Lexically-Accelerated Dense Retrieval

Hrishikesh Kulkarni
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

Sean MacAvaney
University of Glasgow
Glasgow, UK
first.last@glasgow.ac.uk

Nazli Goharian
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

Ophir Frieder
Georgetown University
Washington, DC, USA
first@ir.cs.georgetown.edu

ABSTRACT

Retrieval approaches that score documents based on learned dense vectors (i.e., dense retrieval) rather than lexical signals (i.e., conventional retrieval) are increasingly popular. Their ability to identify related documents that do not necessarily contain the same terms as those appearing in the user's query (thereby improving recall) is one of their key advantages. However, to actually achieve these gains, dense retrieval approaches typically require an exhaustive search over the document collection, making them considerably more expensive at query-time than conventional lexical approaches. Several techniques aim to reduce this computational overhead by *approximating* the results of a full dense retriever. Although these approaches reasonably approximate the top results, they suffer in terms of recall – one of the key advantages of dense retrieval. We introduce 'LADR' (Lexically-Accelerated Dense Retrieval), a simple-yet-effective approach that improves the efficiency of existing dense retrieval models without compromising on retrieval effectiveness. LADR uses lexical retrieval techniques to seed a dense retrieval exploration that uses a document proximity graph. Through extensive experiments, we find that LADR establishes a new dense retrieval effectiveness-efficiency Pareto frontier among approximate k nearest neighbor techniques. When tuned to take around 8ms per query in retrieval latency on our hardware, LADR consistently achieves both precision and recall that are on par with an exhaustive search on standard benchmarks. Importantly, LADR accomplishes this using only a single CPU – no hardware accelerators such as GPUs – which reduces the deployment cost of dense retrieval systems.

CCS CONCEPTS

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

KEYWORDS

dense retrieval, approximate k nearest neighbor, adaptive re-ranking

ACM Reference Format:
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 '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591715

1 INTRODUCTION

To identify relevant documents, search engines typically make use of lexical and/or dense search operations. Lexical approaches use the terms that appear in a user's query to identify and score documents that contain the same terms (e.g., using an approach like BM25 over an inverted index). Lexical approaches like BM25 [35] fail to understand semantic relationships in different scenarios [17]. A major drawback of this approach is that it is limited to documents that use the exact terms that the user searches for, which can potentially affect the recall. Recall is particularly important when results are further re-ranked in a second stage since documents missed in the first stage cannot be identified in subsequent stages using the typical *cascade* approach. To overcome this problem, *dense retrieval* approaches — which encode a user's query into a dense "semantic" vector and retrieve documents that were also encoded into semantic vectors — are becoming increasingly popular. By relying on dense vector similarity measures, retrieved documents no longer need any term overlap with a user's query.

Such approaches come at a comparatively high query-time cost, however. To produce state-of-the-art results, an exhaustive search over all document vectors is usually required. This is significantly more expensive than lexical search since it scales linearly with the number of documents in the corpus. (Meanwhile, lexical search

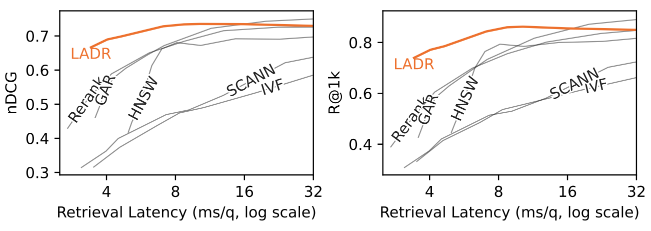

Figure 1: Comparison of our approach (LADR) and baselines when retrieving using TAS-B [13] on the TREC Deep Learning 2019 dataset, in terms of nDCG, Recall@1000, and latency (single CPU). LADR establishes a new Pareto frontier with higher nDCG and Recall at latencies below 16ms/query.

benefits from pruning the search space to only documents that use terms from the user's query, as well as other optimizations [6].) The efficiency problem of dense retrieval has spawned numerous efforts to *approximate* top results. Researchers proposed various approximation methods like HNSW [27], IVF [39, 47], and ScaNN [11]. However, these methods have their own limitations. Most notably, they suffer in terms of recall, which is meant to be one of the key benefits of dense retrieval.

Remarkably, simply re-ranking an initial pool of lexical results (such as BM25) remains highly competitive [20, 42]. We posit that this is due to two reasons. First, lexical signals are inherently valuable in many textual search tasks; intuitively, documents that satisfy a user's query will often contain the terms in the query itself. Second, pools of candidates with matching terms can be obtained very efficiently; decades of optimizations in this area have given us highly-optimized algorithms for the task, like BlockMax WAND [6]. Nevertheless, using a lexical model as a pool of documents for re-ranking has inherent drawbacks, including exclusive reliance on lexical signals for retrieval — another key problem that dense retrieval is meant to solve.

In this paper, we propose a simple-yet-effective technique called Lexically-Accelerated Dense Retrieval (LADR). It builds on the strengths of existing approximate k-nearest neighbor approximation techniques by leveraging a pool of candidate documents retrieved using a lexical model to seed the dense retrieval search. By using the lexical document pool, LADR leverages highly-efficient lexical search to obtain reasonable seed documents. LADR then explores the neighboring documents in the dense retrieval model's semantic space, allowing for documents to be retrieved that do not contain lexical matches with the query. We explore two strategies for exploration: a *proactive* strategy that greedily scores the neighbors of all seed documents, and an *adaptive* strategy that iteratively scores neighbors of documents in the top results until they converge.

Through extensive experiments across several dense retrieval models on standard text retrieval test collections, we find that LADR establishes a Pareto frontier in terms of retrieval effectiveness and efficiency. For example, Figure 1 compares the performance of LADR on the TAS-B [13] model with alternative approximation techniques.[1] At all latencies between 4 and 16ms/query, we see that LADR exceeds the effectiveness of existing techniques in terms of nDCG and Recall@1000. Further analysis shows that the parameters that LADR introduces (e.g., the number of nearest neighbors in the proximity graph) allow the user to easily trade off efficiency, effectiveness, and storage requirements, and that the approach is largely robust to alternative sources of proximity signals, such as approximate proximity graphs or graphs based on lexical proximity.

In summary, our contributions are as follows:

- We introduce LADR, a technique for reducing the computational overhead of dense retrieval while maintaining high effectiveness.
- We explore a proactive and an adaptive strategy to ensure effective exploration and optimal use of time budget for better retrieval.
- We conduct extensive experiments on standard benchmarks and a variety of single-representation dense retrieval models and find that our approach establishes a new Pareto frontier for low-latency approximate dense retrieval.
- Further analysis demonstrates that LADR is robust to its introduced parameters and alternative sources of document proximity.

[1]Each technique is parameterized by the degree of exploration. LADR and re-rank: number of candidates; HNSW: expansion factor; IVF/ScaNN: number of probes.

2 RELATED WORK

A significant number of AI problems can be modelled as search or information retrieval problems. Increasing quantum of information and need of very high efficiency always make this problem exceedingly challenging. Multiple approaches have been tried out to improve information retrieval effectiveness [32]. Generally, frequency of word occurrence in a document gives important measurement of word significance [24]. Typically, heuristic-based search approaches get the exploration directions from the knowledge about search space and query decoding. Tree-based [49] and Product Quantization-based [16, 48] indexing methods have been proposed. Popularly, three types of information retrieval models have inspired traditional approaches viz Boolean models, vector space models and probabilistic models [23]. Thus, different methods and models such as indexing, ranked retrieval, similarity measures and query expansion proposed by different researchers created a platform to solve problem of getting pertinent information challenged by information expansion.

2.1 Lexical Information Retrieval

Generally traditional approaches for retrieval exploit frequency and occurrence of query terms in the target document text [30]. In this approach the thrust is on exact term matching [10] creating basis for traditional IR systems. Varied weighting and normalization formulations over the frequency and occurrence of query terms in target documents [41] led to a variety of TF-IDF models exemplified by BM25 [35]. These bag-of-words based approaches like BM25 created initial avenues to negotiate with efficiency aspects of retrieval problem. Multiple variations of BM25 were introduced to tackle different application specific challenges [35]. Patterns identified in the query and documents should be independent of exact word matching and should highlight the relevance in case of different articulations of the same concept. To counter the issue of tight coupling of vocabulary to IR systems [31], different dictionary-based and n-gram-based techniques evolved to query likelihood [45] and query expansion for effective retrieval. These techniques have offered initial ground for IR and later machine learning techniques offered serious technical leap to enhance these techniques. First-stage retrieval models like SPLADE learn effective and efficient sparse representations where the sparsity can be controlled through regularization [8]. Guided processing heuristic is also used to boost efficiency in sparse models without significant loss in effectiveness [28]. Traditional NLP centric and token driven hand crafted IR evolved to more complex supervised learning models.

2.2 Neural Information Retrieval

Enhanced traditional information retrieval methods have too much vocabulary dependence. These enhancements fail to match the higher semantic levels due to underlined constrained semantic framework [34]. The focus on inspecting query and ignoring all whereabouts coming from the documents, limits the effectiveness of this approach. Neural models produce better results with the use of dense text representations. Typically, two primary types of neural models exist [33]. One is interaction based where interactions between words in queries are modelled. On the other hand, representation-based models learn a single vector representation of the query. The neural IR models can be used in conjunction with other models [46]. Bag-of-words models and other lexical methods can also be enhanced with machine learning models [2]. This may include use of machine learning based query expansion for sparse features coming through bag-of-words or learning based term weight tuning where BERT like pre-trained transformer models could be used [5]. Models like TAS-B [13] and TCT-ColBERT-HNP [22] produce state of the art results in dense retrieval. Considering the efficiency-effectiveness tradeoffs between sparse and dense models, hybrid models were proposed and choice of model was made based on time and resource budget [1]. Sparse auto encoders have also been used to increase efficiency in building parallel inverted index in Siamese-BERT based models [18]. As complimentary information is provided by sparse and dense methods, 'hybrid' methods have also been explored. Researchers combined sparse and dense retrieval ranking on the basis of relevance score interpolation leading to promising results [9, 22, 36].

2.3 Re-ranking Pipeline to Improve Efficiency

Traditional token-based and vocabulary-centric approaches like BM25 are efficient but not effective. On the contrary, dense retrieval approaches mainly based on contextualized transformer language models like BERT are highly effective but not efficient. Many researchers worked on approaches to strike the balance between efficiency and effectiveness. The most popular technique is to perform a two stage retrieval [7]. Efficient lexical methods are used at the first stage to retrieve an initial pool of documents based on the re-ranking budget. Effective dense retrieval methods are used at the second stage to re-rank the obtained results at the first stage [20, 25, 42]. Re-ranking pipelines perform very well but are limited by recall. Our LADR approach builds off this idea by seeding dense exploration with the results from a first-stage lexical ranker. This is similar to an idea proposed by [26] for the purposes of finding additional documents to score when re-ranking; ours differs from this work because we are exploring similar documents as a method of pruning the search space of dense retrieval while keeping comparable performance to an exhaustive search.

2.4 Seed-based Exploration in Proximity Graphs

Hierarchical Navigable Small World graphs (HNSW) [27] are used to perform efficient dense retrieval. Graph-based approaches use the clustering hypothesis which states that relevance among documents could be mapped to the same query [14]. Proximity graph methods exemplified by HNSW are current state of the art methods in ANN search and large scale retrieval using dense vector representations can be modelled as nearest neighbor search. To speed up the proceeding, approximate nearest neighbor search using HNSW is used. This results in substantial decrease in latency when compared to exhaustive dense retrieval. Another competitive index is Inverted Files index (IVF) [39, 47] which is based on Voronoi diagrams (Dirichlet tessellation). IVF also performs clustering to reduce the search scope. IVF based dense retrieval also reduces latency to a great extent delivering decent search-quality.

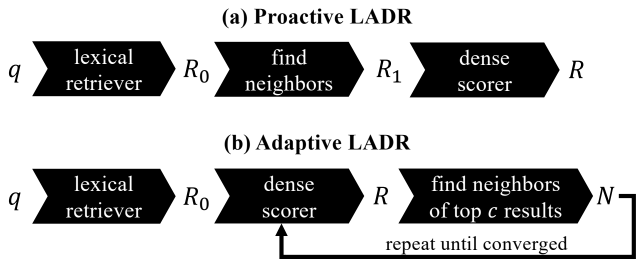

Figure 2: Overview of (a) Proactive and (b) Adaptive LADR.

ScaNN [11] is another quantization based technique which works on anisotropic quantization loss function for inner product search.

Most of the recent approaches couple random seed point-based exploration with dense retrieval. The initiation points are influenced by the proximity graph building algorithms and do not take differentiating query characteristics into account [39, 47]. Random exploration of search space until global minima is reached is often very expensive and the above mentioned approaches usually terminate around local minimas. Identification of right seed points to have informed exploration definitely could add value in terms of efficiency as well as accuracy. Multiple applications have witnessed enhancements in various aspects by using seeding techniques [19, 27, 37]. These seed points could be better derived from input document or query. We empower dense retrieval models with bag-of-words based seeding approach for informed exploration.

3 LADR

Lexically Accelerated Dense Retrieval (LADR) is a simple technique that combines the strengths of two leading dense retrieval techniques: *re-ranking*, which can efficiently identify potentially-relevant documents using lexical signals; and *HNSW*, which limits the search space based on a pre-computed document proximity graph. LADR leverages an efficient lexical model to identify good "seed" documents. It then makes use of a document proximity graph to expand the search space to similar documents. Here, the document proximity graph is computed offline. For each document, neighbors, i.e., 128 closest documents, are determined using the dense vector document similarity method TCT-ColBERT-HNP [26]. We identify two variants of LADR: a *proactive* variant (Section 3.2), which indiscriminately explores the neighbors of the seed documents, and an *adaptive* variant (Section 3.3), which selectively and iteratively explores the neighbors of the top c documents until those results converge. Figure 2 provides an overview of the two approaches. Each variant has its own advantages, which we detail in the following sections.

3.1 R_0 Lexical Seed Documents

Both the proactive and adaptive variants make use of an initial seed document set R_0 obtained from the top n results of a lexical retrieval model, e.g., BM25 [35]. Such models are largely seen as reasonable baselines for text retrieval tasks [21], and are generally robust across tasks and domains [40]. Beyond providing a competitive starting point in terms of recall, there are several key efficiency advantages in using a lexical retrieval method for generating seeds. First, decades of optimizations have resulted in highly-optimized query processing algorithms, such as BlockMax-WAND [6]. These algorithms generally result in sub-linear query processing time with respect to both the number of documents indexed[2] and the number of top results returned.[3] They also are inexpensive to construct, compress, and store without specialized hardware acceleration, especially when compared to the overheads of dense retrieval models.[4] Finally, it is commonplace in industry to use a lexical search index for first stage retrieval already (e.g., tools like ElasticSearch are prevalent), so the choice fits with existing operational structures.

3.2 Proactive LADR

Now that seed documents have been selected, we explore strategies for identifying additional documents to score. We first explore a *Proactive* LADR technique (see Algorithm 1). This variant takes a naïve approach for choosing documents to score; it simply expands the initial seed documents into R_1, by taking the union of the seed document set (R_0) and the k nearest neighbors of those documents. The final results are then selected by scoring R_1 using the dense scoring function (i.e., computing the similarity between the encoded query vector and the pre-computed document vectors associated with documents in R_1).

The degree of approximation of Proactive LADR is controlled by n (seed set size) and k (number of neighbors). As n increases, more potentially-relevant documents (or neighbors of potentially relevant documents) can be introduced by the lexical retriever. Meanwhile, as k increases, every seed document casts a wider net, allowing for more distant (but still similar) documents to be scored. We explore the interplay of these parameters in Section 5.3.

Computational and Storage Overheads. The computational cost of Proactive LADR is trivially bounded by the cost of lexical retrieval (usually sub-linear) plus $O(kn)$ vector similarity calculations. In practice, there is likely to be a substantial overlap between the neighbors of the seed documents due to the clustering hypothesis [14], further reducing the cost. Meanwhile, the storage overheads are the n nearest neighbors of each document, $O(Dn)$, where D represents the number of documents in the corpus.

3.3 Adaptive LADR

Proactive LADR's naïve approach of selecting the neighbors of all seed documents potentially wastes compute by scoring the neighbors of documents that turn out to be not so relevant. On the other hand, Adaptive LADR (see Algorithm 2) attempts to correct for this by adaptively choosing which neighbors to score based on the c top-scoring documents seen up to that point. Adaptive LADR starts by scoring the initial seed set of documents R_0. It then finds and scores the k nearest neighbors of the top c results. The algorithm then repeats the process until no new documents are introduced into the top c results. By being more selective about picking neighbors, Adaptive LADR can potentially spend the limited computational budgets (which are dominated by similarity scoring) on the most

Algorithm 1 Proactive LADR

```
Input: $q$ query, $n$ seed set size, $k$ number of neighbors
Output: $R$ dense retrieval results
  $R_0 \leftarrow \textsc{LexicalRetrieve}(q, n)$            ▷ seed with lexical results
  $R_1 \leftarrow R_0 \cup \textsc{Neighbors}(R_0, k)$             ▷ add neighbors
  $R \leftarrow \textsc{Score}(q, R_1)$          ▷ score seeds and their neighbors
```

Algorithm 2 Adaptive LADR

```
Input: $q$ query, $n$ seed set size, $c$ exploration depth, $k$ number of neighbors
Output: $R$ Dense Retrieval Results
  $R_0 \leftarrow \textsc{LexicalRetrieve}(q, n)$           ▷ seed with lexical results
  $R \leftarrow \textsc{Score}(q, R_0)$                           ▷ score seeds
  $N \leftarrow \textsc{Neighbors}(\text{top } c \text{ from } R, k)$      ▷ neighbors of top results
  $N \leftarrow N \setminus R$                   ▷ skip docs we've already scored
  while $|N| \neq 0$ do           ▷ iterate until no new docs identified
      $R \leftarrow R \cup \textsc{Score}(N)$               ▷ score and add neighbors
      $N \leftarrow \textsc{Neighbors}(\text{top } c \text{ from } R, k) \setminus R$
  end while
```

promising documents. When compared with the proactive strategy, it also makes it possible to move multiple hops in the document proximity graph, when a neighbor scores within the top c results.

Computational and Storage Overheads. Unlike Proactive LADR, the worst-case computational cost of Adaptive LADR is the cost of lexical retrieval plus $O(D)$ vector similarity calculations, which assumes a very poor selection of seed documents, necessitating a full exploration of the document graph. This case is very unlikely, given the well-established reasonable effectiveness of lexical search models like BM25. Even so, practitioners may consider including a timeout or other suitable termination criteria in Adaptive LADR to conclude the search early; we leave the exploration of the effect of such strategies for future work. The storage overheads are the same as for Proactive LADR: $O(Dn)$ needed to store the document graph where n represents the number of neighbors stored for each document.

3.4 Comparison of Variants

Proactive LADR expands the initial seed documents by taking into consideration the k nearest neighbors of each of the initial seed document. While Adaptive LADR keeps expanding the initial seed documents by considering k nearest neighbors selectively (only of top c documents) and iteratively (until convergence). Though relatively simple, Proactive LADR has some distinct advantages over more complicated LADR approaches. First, there is a trivial $O(kn)$ upper bound to the number of vector comparisons made which is $O(D)$ in the case of Adaptive LADR. In practice, the number of comparisons is going to be significantly less due to overlap in the neighbors of seed documents based on the clustering hypothesis and reasonable effectiveness of BM25. But in worst case scenario Adaptive LADR will be equivalent to exhaustive retrieval. As Proactive LADR is a pipeline approach, operations can be trivially delegated (e.g., a separate service might be responsible for computing vector similarities, which only needs to be invoked a single time for Proactive LADR). On the other hand Adaptive LADR operations tend to be slightly demanding and costlier due to the

[2]For instance, PISA [29] retrieves the top 1000 BM25 results over 100k, 1M, and 10M passages in 0.4, 1.1, and 4.2 ms on average, respectively.
[3]For instance, PISA [29] retrieves the top 10, 100, and 1000 BM25 results for 10M passages in 1.1, 1.9, and 4.2 ms on average, respectively.
[4]For instance, it takes PISA [29] around 2 minutes to build an optimized BM25 index for MS MARCO (727MB in size), while it takes around 3 hours to compute TAS-B [13] vectors for MS MARCO using an A6000 GPU (26GB in size).

feedback loop introduced by the iterative exploration approach (e.g., a separate vector similarity service would need to be invoked multiple times, which could incur higher costs).

3.5 Comparison with Existing Methods

Re-ranking. Both LADR variants build upon the high effectiveness of the re-ranking technique [20, 42]. Using a lexical model for re-ranking alone is a major drawback, however: it inherently limits the retrievable documents to those with lexical matches. Though we argue that this is a reasonable starting point, it is not ideal to exclusively use such signals for dense retrieval because the highest-scoring results from a dense retrieval model need not have lexical overlap. LADR addresses this problem by expanding the scope of the search to the neighboring documents in the semantic space.

HNSW [27]. Both HNSW and LADR make use of document similarity scores when finding documents to score. While LADR makes use of an efficient lexical model to identify potentially relevant documents, HNSW uses a hierarchical structure to narrow into the neighborhood of the query vector. We argue that when strong and efficient lexical models are available to accelerate the selection process (like in many text retrieval settings), they should be used. Doing so allows LADR to allocate the compute to scoring more documents, rather than narrowing in on the neighborhood.

GAR [26]. LADR is perhaps most similar to Graph-Based Adaptive Re-ranking (GAR). Both techniques make use of an initial ranking and document similarities to select documents for scoring. GAR uses an alternating strategy to explore either the initial results, or exploit the results of the top-scoring document, with a major focus on prioritizing which to score next. This is important in the setting of expensive re-rankers, such as cross encoders, because the overall cost of scoring each document is very high (e.g., it's often only practical to score fewer than 1000 documents). Meanwhile, the cost of scoring a document in a bi-encoder setting is relatively cheap (it is reasonable to score a few thousand documents per query). Therefore, it is less important to be very selective about which documents to score, and instead broadly search in areas that have the *potential* to yield relevant documents. This motivates the strategy of iterative exploration of neighbors of LADR. Indeed, in Section 5.1 we show that even when the GAR technique is properly optimized for a dense retrieval setting, it underperforms LADR.

Hybrid Retrieval [9, 22, 36]. Ensembles of lexical and dense retrieval systems, often referred to as "hybrid" retrieval systems, also bear some semblance to LADR. Unlike hybrid retrieval systems, LADR only makes use of lexical retrieval systems insofar as to select a set of candidate documents, rather than using the lexical ranking score in the final order of the system itself. Consequently, LADR could be used in conjunction with a hybrid ensemble.

Guided Traversal [28]. This strategy computes document scores for an arbitrary scoring function (in their work, a learned sparse scorer) for each document encountered while processing a lexical query. This can be seen as a generalization of the re-ranking strategy that also considers *some*[5] documents with lexical matches but not a high lexical ranking score. In contrast, LADR (1) is not constrained to only lexical matches; (2) avoids scoring documents that are neither high-scoring lexically nor neighbors of strong candidates; and (3) can easily augment existing retrieval pipelines, since it does not involve modifying the query traversal strategy, it simply takes the final output.

[5]In addition to the top n lexical results, Guided Traversal scores documents encountered while traversing the posting lists for finding the top n lexical results.

4 EXPERIMENTAL SETUP

Via experimentation, we answer the following research questions:

RQ1 How does LADR compare to other approximation techniques in terms of effectiveness and efficiency?
RQ2 Is LADR applicable to a variety of single-representation dense retrieval models?
RQ3 What are the computational overheads of LADR?
RQ4 How do the parameters introduced by LADR (e.g., number of neighbors k) affect the effectiveness and efficiency of LADR?
RQ5 Is an exact nearest neighbor graph needed for LADR to be effective?
RQ6 What are the trade-offs between proactive and adaptive LADR in terms of precision, recall and latency?

We have released the code to reproduce the results of our experiments.[6]

4.1 Datasets and Measures

TREC 2019 Deep Learning (Passage Subtask). This is the official evaluation query set used in the TREC 2019 Deep Learning shared task [4]. It consists of 43 manually-judged queries using four relevance grades (215 relevance assessments per query, on average).

TREC 2020 Deep Learning (Passage Subtask). This is the official evaluation query set used in the TREC 2020 Deep Learning shared task [3]. It consists of 54 queries with manual judgments from NIST annotators (211 relevance assessments per query, on average).

Dev (small). This is the official small version of the dev set and has 6980 queries which is about 6.9% of the full MS-Marco passage dev set. Unlike TREC DL 19/20, this dataset only contains a few known relevant passages per query (1.1 on average).

We evaluate using nDCG and Recall (with a minimum relevance score of 2) at 1000 for TREC DL 2019 and 2020. Similarity, we evaluate using Mean Reciprocal Rank at 10 and Recall at 1000 for Dev (small). To measure how faithful the approximations are to the true rankings, we report Rank Biased Overlap (RBO [43]) with $p = 0.99$ of each method to the results from an exhaustive search. We also report the per-query mean retrieval latency in milliseconds. Since query encoding time (i.e., the time to compute the query vector) is constant over all approximate nearest neighbor methods, we only consider the retrieval time in our measurements (i.e., we exclude query encoding latency).[7] Given the larger query sample, we use the Dev (small) dataset for all latency measurements.

4.2 Models and Parameters

We apply LADR and the baselines over the following diverse sample of single-representation dense retrieval models: TAS-B[8] [13],

[6]http://github.com/Georgetown-IR-Lab/ladr
[7]Indeed, we also found that query encoding time exhibited higher variance across dense retrieval models than the the approximate nearest neighbor method.
[8]`sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco`

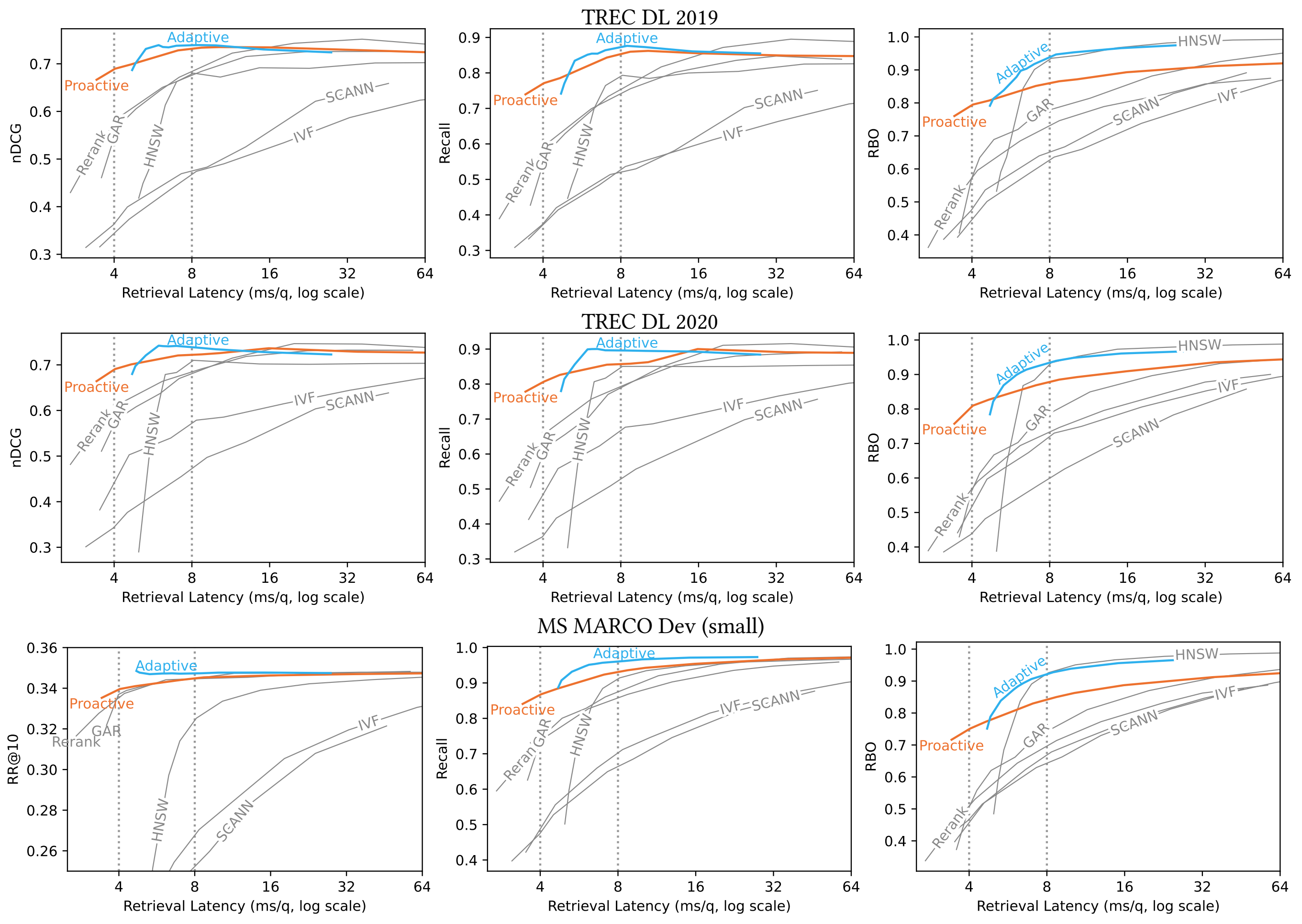

Figure 3: Performance of LADR over TAS-B and baselines across various operational points.

RetroMAE[9] [38], TCT-ColBERT-HNP[10] [22], and ANCE[11] [44]. We use BM25 over a BlockMax-WAND [6] index encoding with default parameters using PISA [29] as our initial seed documents (R_0).

For Proactive LADR, we primarily explore with a fixed $k = 128$ (neighboring documents), varying $n \in [10, 1000]$ (seed set size) to trade off efficiency and effectiveness. For Adaptive LADR, we primarily explore with a fixed $n = 200$ and $k = 128$, varying $c \in [1, 200]$ (exploration depth) to trade off efficiency and effectiveness. We further explore the effects of n, k, and c in Section 5.3. By default, we use exact neighbors for LADR by preforming an exhaustive k nearest neighbor search over all documents; we further explore the effect of using approximate or alternative graphs in Section 5.4.

4.3 Baselines and Implementation

We compare LADR with the following baselines:

HNSW [27] is a nearest neighbor search based indexing techniques which enables efficient dense retrieval. We use $k = 128$ neighbors to match the settings of LADR, and vary the expansion factor in the range $[1, 4096]$ to trade off effectiveness and efficiency.

IVF [39, 47] is an inverted file index which is based on reduction of search scope using clustering. We evaluated IVF models with 4096 and 65536 centroids. IVF model with 4096 centroids was relatively efficient and delivered better results when compared with the one having 65536 centroids. We use between $[1, 128]$ probes to trade off effectiveness and efficiency.

Re-ranking pipeline was also evaluated using BM25 for first-stage retrieval, re-scored using dense model. We trade off efficiency and effectiveness by varying $n \in [10, 20000]$ (documents re-ranked).

GAR [26] alternates between exploring the initial results and the neighbors of the top-scoring documents. We use the same $k = 128$ as with LADR, and vary the scoring budget in the range $[64, 4096]$.

ScaNN [11] is a quantization-based technique which works on anisotropic quantization loss function for inner product search. We set the number of leaves to 4096 and search between $[1, 128]$ of them to trade off effectiveness and efficiency.

For re-ranking, GAR, and LADR, we use our own implementations, leveraging the efficient dot product implementation provided

[9]`Shitao/RetroMAE_MSMARCO_distill`, we found our results to be slightly lower than the results reported in the paper because we use the official MS MARCO passage data, excluding the extra titles used in the RetroMAE experiments.
[10]`castorini/tct_colbert-v2-hnp-msmarco`
[11]`sentence-transformers/msmarco-roberta-base-ance-firstp`

Table 1: Retrieval effectiveness comparison of various dense scoring models and approximate retrieval methods at two operational points (approximately 4ms/query and 8ms/query) for TREC DL19, TREC DL20, and MS MARCO Dev (small). Results that cannot be matched to the operational point are indicated with "-". Significant improvements between approaches and LADR are indicated using superscript/subscript indicators (paired t-test, $p < 0.05$). The indicators are I, S, H, G and R that correspond to respective baselines. For context, exhaustive retrieval effectiveness is also provided for each model in gray, though note that these values do not correspond with the operational points.

	DL19 ~4ms		DL19 ~8ms		DL20 ~4ms		DL20 ~8ms		Dev (sm) ~4ms		Dev (sm) ~8ms	
Method	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k	nDCG	R@1k	RR@10	R@1k	RR@10	R@1k
TAS-B (Exh.)	0.715	0.842	0.715	0.842	0.713	0.875	0.713	0.875	0.347	0.978	0.347	0.978
IVF [I]	0.374	0.414	0.474	0.536	0.503	0.559	0.579	0.677	0.217	0.556	0.270	0.712
ScaNN [S]	0.475	0.519	0.537	0.598	0.476	0.527	0.553	0.641	0.254	0.669	0.292	0.774
HNSW [H]	-	-	0.614	0.707	-	-	0.699	0.836	-	-	0.310	0.872
GAR [G]	0.543	0.540	0.688	0.755	0.568	0.594	0.684	0.796	0.337	0.732	0.345	0.876
Re-Ranking [R]	0.589	0.605	0.684	0.755	0.615	0.667	0.691	0.805	0.337	0.748	0.345	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.690}$	$^{IS}_{GR}\mathbf{0.771}$	$^{ISH}_{GR}0.730$	$^{ISH}_{GR}0.850$	$^{IS}_{GR}\mathbf{0.691}$	$^{IS}_{GR}\mathbf{0.807}$	$^{IS}_{GR}0.722$	$^{IS}_{GR}0.857$	$^{IS}\mathbf{0.340}$	$^{IS}_{GR}\mathbf{0.868}$	$^{ISH}0.345$	$^{ISH}_{GR}0.932$
Adaptive LADR	-	-	$^{ISH}_{GR}\mathbf{0.738}$	$^{ISH}_{GR}\mathbf{0.872}$	-	-	$^{ISH}_{GR}\mathbf{0.739}$	$^{ISH}_{GR}\mathbf{0.900}$	-	-	$^{ISH}_{GR}\mathbf{0.347}$	$^{ISH}_{GR}\mathbf{0.960}$
RetroMAE (Exh.)	0.699	0.806	0.699	0.806	0.701	0.839	0.701	0.839	0.375	0.981	0.375	0.981
IVF [I]	0.226	0.225	0.346	0.358	0.272	0.263	0.372	0.375	0.157	0.381	0.221	0.541
ScaNN [S]	0.468	0.502	0.525	0.588	0.486	0.509	0.555	0.606	0.275	0.665	0.312	0.769
HNSW [H]	-	-	0.630	0.720	-	-	0.673	0.798	-	-	0.338	0.874
GAR [G]	0.559	0.553	0.696	0.763	0.578	0.604	0.692	0.789	**0.357**	0.750	0.368	0.890
Re-Ranking [R]	0.594	0.605	0.685	0.755	0.622	0.667	0.696	0.805	0.355	0.748	0.369	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.691}$	$^{IS}_{GR}\mathbf{0.765}$	$^{ISH}_{GR}0.733$	$^{ISH}_{GR}0.844$	$^{IS}_{GR}\mathbf{0.702}$	$^{IS}_{GR}\mathbf{0.811}$	$^{ISH}_{G}0.723$	$^{IS}_{G}0.846$	$^{IS}0.356$	$^{IS}_{GR}\mathbf{0.864}$	$^{ISH}0.368$	$^{ISH}_{GR}0.938$
Adaptive LADR	-	-	$^{ISH}_{R}\mathbf{0.740}$	$^{ISH}_{GR}\mathbf{0.866}$	-	-	$^{ISH}_{G}\mathbf{0.731}$	$^{ISH}_{GR}\mathbf{0.879}$	-	-	$^{ISH}_{GR}\mathbf{0.374}$	$^{ISH}_{GR}\mathbf{0.973}$
TCT-HNP (Exh.)	0.708	0.830	0.708	0.830	0.689	0.848	0.689	0.848	0.359	0.970	0.359	0.970
IVF [I]	0.340	0.366	0.437	0.469	0.369	0.383	0.470	0.522	0.219	0.527	0.276	0.687
ScaNN [S]	0.378	0.410	0.444	0.496	0.355	0.376	0.427	0.459	0.215	0.522	0.253	0.632
HNSW [H]	-	-	0.625	0.721	-	-	0.634	0.762	-	-	0.315	0.853
GAR [G]	0.546	0.547	0.687	0.755	0.569	0.598	0.678	0.797	0.342	0.733	0.354	0.878
Re-Ranking [R]	0.586	0.605	0.679	0.755	0.614	0.667	0.685	0.805	0.342	0.748	0.353	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.680}$	$^{IS}_{GR}\mathbf{0.747}$	$^{ISH}_{GR}0.719$	$^{ISH}_{GR}0.827$	$^{IS}_{GR}\mathbf{0.682}$	$^{IS}_{GR}\mathbf{0.803}$	$^{ISH}_{G}0.709$	$^{ISH}0.841$	$^{IS}_{G}\mathbf{0.346}$	$^{IS}_{GR}\mathbf{0.856}$	$^{ISH}0.354$	$^{ISH}_{GR}0.927$
Adaptive LADR	-	-	$^{ISH}\mathbf{0.729}$	$^{ISH}_{GR}\mathbf{0.848}$	-	-	$^{ISH}_{GR}\mathbf{0.721}$	$^{ISH}_{GR}\mathbf{0.878}$	-	-	$^{ISH}_{GR}\mathbf{0.359}$	$^{ISH}_{GR}\mathbf{0.962}$
ANCE (Exh.)	0.617	0.755	0.617	0.755	0.634	0.777	0.634	0.777	0.330	0.957	0.330	0.957
IVF [I]	0.358	0.395	0.441	0.500	0.407	0.437	0.498	0.549	0.212	0.530	0.268	0.703
ScaNN [S]	0.374	0.405	0.433	0.488	0.440	0.495	0.535	0.614	0.262	0.691	0.287	0.783
HNSW [H]	-	-	0.606	0.737	-	-	0.635	0.790	-	-	0.311	0.897
GAR [G]	0.527	0.540	0.648	0.750	0.568	0.622	0.655	0.794	**0.326**	0.751	0.329	0.888
Re-Ranking [R]	0.578	0.605	0.653	0.755	0.602	0.667	**0.674**	0.805	0.325	0.748	**0.333**	0.868
Proactive LADR	$^{IS}_{GR}\mathbf{0.645}$	$^{IS}_{GR}\mathbf{0.751}$	$^{IS}0.657$	$^{ISH}0.800$	$^{IS}_{GR}\mathbf{0.660}$	$^{IS}_{GR}\mathbf{0.807}$	$^{IS}0.666$	$^{IS}0.822$	$^{IS}0.321$	$^{IS}_{GR}\mathbf{0.872}$	$^{ISH}0.327$	$^{ISH}_{GR}0.932$
Adaptive LADR	-	-	$^{ISH}\mathbf{0.665}$	$^{ISH}\mathbf{0.820}$	-	-	$^{ISH}0.665$	$^{IS}\mathbf{0.830}$	-	-	$^{ISH}0.329$	$^{ISH}_{GR}\mathbf{0.959}$

by NumPy [12]. For HNSW and IVF we use the FAISS implementation [15]. For ScaNN, we use the authors' implementation.[12] All methods run a single query at a time on a single CPU with the document vectors pre-loaded into memory. Experiemnts were conducted using AMD Ryzen Threadripper 1920X (3.9GHz) processor.

5 RESULTS AND ANALYSIS

This section presents results and analysis of different experiments with reference to our hypothesis and research questions. Figure 3 presents the results of LADR and the baseline approaches across their effectiveness/efficiency trade-off parameters for TAS-B. We start by exploring the relative strengths of the baselines. In terms of the precision-oriented measures, nDCG and RR@10, re-ranking and GAR out-perform HNSW, IVF, and ScaNN at most operational points. In terms of Recall@1k, we observe a similar trend, but with HNSW out-performing re-ranking and GAR above around 6ms/query on Dev (small), though not on TREC DL. Across all three datasets, HNSW produces result sets that are more faithful to exhaustive searches, as evidenced by far higher RBO scores than other baselines. Finally, we note that through our testing, we were unable to tune HNSW at or below 4ms/query with $k = 128$.

Next, we see that both LADR variants perform favorably to the baselines across all three measures. At low-latency settings (e.g., at or below 8ms/query), Adaptive LADR establishes a new Pareto frontier in terms of nDCG, RR@10, Recall@1k and RBO. However,

[12] https://github.com/google-research/google-research/tree/master/scann

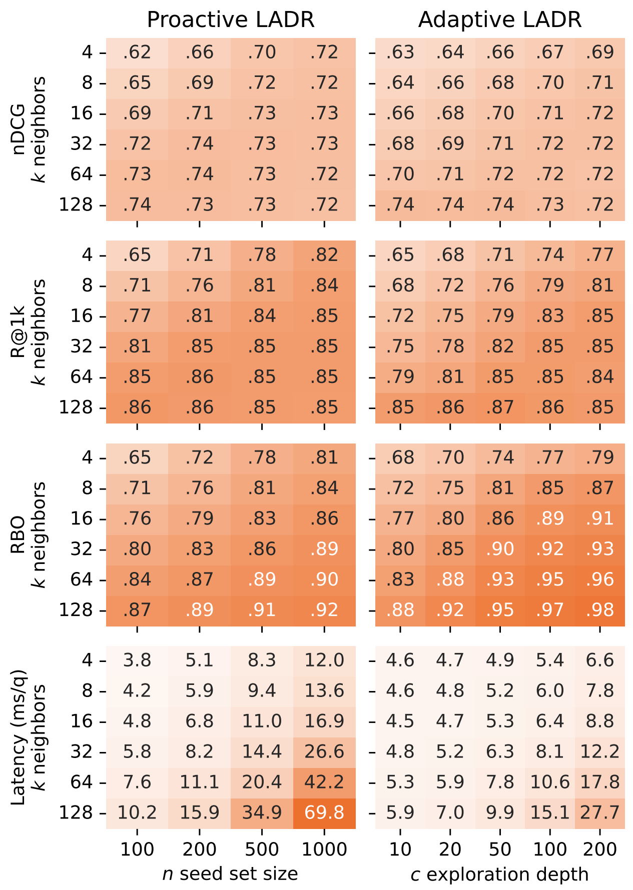

Figure 4: Comparison of LADR performance over TAS-B on TREC DL 2019 while varying k, n, and c.

akin to HNSW, we were unable to tune it such that it performed at or below 4ms/query with $k = 128$; in these cases, the Proactive LADR variant is preferable. At higher latency settings, Adaptive LADR performs on-par with the best baselines across all datasets and measures (re-ranking/GAR for nDCG, RR@10, and Recall@1k, and HNSW for RBO). LADR achieves the "best of both worlds": higher effectiveness than the baselines at low latency settings, and effectiveness on par with the top baseline at high latency settings.

5.1 RQ1: nDCG and Recall@1k

To establish whether LADR is truly more effective at low-latency settings, we conduct experiments at two operational points: 4ms/query and 8ms/query. For each variant of LADR and the baselines, we select parameters that yield latencies as close as possible to these values. We then apply the approach across four different dense retrieval models: TAS-B, RetroMAE, TCT-ColBERT-HNP and ANCE, and present the results in Table 1. For additional context, we also provide the performance of an exhaustive search for each of the models, though we note that these values do not correspond to the 4ms and 8ms operational points – instead, they take around 60ms/query when using a modern GPU.

When used with TAS-B, the baselines achieved at best an nDCG of 0.589 and 0.615 for DL 2019 and DL 2020 datasets with a 4ms time budget. These results were clearly outperformed by Proactive LADR with statistically significant improvement leading to a nDCG of 0.690 and 0.691, respectively. Similarly with a 8ms budget for TAS-B, both Proactive and Adaptive LADR outperform the baselines with Adaptive LADR giving a nDCG of 0.738 (DL19) and 0.739 (DL20) over 0.688 and 0.699, the top baseline results. The improvement in terms of Recall@1k is even more remarkable. Recall@1k of 0.771 and 0.807 was observed over 0.605 and 0.667 (baselines) for Proactive LADR on DL 2019 and DL 2020 for a 4ms budget. Similarly, Recall@1k of 0.872 and 0.900 was observed over 0.755 and 0.836 (baselines) for Adaptive LADR on DL 2019 and DL 2020 for a 8ms budget. On the Dev (small) dataset we observe that both Proactive and Adaptive LADR outperform the baselines in the Reciprocal Rank @10 and Recall@1k metrics in both 4ms and 8ms time budgets. We observe similar trends across other methods namely RetroMAE, TCT-ColBERT-HNP and ANCE. Another key observations is that both Proactive and Adaptive LADR with 8ms time budget yields higher nDCG and Recall@1k metrics than the exhaustive search across all dense retrieval methods on the DL 2019 and DL 2020 datasets. However, we note that this may be due, in part, to pooling bias, which could favor results included in the annotation pool (like BM25, which contributes to LADR).

These results answer RQ1: LADR consistently performs better than existing approximation techniques, both nDCG and Recall@1k, in low-latency settings. Further, it performs competitively with exhaustive retrieval, particularly at the 8ms/query operational point.

5.2 RQ2: Model Applicability

Table 1 presents the Proactive and Adaptive LADR results across a variety of dense retrieval models. It shows statistically significant improvements over the baselines for DL19 and DL20 across all four dense retrieval methods, with the only exception being ANCE on DL20. These improvements are observed in both nDCG and Recall@1k metrics at multiple operational points. Further, this trend also largely holds on the Dev (small) dataset, though occasionally re-ranking and GAR achieve higher RR@10 performance (RetroMAE and ANCE). Hence, addressing RQ2 we conclude that both Proactive and Adaptive LADR are applicable to a variety of single-representation dense retrieval models.

5.3 RQ3 and RQ4: Overhead and Parameters

Next, we perform an analysis of the parameters introduced by LADR: the number of neighbors per document in the proximity graph (k), the seed set size (n) and the exploration depth (c). Figure 4 reports the nDCG, Recall@1k, RBO, and mean latency per query, when using the TAS-B model over the DL19 dataset.

We observe several trends, which should help researchers and practitioners tune LADR for their particular needs. First, as the number of neighbors in the proximity graph (k) increases, the performance in nDCG, Recall@1k and RBO (Rank Biased Overlap) also increase, but so does latency. This is expected, given that increasing k results in LADR scoring more documents that are nearby potentially relevant ones [14]. However, increasing k comes at the cost of additional storage requirements; at $k = 128$, 4.3GB of additional storage is required for the MS MARCO dataset. Gains can also be made when increasing the seed set size (n) or the exploration depth (c), though it is costlier in terms of compute (e.g., 12.0ms/q vs

10.2ms/q) for less gains (0.82 vs 0.86 Recall@1k). In terms of nDCG and Recall@1k, the gains from these parameters are complementary up to a point, which allows the user to balance their effectiveness, query-time efficiency, and storage requirements. Meanwhile, if the goal is to best approximate an exhaustive search, Adaptive LADR is the way to go; 0.98 RBO can be achieved at only 27.7ms/q when both a large k and c are used, compared to only 0.92 RBO for Proactive LADR, costing 69.8ms/q. Thus, RQ3 and RQ4 have been addressed by studying changes in latency, effectiveness and efficiency with respect to parameters introduced by LADR, and adjusting these parameters allows the user to trade off effectiveness, query-time efficiency, and storage requirements.

5.4 RQ5: Document Proximity Sources

Up to this point, we have explored the setting where the exact top k nearest documents to each document in the target vector space are known. This is not always a realistic requirement, however; computing such scores is a quadratic operation over the number of documents in the collection. We therefore now explore two alternative sources of document proximity to test whether LADR can be used without requiring a precise similarity graph.

We explore two alternatives. First, we identify that HNSW [27] constructs its proximity graph in loglinear time by making use of approximate searches through its hierarchical structure for each new document. We therefore explore an *Approximate* setting, wherein the document similarity graph is constructed by the HNSW index construction algorithm (Algorithm 1 in [27]). Meanwhile, MacAvaney et al. [26] found that using document proximity via BM25 provides similar effectiveness as dense vector document similarity in their adaptive re-ranking algorithm. Given that a BM25 graph would be model-agnostic, we also consider this strategy here.

We construct the two alternative document similarity graphs (Approx. and BM25), each linking up to $k = 128$ nearby documents, and report the results in Table 2 for TAS-B. Table 2 shows results for Exact proximity graph with TAS-B, Approx method (HNSW) and BM25 for the DL 2019, DL 2020 and Dev (small) datasets. As evident in Table 2, differences in nDCG and Recall@1k of Approx proximity graph when compared with Exact proximity graph are not significant for datasets DL 2019 and DL 2020. Similar trends are observed when BM25 based proximity graph and Exact proximity graph are compared. Further statistical equivalence for results without significant differences are also depicted in Table 2. Both Proactive and Adaptive LADR show high performance irrespective of the the proximity graph construction method. The results suggest that both Proactive and Adaptive LADR are remarkably robust to these alternative proximity signals addressing RQ5.

5.5 RQ6: Proactive and Adaptive Trade-offs

Adaptive LADR performs better than Proactive LADR in the metrics of nDCG and Recall@1k in the same time budget. Adaptive LADR by being more selective about neighbor selection through an iterative approach ensures that the limited computational resources are being spent on promising documents. One limitation of Adaptive LADR is that operational points with very low time budget do not exist. Additionally, Proactive LADR has O(kn) worst case time complexity where k is the number of neighbors and n is the number of seed documents. While for Adaptive LADR worst case time

Table 2: Effectiveness of LADR over TAS-B when using alternative sources of document similarity, namely an approximation of the exact similarity graph from HNSW and a graph constructed using BM25. Operational points for Exact LADR are selected from the ~8ms setting from Table 1; alternative sources use the same settings. Significant differences as compared to Exact are indicated with $*$ (paired t-test, $p < 0.05$). For results without significant differences, we indicate statistical equivalence with = (paired TOST, $p < 0.05$, $|\Delta| < 0.02$).

	DL19		DL20		Dev (sm)	
Graph	nDCG	R@1k	nDCG	R@1k	RR@10	R@1k
			Proactive LADR			
Exact	0.730	**0.850**	**0.722**	**0.857**	**0.345**	**0.932**
Approx.	$^{=}$0.731	0.845	$^{=}$0.720	0.849	*0.343	*0.916
BM25	$^{=}$**0.732**	0.835	$^{=}$0.720	0.853	*0.339	*0.883
			Adaptive LADR			
Exact	0.738	**0.872**	0.739	**0.900**	**0.347**	0.960
Approx.	$^{=}$0.736	0.861	$^{=}$0.737	$^{=}$**0.900**	$^{=}$**0.347**	***0.966**
BM25	**0.743**	0.859	$^{=}$**0.742**	**0.900**	*0.345	*0.933

complexity is O(D) where D is the number of documents in the collection. Even though worst case is unlikely, latency for Adaptive LADR will be at par with Exhaustive retrieval. Considering the trade-offs between Proactive and Adaptive LADR for very low operational points Proactive LADR would be a better choice. While, Adaptive LADR with a timeout strategy would be a better choice for cases with relatively higher time budgets addressing RQ6.

6 CONCLUSIONS AND FUTURE DIRECTIONS

The ability to identify related documents that do not necessarily contain same terms but are closer in the semantic space is the key of dense retrieval approaches. This requires exhaustive search which is an expensive operation. Current state of the art approximate k nearest neighbor methods like HNSW, ScaNN, IVF etc try to provide some resolve to this challenging task but fail to match the precision and recall of exhaustive search. Our simple-yet-effective method 'LADR' (Lexically-Accelerated Dense Retrieval) uses lexical retrieval techniques to seed a dense retrieval exploration that uses a document proximity graph. Both Proactive and Adaptive LADR establish a new dense retrieval effectiveness-efficiency Pareto frontier among approximate k nearest neighbor techniques. While Proactive LADR outperforms baselines at lower time budgets, Adaptive LADR focuses on optimal use of compute on promising documents leading to a major leap in recall. LADR consistently achieves both nDCG and recall that are on par with an exhaustive search on standard benchmarks without needing GPUs.

One natural area for further exploration is in developing new LADR variants; Proactive and Adaptive are appealing in their simplicity and remarkable effectiveness, but there likely exist smarter exploration strategies. Given that LADR relies on intra-document similarity, another interesting direction would be to train dense retrieval models with this quality in mind – e.g., by introducing a new intra-document similarity objective in the training process, rather than query-document similarity alone.

REFERENCES

[1] Negar Arabzadeh, Xinyi Yan, and Charles L. A. Clarke. 2021. Predicting Efficiency/Effectiveness Trade-Offs for Dense vs. Sparse Retrieval Strategy Selection. In *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 2862–2866. https://doi.org/10.1145/3459637.3482159

[2] Arian Askari and Suzan Verberne. 2021. Combining Lexical and Neural Retrieval with Longformer-based Summarization for Effective Case Law Retrieval. In *DESIRES*.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. https://doi.org/10.48550/ARXIV.2102.07662

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. https://doi.org/10.48550/ARXIV.2003.07820

[5] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[6] Shuai Ding and Torsten Suel. 2011. Faster Top-k Document Retrieval Using Block-Max Indexes. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 993–1002. https://doi.org/10.1145/2009916.2010048

[7] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, and Jiafeng Guo. 2021. Pre-training Methods in Information Retrieval. https://doi.org/10.48550/ARXIV.2111.13853

[8] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. https://doi.org/10.1145/3404835.3463098

[9] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement Lexical Retrieval Model with Semantic Residual Embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part I*. Springer-Verlag, Berlin, Heidelberg, 146–160. https://doi.org/10.1007/978-3-030-72113-8_10

[10] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 55–64. https://doi.org/10.1145/2983323.2983769

[11] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *International Conference on Machine Learning*. https://arxiv.org/abs/1908.10396

[12] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[13] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 113–122. https://doi.org/10.1145/3404835.3462891

[14] N. Jardine and C. J. van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Inf. Storage Retr.* 7 (1971), 217–240.

[15] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[16] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[17] Kyungho Kim, Kyungjae Lee, Seung-won Hwang, Young-In Song, and Seungwook Lee. 2021. Query Generation for Multimodal Documents. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 659–668. https://doi.org/10.18653/v1/2021.eacl-main.54

[18] Carlos Lassance, Thibault Formal, and Stéphane Clinchant. 2021. Composite code sparse autoencoders for first stage retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2136–2140.

[19] Fabian Lechtenberg, Javier Farreres, Aldwin-Lois Galvan-Cara, Ana Somoza-Tornos, Antonio Espuña, and Moisès Graells. 2022. Information retrieval from scientific abstract and citation databases: A query-by-documents approach based on Monte-Carlo sampling. *Expert Systems with Applications* 199 (2022), 116967. https://doi.org/10.1016/j.eswa.2022.116967

[20] Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2022. Efficient Neural Ranking Using Forward Indexes. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 266–276. https://doi.org/10.1145/3485447.3511955

[21] Jimmy Lin. 2019. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (jan 2019), 40–51. https://doi.org/10.1145/3308774.3308781

[22] 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

[23] Bing Liu. 2007. Web usage mining. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* (2007), 449–483.

[24] H. P. Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* 2, 2 (1958), 159–165. https://doi.org/10.1147/rd.22.0159

[25] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1573–1576. https://doi.org/10.1145/3397271.3401262

[26] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking with a Corpus Graph. In *31st ACM International Conference on Information and Knowledge Management*. https://doi.org/10.1145/3511808.3557231

[27] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[28] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster Learned Sparse Retrieval with Guided Traversal. 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, 1901–1905. https://doi.org/10.1145/3477495.3531774

[29] Antonio Mallia, Michal Siedlaczek, Joel M. Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *OSIRRC@SIGIR*.

[30] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[31] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval* 13, 1 (December 2018), 1–126. https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/

[32] Bhaskar Miutra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Found. Trends Inf. Retr.* 13, 1 (dec 2018), 1–126. https://doi.org/10.1561/1500000061

[33] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. https://doi.org/10.48550/ARXIV.1606.04648

[34] Samarth Rawal and Chitta Baral. 2020. Multi-Perspective Semantic Information Retrieval. https://doi.org/10.48550/ARXIV.2009.01938

[35] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[36] Saar Kuzi, Mingyang Zhang 001, Cheng Li 012, Michael Bendersky, and Marc Najork. 2020. Leveraging Semantic and Lexical Matching to Improve the Recall of Document Retrieval Systems: A Hybrid Approach. *CoRR* (2020).

[37] Sheikh Muhammad Sarwar, John Foley, Liu Yang, and James Allan. 2019. Sentence Retrieval for Entity List Extraction with a Seed, Context, and Topic. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval* (Santa Clara, CA, USA) *(ICTIR '19)*. Association for Computing Machinery, New York, NY, USA, 209–212. https://doi.org/10.1145/3341981.3344250

[38] Yingxia Shao Zhao Cao Shitao Xiao, Zheng Liu. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *EMNLP*. https://arxiv.org/abs/2205.12035

[39] Sivic and Zisserman. 2003. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*. 1470–1477 vol.2. https://doi.org/10.1109/ICCV.2003.1238663

[40] 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

[41] Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *J. Artif. Int. Res.* 37, 1 (jan 2010), 141–188.

[42] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. An Inspection of the Reproducibility and Replicability of TCT-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, 2790–2800. https://doi.org/10.1145/3477495.3531721

[43] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.* 28 (2010), 20:1–20:38.

[44] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *ArXiv* abs/2007.00808 (2021).

[45] Xiaobing Xue, Van Dang, and W. Bruce Croft. 2009. Query Substitution based on N-gram Analysis. In *SIGIR*.

[46] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials*. Association for Computational Linguistics, Online, 1–4. https://doi.org/10.18653/v1/2021.naacl-tutorials.1

[47] Jiangbo Yuan, Guillaume Gravier, Sébastien Campion, Xiuwen Liu, and Hervé Jégou. 2012. Efficient Mining of Repetitions in Large-Scale TV Streams with Product Quantization Hashing. In *Computer Vision - ECCV 2012. Workshops and Demonstrations - Florence, Italy, October 7-13, 2012, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 7583)*, Andrea Fusiello, Vittorio Murino, and Rita Cucchiara (Eds.). Springer, 271–280. https://doi.org/10.1007/978-3-642-33863-2_27

[48] Han Zhang, Hongwei Shen, Yiming Qiu, Yunjiang Jiang, Songlin Wang, Sulong Xu, Yun Xiao, Bo Long, and Wen-Yun Yang. 2021. Joint Learning of Deep Retrieval Model and Product Quantization Based Embedding Index. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1718–1722. https://doi.org/10.1145/3404835.3462988

[49] Jingwei Zhuo, Ziru Xu, Wei Dai, Han Zhu, Han Li, Jian Xu, and Kun Gai. 2020. Learning Optimal Tree Models under Beam Search. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 11650–11659. https://proceedings.mlr.press/v119/zhuo20a.html