrieved by PLAID, and the original tokens that contributed to each cluster. Bold tokens are (stemmed) lexical matches from the query.

We note that re-ranking comes with downsides, however. It requires building and maintaining a lexical index alongside ColBERT's index, which adds storage costs, indexing time, and overall complexity to the retrieval system. Nonetheless, these costs are comparatively low compared to those of deploying a ColBERTv2 system itself. For instance, a ColBERTv2 index of MS MARCO v1 consumes around 22GB of storage, while a lexical PISA index uses less than 1GB. Meanwhile, hybrid retrieval systems (i.e., those that combine signals from both a lexical and a neural model) will need to incur these costs anyway. LADR adds additional costs in building and maintaining a document proximity graph (around 2GB for a graph with 64 neighbors per document on MS MARCO).

5.3 Cluster Analysis

Curious as to why re-ranking a lexical system is competitive compared to PLAID, we conduct an analysis of the token representation clusters PLAID uses for retrieval vis-à-vis the lexical form of the token. We use the ColBERTv2 MS MARCO v1 passage index from the previous experiments, and modify the source to log the original token ID alongside the cluster ID and residuals of each token. We then conduct our analysis using this mapping between the token IDs and cluster IDs.

We start by investigating how homogeneous token clusters are. In other words, we ask the question: *Do most of a cluster's representations come from the same source token?* We first observe that most clusters map to multiple tokens (the median number of tokens a cluster maps to is 15, while only 2.2% of tokens only map to a single token). However, this does not tell the complete story since the distribution of tokens within each cluster is highly skewed. To overcome this, we measure the proportion of each cluster that belongs to the majority (or plurality) token. Figure 5 presents the distribution of the majority token proportions across all clusters. We observe that 39% of clusters have a majority proportion above 0.95 (i.e., over 95% of representations in these clusters come from the same token). Meanwhile, the median proportion among all clusters is 0.86. Only 2.7% of clusters have a majority proportion less than 10%. Collectively, these results suggest that although clusters are frequently formed of multiple tokens, they are usually heavily dominated by a single token. In other words, they largely perform lexical matching.

Within a cluster, what exactly are the other matching tokens? Figure 6 provides example clusters for the MS MARCO query "do goldfish grow". Some of the matching clusters (48169 and 225987) perform rather opaque semantic matching over `[CLS]` and `[SEP]` tokens. These clusters match either other such control tokens or (much less frequently) function words like `and`, `but`, and `the`. We suspect these function words are coopted to help emphasize the central points of a passage, given that they typically do not provide much in terms of semantics on their own. Next, three clusters (48169, 30151, and 227745) each have majority token proportions below or near the median. However, many of the minority tokens within a cluster are just other morphological forms of the same word: `grow`, `grows`, `growing`, etc. In other words, they share a common stem. When merging stems, these three clusters all have majority token proportions above 95%. The final two clusters (21395 and 130592) are dominated (>90% majority token proportion) by a single token. Like the control tokens, these pick up on punctuation tokens, which we suspect are coopted to help emphasize particularly salient tokens within a passage. This qualitative analysis suggests that although some clusters likely perform semantic matching, Figure 5 may actually be underestimate the overall prevalence of lexical matching among PLAID clusters.

The observation that most clusters map to a single source token only tells half the story, however. Perhaps PLAID is effectively performing a form of dynamic pruning [2], wherein query terms only match to a certain subset of lexical matches (i.e., the most semantically related ones) rather than all of them. After all, Figure 6 showed three separate clusters with the same majority token (`grow`). Therefore, we ask the inverse of our first question: *Do most of a token's representations map to the same cluster?* Akin to the cluster analysis, we measure the majority cluster proportion for each token, and plot the distribution in Figure 7. Here, 33% of tokens have a majority cluster proportion greater than 0.95. Unlike our observations in Figure 5, the tail is flatter and more uniform, giving a median majority cluster proportion of 0.62. These results suggest that although a sizable number of tokens map to a single prominent cluster, many tokens are spread among many different clusters. However, as can be seen in Figure 6, just because a token appears in many different clusters doesn't mean that it will necessarily be pruned off completely: two clusters that feature `grow` (30151 and 227745) are captured directly by the `[PAD]` "expansion" tokens of the query.

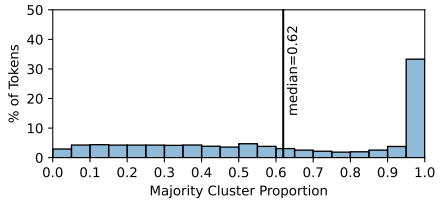

Figure 7: The distribution of Majority Cluster Proportions among tokens for ColBERTv2.

This analysis demonstrates that PLAID performs a considerable amount of lexical matching (though not exclusively so) when identifying documents to score. It also provides some insights into why re-ranking is competitive against PLAID.

6 CONCLUSION

In this paper, we conducted a reproducibility study of PLAID, an efficient retrieval engine for ColBERTv2. We were able to reproduce the study's main results, and showed that they successfully generalize to a dataset with more complete relevance assessments. We also showed that PLAID provides an excellent approximation of an exhaustive ColBERTv2 search. Using an in-depth investigation of PLAID's parameters, we found that they are highly interdependent, and the suggested settings are not necessarily optimal. Specifically, it is almost always worth increasing `ndocs` beyond the recommended 256, given the low contribution to latency and high boost in effectiveness that the change provides. Meanwhile, the missing baseline of simply re-ranking a lexical system using ColBERTv2 (and its recent variant, LADR) provides better trade-offs in terms of efficiency and effectiveness at low-latency operational points. However, these baselines do not provide as strong of a true approximation of an exhaustive ColBERTv2 search. Finally, an analysis showed that PLAID relies heavily on lexical matches for the initial retrieval of documents.

Our study provides important operational recommendations for those looking to deploy a ColBERTv2 system, both with and without the PLAID engine. It also further highlights the importance of comparing against versatile re-ranking systems when evaluating the efficiency of retrieval algorithms. Given the indirect way that PLAID performs first-stage lexical matching, future work could investigate methods for hybrid PLAID-lexical retrieval. By relying on PLAID for semantic matches and a traditional inverted index for lexical matches, we may be able to achieve the "best of both worlds": the high-quality ColBERTv2 approximation of PLAID and the high efficiency of re-ranking.

ACKNOWLEDGMENTS

This work is supported, in part, by the Spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, the Spoke "Human-centered AI" of the M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research", funded by European Union – NextGenerationEU, the FoReLab project (Departments of Excellence), and the NEREO PRIN project funded by the Italian Ministry of Education and Research Grant no. 2022AEFHAZ.

REFERENCES

[1] Timo Breuer, Nicola Ferro, Norbert Fuhr, Maria Maistro, Tetsuya Sakai, Philipp Schaer, and Ian Soboroff. 2020. How to Measure the Reproducibility of System-oriented IR Experiments. 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, 349–358. https://doi.org/10.1145/3397271.3401036

[2] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Y. Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*. ACM, 426–434. https://doi.org/10.1145/956863.956944

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. 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, 1566–1576. https://doi.org/10.1145/3404835.3462804

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR* abs/2003.07820 (2020). arXiv:2003.07820 https://arxiv.org/abs/2003.07820

[5] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 126–134. https://doi.org/10.1145/3159652.3159659

[6] Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, Wei-Ying Ma, Jian-Yun Nie, Ricardo Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft (Eds.). ACM, 993–1002. https://doi.org/10.1145/2009916.2010048

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *CoRR* abs/2109.10086 (2021). arXiv:2109.10086 https://arxiv.org/abs/2109.10086

[8] Christophe Van Gysel and Maarten de Rijke. 2018. Pytrec_eval: An Extremely Fast Python Interface to trec_eval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 873–876. https://doi.org/10.1145/3209978.3210065

[9] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020) (Frontiers in Artificial Intelligence and Applications, Vol. 325)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.). IOS Press, 513–520. https://doi.org/10.3233/FAIA200133

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 6769–6781. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.550

[11] 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, 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, 39–48. https://doi.org/10.1145/3397271.3401075

[12] Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2023. Lexically-Accelerated Dense Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 152–162. https://doi.org/10.1145/3539618.3591715

[13] Carlos Lassance and Stéphane Clinchant. 2023. The Tale of Two MSMARCO - and Their Unfair Comparisons. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 2431–2435. https://doi.org/10.1145/3539618.3592071

[14] Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2021. Fast Forward Indexes for Efficient Document Ranking. *CoRR* abs/2110.06051 (2021). arXiv:2110.06051 https://arxiv.org/abs/2110.06051

[15] 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

[16] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[17] Antonio Mallia, Michal Siedlaczek, Joel M. 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 (CEUR Workshop Proceedings, Vol. 2409)*, Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu (Eds.). CEUR-WS.org, 50–56. https://ceur-ws.org/Vol-2409/docker08.pdf

[18] 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

[19] 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

[20] Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019). arXiv:1901.04085 http://arxiv.org/abs/1901.04085

[21] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994 (NIST Special Publication, Vol. 500-225)*, Donna K. Harman (Ed.). National Institute of Standards and Technology (NIST), 109–126. http://trec.nist.gov/pubs/trec3/papers/city.ps.gz

[22] 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, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 1747–1756. https://doi.org/10.1145/3511808.3557325

[23] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (Eds.). Association for Computational Linguistics, 3715–3734. https://doi.org/10.18653/V1/2022.NAACL-MAIN.272

[24] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, Faegheh Hasibi, Yi Fang, and Akiko Aizawa (Eds.). ACM, 297–306. https://doi.org/10.1145/3471158.3472250

[25] Xiao Wang, Craig MacDonald, Nicola Tonellotto, and Iadh Ounis. 2023. ColBERT-PRF: Semantic Pseudo-Relevance Feedback for Dense Passage and Document Retrieval. *ACM Trans. Web* 17, 1 (2023), 3:1–3:39. https://doi.org/10.1145/3572405

[26] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.* 28, 4 (2010), 20:1–20:38. https://doi.org/10.1145/1852102.1852106

[27] Giulio Zhou and Jacob Devlin. 2021. Multi-Vector Attention Models for Deep Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 5452–5456. https://doi.org/10.18653/V1/2021.EMNLP-MAIN.443