On the Interplay Between Graph Quality, Traversal Strategies, and Performance of ANN Retrieval Methods

Hrishikesh Kulkarni
Instacart
San Francisco, CA, 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

State-of-the-art approximate nearest neighbor (ANN) methods like HNSW and LADR use document-document proximity graphs (also known as corpus graphs) to identify relevant documents efficiently. Complete graph construction latency (though built offline) has a quadratic time complexity of the number of documents, which is a major hurdle when scaling these methods. Graph approximations are popular ways to reduce the computational cost of building such corpus graphs. However, approximations come with a cost, namely, a lower quality of corpus graphs. Hence, there is a practical need to understand the tradeoffs between a corpus graph's quality and its effectiveness when used with various ANN methods; in other words, how 'approximate' can a corpus graph be while maintaining strong retrieval effectiveness? We construct approximate (i.e. poorer quality) corpus graphs using various methods and present extensive experiments that analyze the robustness and performance of popular ANN methods on these graphs. Our analysis is performed on multiple datasets, with different parameters and various poor graph simulation strategies. We also analyze different graph traversal approaches for robust and efficient retrieval across graphs of poor quality. We conclude by addressing the utility of these approaches at the billion-scale, practical scenarios by optimizing graph construction and graph traversal stages. We show that robust ANN methods like Adaptive LADR show statistically equivalent performance on poor quality graphs while saving 33% graph construction time.

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

Approximate Nearest Neighbor (ANN) methods; Hierarchical Navigable Small World (HNSW) graphs; Lexically Accelerated Dense Retrieval (LADR); Heuristic Search

ACM Reference Format:
Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2025. On the Interplay Between Graph Quality, Traversal Strategies, and Performance of ANN Retrieval Methods. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP 2025), December 7–10, 2025, Xi'an, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3767695.3769495

SIGIR-AP 2025, Xi'an, China

ACM ISBN 979-8-4007-2218-9/2025/12
https://doi.org/10.1145/3767695.3769495

1 Introduction

Information Retrieval (IR) systems answer user queries by providing a ranked list of documents [36]. Both lexical and neural approaches are commonly used. Lexical approaches compare terms appearing in user queries and their synonyms with those appearing in documents to identify and score documents efficiently, while neural approaches work on effective semantic representation independent of the typical word-matching paradigm. Combinations of lexical and neural approaches balance efficiency and effectiveness [20], often relying on approximate nearest neighbor (ANN) methods.

A primary challenge in using ANN methods in billion-scale IR scenarios [14] is the expense of a high graph construction time [28]. The quadratic time complexity of the number of documents makes the graph construction process highly expensive even though it is performed offline at the indexing stage [13]. For example, on MS MARCO document corpus with 8.8 million documents, it takes about 2 hours to build an exact (i.e. full-quality) nearest neighbor graph on an NVIDIA RTX A6000 GPU. Given the quadratic time complexity, at billion-scale, a multiplicative factor of 10^4 leads to roughly 1000 days for graph construction. Thus, approximations are usually used, leading to inexact (i.e., poorer-quality) graphs.

ANN search methods come with their own challenges related to scalability and recall. The efficiency and effectiveness of graph-based ANN methods depend on the corpus graph quality and traversal strategies. Thus, we investigate the robustness of graph-based ANN search methods across parameters with respect to graph quality and graph traversal strategies and provide insights into efficient graph construction and traversal for practical search applications.

Our contributions are as follows:

- We evaluate the impact of graph quality on graph-based ANN search approaches and observe that ANN methods with informed seeding strategies are less susceptible to poor graph quality.
- We provide a robustness comparison of graph-based ANN approaches across poor graphs and establish the robustness of LADR over HNSW.
- We study the tradeoff between ANN effectiveness and graph construction latency and determine threshold graph construction quality parameter resulting in acceptable ANN effectiveness.
- We analyze the performance of LADR with different exploration strategies across different graph qualities and observe that Best First Search and A* are the strongest ANN heuristic search strategies we explored.

2 Related Work

We briefly review related prior efforts, partitioning our review into lexical, dense, and approximate nearest neighbor approaches.

2.1 Lexical Retrieval

Traditional IR approaches are more word-centric and dictionary-centric. In these approaches, query terms are used to associate with target documents according to their frequency in those documents [27]. Lexical methods are known for their efficiency, as they use an inverted index. The different weighting possibilities [33] resulted in more effective variants of TF-IDF such as BM25 [30]. The efforts continued to eliminate exact word matching and focused more on relevance in the patterns. It led to machine learning-based IR, resulting in better performance, and overcome some of the limitations posed by lexical approaches [31]. Subsequently, the dense and hybrid approaches tried to consider contextual insights in relevance by learning semantic representations. They resulted in unprecedented mileage on the effectiveness front.

2.2 Dense Retrieval

Semantic level constraints and the problem of vocabulary mismatch restrict the performance of traditional lexical approaches. Neural representations help IR approaches go beyond the words matching paradigm by using text representations and embeddings. Thus, standard Bag-Of-Words (BOW) approaches are substituted by neural approaches. There are interaction-based models and representation-based models [29]. Interaction-based models focus on learning interaction between words in queries. On the other hand, representation-based models learn an independent single-vector representation of a query and focus on matching it with documents.

This resulted in different re-ranking pipelines to improve overall IR efficiency [18]. In typical two-stage approaches, the first stage tries to deliver efficient short listing of probable candidates while the second one ranks these probable candidates in a more meticulous manner using costlier approaches. Efficient token-based and vocabulary-centric approaches, such as BM25 and effective dense retrieval approaches, such as those based on BERT, paved the way for two-stage retrieval to strike the balance between efficiency and effectiveness [6]. The efficiency of lexical methods made them the first choice for the retrieval of the initial set of documents, and the effectiveness of dense methods is used to re-rank these results in the second stage [19, 23, 35]. The effectiveness of dense IR methods comes with the cost of efficiency, as inverted index cannot be used. This induced research interest in learning sparse representations for queries and documents using dense methods [8, 26]. Regularization-based learned-sparse models such as SPLADE are effective and efficient, as they utilize the inverted index [7, 8, 17].

The need to perform dense retrieval efficiently led to Approximate Nearest Neighbor (ANN) approaches. Using ANN to reduce the search space and establish first-stage candidate set leads to effective and efficient end-to-end retrieval with high recall [24]. Challenges such as efficiency and scalability have raised the need for models where most of the computation can be performed offline i.e. at indexing time [26]. ANN search builds corpus graphs offline and helps to address efficiency needs during retrieval.

2.3 Approximate Nearest Neighbor Methods

Nearest neighbor search looks for the vectors closest to the query vector in focus. K-Nearest Neighbor is one of the popular information search approaches. This approach relies on K-nearest neighboring documents from the query to retrieve the relevant results [2]. K-Nearest Neighbor search algorithm (K-NNS) performs well in the case of low dimensionality but the complexity of such algorithms restricts their applicability in large-dimensional spaces. To overcome this issue, an approach of K-Approximate Nearest Neighbor search (K-ANNS) is introduced [10]. This relaxes the most vital constraint of the exact search by allowing a small number of deviations [21]. The scaling of algorithmic complexity and the unavailability of global connectivity restrict the applications and accuracy of this approach.

K-ANNS algorithms based on proximity graphs are popularly known as Navigable Small Worlds (NSW) algorithms and help overcome scalability limitations. These graphs are navigable because they have logarithmic or polylogarithmic scaling of the number of hops during greedy traversal with respect to network size [25]. Thus, ANN search relaxes exactness to obtain higher efficiency. It uses vector compression and focuses on searching a relatively smaller part of the database for user query [21]. ANN method parameters control the size of the database subset searched. The larger is the subset, the higher is the effectiveness and latency. Approximation is exploited strictly under the premise of limited compromise in quality [1].

2.3.1 Non-graph-based ANN methods. The methods like tree-based methods, hashing-based methods, and quantization-based methods fall into the category of non-graph-based ANN IR methods [9]. A limitation of these methods is the need to check many nearby cells for improved accuracy. This is because most non-graph-based methods partition the space to answer ANNS queries. Later, they index the resulting partitions. [9]. Unfortunately, indexing partitions to evaluate neighbors to find the nearest neighbors of a given query is always challenging and expensive. These issues with non-graph-based methods are overcome by graph-based methods [22]. Graph-based methods exploit 'neighbor-neighbor' relationship in a better way to approach the query efficiently [34].

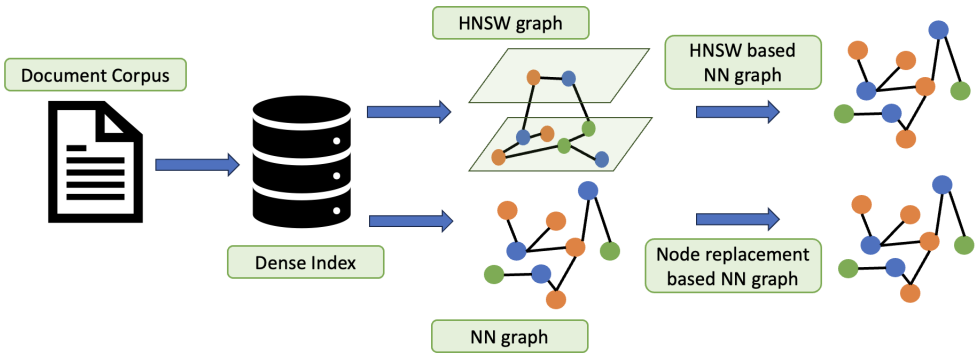


Figure 1: The flow for poor quality graph simulation using: a.) Random replacement strategy with Random, approximate, and malicious nodes and b.) Using the HNSW graph with different ef_construction parameter values reduced from 40 to 10. NN graph: Nearest Neighbor graph.

2.3.2 Graph-based methods HNSW and LADR. There is a trade-off between effectiveness and efficiency in IR methods. ANN methods help to achieve the best possible trade-off. Hierarchical Navigable Small World graphs (HNSW) [25], Adaptive and Proactive Lexically Accelerated Dense Retrieval (LADR) [16] are some of the state-of-the-art algorithms developed with this objective. A fully graph-based incremental K-ANN search approach, HNSW, was proposed to address the limitations in prior methods regarding scalability and efficiency. The HNSW index includes multiple layers of proximity graphs where each graph node represents a document vector. HNSW uses K-ANN graphs and the connected neighbors might not be the ground truth nearest neighbors [25]. Compared to exhaustive dense retrieval, the HNSW-based approach results in substantial latency reduction. HNSW uses random seed points for final exploration with a dense approach. On the other hand, LADR empowers dense retrieval models with bag-of-words based seeding for informed exploration resulting in improved effectiveness [16]. It uses a lexical foundation to identify seed documents to get better insight into potentially relevant documents. The ANN search for relevant documents in the proximity graph starts from these seed documents, leading to informed exploration of the search space. LADR is a framework which can be tuned as per application. LADR has two variants, namely Proactive LADR and Adaptive LADR. Proactive LADR simply expands on the initial seed documents by combining the seed documents and their nearest neighbors. On the other hand, Adaptive LADR instead of simple exploration looks for the neighbors of selected few top seed documents. This process is repeated by exploring the neighbors of the top retrieved set until it becomes stable. In general, in graph-based ANN, precomputed document proximity graphs help to restrict search complexity and retrieval-time computational needs. Further, methods like LexBoost demostrate the use of corpus graphs to boost lexical retrieval without the need of a dense reranker [15].

We investigate the performance of graph-based ANN IR methods HNSW, Proactive LADR, and Adaptive LADR with reference to graph quality and graph traversal approaches. ANN search efficiency and effectiveness are governed by the quality of the originating search node(s) and their neighbors. These neighboring nodes in a corpus graph provide context, namely the relatedness, among the nodes. Even though corpus graphs are constructed offline, the cost is often prohibitively large in high-volume data scenarios. The motivation of this paper is to identify the optimal quality of graph and the most suitable exploration approach. Further, the investigation also aims to evaluate robustness of these approaches with reference to quality of graph, which is extremely crucial for practical billion-scale applications.

3 Evaluation

To evaluate the performance of graph-based ANN IR methods, we focused on graph quality and graph traversal.

3.1 Graph Simulation for Graph Quality

To simulate poor quality graphs, we use two different strategies. One is node replacement strategy, and another is ef_construction parameter-based strategy. Here, ef_construction parameter controls the number of nearest neighbors considered while building the graph using HNSW [22]. Here, ef stands for expansion factor. Figure 1 depicts the flow for poor quality graph simulation using a.) Random, approximate, and malicious node replacement and b.) Using the HNSW graph with different ef_construction parameter values. This also mirrors real-world scenarios with missing documents and indexing errors.

For poor graph simulation through node replacement, we used the node replacement strategy in three different ways. In **replacement with a random node**, a node from the corpus is randomly sampled to replace the relevant node. In **replacement with an approximate similar node**, a node almost similar to the existing node is used for the replacement. In **replacement with a malicious node**, a possible worst node is chosen, i.e., the node farthest in relationship, for the replacement. This is done by considering the distance in embedding space. In addition, to increase impurity, we increase the percentage of nodes replaced. We evaluate the impact of the degrading quality of graphs on ANN effectiveness.

In another experiment, graphs are simulated using HNSW with different ef_construction values. In an HNSW graph, ef_construction value controls the dynamic list of nodes for constructing the graph. This, in fact, determines the number of nearest neighbors considered

when adding a new node. Hence, ef_construction value controls the quality of graph. Higher ef_construction values improve the quality of the graph. We reduced the ef_construction values from 40 to 10 in steps to simulate different poor quality graphs with deteriorating quality. These simulated poor-quality graphs are used for experimentation to test the robustness of various ANN approaches for standard performance measures. We use single-representation dense retrieval model TAS-B [11] for indexing and graph building across methods.

3.2 Graph Traversal

Graph traversal approach is an important component of ANN search methods. Typically, ANN traversal techniques are adapted variants of greedy search across layers [9, 32]. To investigate the impact of graph traversal strategies, we adapted different graph traversal approaches like Best First Search, Hill Climbing, and A* [5] to analyze their impact on the performance of Adaptive LADR with the changing quality of the graphs for each of them.

In the Best First Search approach, the best neighboring node is picked to continue the hunt for the best node until the goal state is reached. In the Hill Climbing search, the depth-first search traversal is undertaken with selection of the next move based on the neighborhood. In the A* search, the heuristic value is clubbed with the cost incurred. Here, the cost incurred is a function of the distance from the seed node. We adapted these approaches for Adaptive LADR and evaluated their performance on graphs of different qualities. In case of the Hill Climbing and Best First search, the tie (multiple neighboring nodes have the same evaluation score) is resolved by random selection of a node, but in case of the A* search, node closer to the seed node is selected.

3.3 Datasets and Evaluations

We used two standard datasets, **TREC DL 2019**: TREC 2019 Deep Learning (Passage Subtask) and **TREC DL 2020**: TREC 2020 Deep Learning (Passage Subtask). These are the official evaluation query sets used in the TREC 2019 [3] and TREC 2020 [4] Deep Learning shared tasks. While TREC DL 2019 has 43 manually-judged queries using four relevance grades (215 relevance assessments per query, on average) TREC DL 2020 has 54 queries with manual judgments (211 relevance assessments per query, on average). Annotations for both aforementioned query sets were performed by National Institute of Standards and Technology (NIST) annotators.

We evaluate using nDCG@10, nDCG@1000 and Recall@1000 for both TREC DL 2019 and 2020. The analysis is performed for nDCG@1000 and Recall@1000. Since for many precision-oriented tasks like Conversational IR, a few top results matter, we also evaluate nDCG@10 to compare robustness across degrading graph quality.

3.4 Experiment

We evaluate the robustness of the state-of-the-art graph-based ANN IR methods. We evaluate them with different input parameters and their combinations with poor graph simulation strategies. We also evaluate the performance of Adaptive LADR with different graph qualities for different graph traversal strategies. We address the following research questions:

RQ1: How do graph-based ANN IR methods perform on poor quality corpus graphs simulated using node replacement strategies?
RQ2: How do graph-based ANN IR methods perform on poor quality corpus graphs simulated using HNSW with different ef_construction parameter values?
RQ3: How do the number of neighbors and seed set size affect the performance of LADR on poor quality graphs?
RQ4: How does the robustness of LADR compare to HNSW with top results (nDCG@10) in consideration across different graph qualities?
RQ5: What is the trade-off between graph-based ANN search effectiveness and graph construction latency?
RQ6: How does the graph exploration strategy affect the performance of LADR across different graph qualities?

We have released the code to reproduce the results of our experiments.[1]

4 Results and Analysis

We now discuss and analyse of the results of graph-based ANN search robustness with reference to poor quality corpus graphs across parameters. ANN effectiveness and graph construction latency tradeoff is also established and analyzed. We also analyze how graph-based ANN effectiveness varies with different heuristic search-based graph traversal methods.

4.1 RQ1: Node replacement policy impact

This RQ focuses on the performance impact of node replacement policy. We simulated poor quality corpus graphs using three replacement strategies, namely, replacement with a random node, replacement with an approximately similar node, and replacement with a malicious node. In steps, we increase the percentage of nodes replaced with poor-quality nodes to evaluate the impact of degrading quality of graphs on ANN effectiveness. In Table 1, we show the effectiveness of HNSW, Proactive LADR and Adaptive LADR with changing graph quality through node replacement strategies. nDCG@1k and Recall@1k are given for random, approximate, and malicious node replacement strategies with varying parameters and changing the percentage of nodes replaced. We conducted experiments with 16 and 64 neighbors for Proactive LADR, and Adaptive LADR. We used ef_search value of 16, 64 and 'no space bounded queue (nsbq)' for HNSW experiments. ef_search controls the size of candidate list considered during search process in HNSW.

From Table 1, we note that nDCG@1k and Recall@1k decrease as the quality of the graphs degrades for HNSW. For Proactive and Adaptive LADR however, the degrading graph quality leads to comparatively less decrease in nDCG@1k and Recall@1k irrespective of node replacement strategies. On TREC DL 2019 query set, for HNSW with no space bounded queue, nDCG@1k reduces to 0.179 from 0.649 and Recall@1k reduces to 0.137 from 0.742 with 80% malicious replacement. For the same setting, in Adaptive LADR with 64 neighbors, nDCG@1k reduces to 0.695 from 0.724 and Recall@1k reduces to 0.779 from 0.857. Similar trends are observed on TREC DL 2020. Table 1 shows that LADR variants are more robust

[1] https://github.com/Georgetown-IR-Lab/Graph-Quality

Dataset			TREC DL 2019						TREC DL 2020					
Metric			nDCG@1k			Recall@1k			nDCG@1k			Recall@1k		
Method	n	Graph	0%	40%	80%	0%	40%	80%	0%	40%	80%	0%	40%	80%
HNSW	16	random	**0.303**	0.186	0.059	**0.303**	0.193	0.062	**0.411**	0.251	0.087	**0.455**	0.273	0.108
		approx	**0.303**	0.199	0.101	**0.303**	0.204	0.105	**0.411**	0.272	0.119	**0.455**	0.285	0.149
		malicious	**0.303**	0.187	0.064	**0.303**	0.179	0.048	**0.411**	0.263	0.099	**0.455**	0.282	0.092
	64	random	**0.535**	0.333	0.100	**0.579**	0.364	0.103	**0.544**	0.336	0.121	**0.631**	0.385	0.144
		approx	**0.535**	0.354	0.197	**0.579**	0.391	0.241	**0.544**	0.359	0.169	**0.631**	0.410	0.250
		malicious	**0.535**	0.340	0.144	**0.579**	0.341	0.114	**0.544**	0.360	0.142	**0.631**	0.392	0.123
	nsbq	random	**0.649**	0.403	0.118	**0.742**	0.468	0.135	**0.673**	0.410	0.135	**0.792**	0.482	0.159
		approx	**0.649**	0.426	0.253	**0.742**	0.505	0.336	**0.673**	0.449	0.234	**0.792**	0.512	0.350
		malicious	**0.649**	0.420	0.179	**0.742**	0.429	0.137	**0.673**	0.457	0.172	**0.792**	0.498	0.148
Proactive LADR	16	random	**0.728**	0.714	0.692	**0.847**	0.813	0.774	**0.730**	0.716	0.696	**0.878**	0.852	0.819
		approx	**0.728**	0.712	0.706	**0.847**	0.816	0.801	**0.730**	0.713	0.707	**0.878**	0.849	0.836
		malicious	**0.728**	0.712	0.697	**0.847**	0.819	0.778	**0.730**	0.713	0.700	**0.878**	0.854	0.823
	64	random	**0.725**	0.709	0.690	**0.850**	0.808	0.776	**0.731**	0.716	0.695	**0.889**	0.859	0.818
		approx	**0.725**	0.707	0.706	**0.850**	0.814	0.810	**0.731**	0.713	0.709	**0.889**	0.855	0.840
		malicious	**0.725**	0.706	0.695	**0.850**	0.816	0.779	**0.731**	0.713	0.699	**0.889**	0.863	0.827
Adaptive LADR	16	random	**0.729**	0.712	0.691	**0.860**	0.812	0.778	**0.739**	0.719	0.696	**0.903**	0.861	0.819
		approx	**0.729**	0.709	0.701	**0.860**	0.817	0.793	**0.739**	0.719	0.709	**0.903**	0.863	0.840
		malicious	**0.729**	0.709	0.697	**0.860**	0.819	0.779	**0.739**	0.719	0.698	**0.903**	0.871	0.818
	64	random	**0.724**	**0.712**	0.694	**0.857**	0.823	0.779	**0.735**	0.719	0.696	**0.904**	0.869	0.823
		approx	**0.724**	**0.709**	0.704	**0.857**	0.826	0.812	**0.735**	0.717	0.710	**0.904**	0.863	0.846
		malicious	**0.724**	**0.710**	0.695	**0.857**	**0.833**	0.779	**0.735**	0.717	0.703	**0.904**	0.873	0.832

Table 1: Retrieval effectiveness of ANN methods across graphs of different qualities simulated using 'replacement' strategy. n is the ANN method parameter. For HNSW n is ef_search and for LADR variants n is the number of neighbors. In random replacement, we replace the nodes in the graph randomly from the document corpus. In approx replacement, we use the 128^{th} ranked neighbor for replacement. In malicious replacement, we use the farthest document from the corpus for replacement. 0%, 40% and 80% replacement results are shown above. Statistically equivalent values in each row are denoted in 'bold' using TOST p<0.05. 0% replacement implies that the original high quality graph is used. With an increase in the percentage of nodes replaced (graph degradation) we can observe a decrease in both nDCG@1k and Recall@1k for both HNSW and LADR variants. But LADR result degradation is lower compared to that of HNSW making it more robust.

than HNSW when evaluated on poor graphs simulated using node replacement strategy addressing RQ1.

4.2 RQ2: HNSW-based (ef_construction) poor quality corpus graph impact

We simulate poor quality corpus graphs using HNSW with different ef_construction values. Table 2 shows the effectiveness of HNSW, Proactive LADR and Adaptive LADR across different quality graphs simulated using HNSW with different ef_construction parameter values. nDCG@10, nDCG@1k and Recall@1k results are shown across ef_construction and ANN parameter values. The ef_construction values are reduced from 40 to 10 to construct different poor quality graphs. As we reduce the ef_construction value, nDCG@1k results are compromised in the case of both HNSW and LADR variants. However, a lower impact on LADR variant results is observed. These trends hold on both TREC DL 2019 and TREC DL 2020 query sets. Similarly, Recall@1k decreases for both HNSW and LADR variants as we reduce ef_construction, but LADR variants are more robust than HNSW.

For TREC DL 2019, when ef_construction value is reduced from 40 to 10, nDCG@1k for HNSW (nsbq) reduces to 0.466 from 0.671 while Recall@1k reduces to 0.467 from 0.782. On the other hand, in the same setting, nDCG@1k for Adaptive LADR (64 neighbors) reduces to 0.716 from 0.729 while Recall@1k reduces to 0.832 from 0.856. Similar trends are observed on TREC DL 2020. The majority of the nDCG@1k and Recall@1k results produced with low-quality graphs for both LADR variants are statistically equivalent. Table 1 and Table 2 together show that the impact on the results and the trend of change in performance remain the same regardless of the poor graph simulation strategies. Further, LADR variants are more robust than HNSW on poor graphs simulated by varying ef_construction value addressing RQ2.

4.3 RQ3: Impact of the seed set size and number of neighbors

This RQ evaluates the impact of the size of the seed set and the number of neighbors considered on the robustness of the method for poor quality graphs. We perform a grid search by varying the seed node set size and number of neighbors in order to analyze the performance of Proactive and Adaptive LADR for different poor quality graphs. Figure 2 shows a heat map depicting nDCG@1k and Recall@1k over various combinations of the above parameters

Dataset			TREC DL 2019				TREC DL 2020			
Setup			ef construction value							
Metric	Method	n	40	30	20	10	40	30	20	10
nDCG@10	HNSW	16	**0.505**	**0.473**	**0.440**	0.269	**0.659**	0.589	0.497	0.218
		32	**0.579**	**0.491**	**0.482**	0.311	**0.659**	**0.631**	0.549	0.254
		64	**0.611**	**0.569**	**0.545**	0.354	**0.680**	0.634	0.600	0.360
		nsbq	**0.671**	**0.669**	**0.674**	0.579	**0.681**	**0.681**	**0.681**	0.612
	Proactive LADR	16	**0.713**	**0.707**	**0.710**	**0.705**	**0.688**	**0.693**	**0.690**	**0.684**
		64	**0.708**	**0.711**	**0.713**	**0.706**	**0.688**	**0.690**	**0.688**	**0.684**
	Adaptive LADR	16	**0.710**	**0.707**	**0.713**	**0.708**	**0.685**	**0.689**	**0.689**	**0.684**
		64	**0.711**	**0.708**	**0.710**	**0.701**	**0.685**	**0.687**	**0.689**	**0.683**
nDCG@1k	HNSW	16	**0.481**	**0.448**	0.364	0.165	**0.649**	0.550	0.451	0.144
		32	**0.577**	0.486	0.425	0.206	**0.679**	0.629	0.521	0.187
		64	**0.614**	**0.567**	0.515	0.238	**0.700**	0.645	0.584	0.275
		nsbq	**0.671**	**0.663**	**0.645**	0.466	**0.700**	**0.696**	0.682	0.532
	Proactive LADR	16	**0.722**	**0.717**	0.714	0.706	**0.719**	**0.722**	**0.720**	0.710
		64	**0.723**	**0.723**	**0.723**	**0.716**	**0.728**	**0.729**	**0.726**	**0.718**
	Adaptive LADR	16	**0.729**	**0.723**	**0.723**	0.712	**0.728**	**0.730**	**0.721**	0.710
		64	**0.729**	**0.727**	**0.729**	**0.716**	**0.727**	0.730	**0.729**	**0.719**
Recall@1k	HNSW	16	**0.536**	**0.486**	0.374	0.141	**0.759**	0.624	0.505	0.119
		32	**0.659**	0.534	0.447	0.193	**0.805**	0.730	0.600	0.163
		64	**0.707**	**0.633**	0.564	0.221	**0.836**	0.752	0.677	0.255
		nsbq	**0.782**	**0.762**	0.738	0.467	**0.848**	**0.843**	**0.829**	0.584
	Proactive LADR	16	**0.840**	0.833	0.819	0.805	**0.865**	**0.868**	**0.858**	0.843
		64	**0.845**	**0.841**	**0.842**	**0.826**	**0.887**	**0.889**	**0.880**	**0.867**
	Adaptive LADR	16	**0.854**	**0.847**	0.836	0.813	**0.883**	**0.885**	**0.864**	0.846
		64	**0.856**	**0.855**	**0.856**	**0.832**	**0.890**	**0.893**	**0.890**	**0.868**

Table 2: Retrieval effectiveness of ANN methods across graphs of different quality created using HNSW with different ef_construction values. Lower the ef_construction value poorer is the quality of graph. n is the ANN method parameter. For HNSW, n is ef_search and for LADR variants, n is the number of neighbors. Statistically equivalent values in each row are denoted in 'bold'. For both HNSW and LADR variants, results degrade with reduction in ef_construction value. However, LADR results are more robust than HNSW and hence witness lower impact. Both Proactive and Adaptive LADR show statistical equivalence in all cases with 64 number of neighbors.

Method	Graph	n	TREC DL19	TREC DL20
Proactive LADR	exhaustive	16	88.14%	89.07%
		64	86.74%	88.33%
	HNSW	16	88.37%	91.67%
		64	85.81%	88.70%
Adaptive LADR	exhaustive	16	85.35%	89.26%
		64	85.12%	88.15%
	HNSW	16	85.35%	89.82%
		64	84.65%	87.78%

Table 3: Percentage of documents in the top ten results that are a part of BM25 seed set. Our analysis considered both exhaustive (exact neighbors) and HNSW-based graphs. n is the number of neighbors considered. In both cases, approximately 80 to 90% of documents are part of the BM25 seed documents. Further, we note that the percentage is slightly greater when the number of neighbors lower.

for the TREC DL 2019 query set. Our evaluations were conducted on graphs simulated using HNSW for ef_construction values of 40 and 10. Here, we vary the number of seeds from 200 to 1000 at intervals of 200 and number of neighbors in 8, 16, 32, 64. The darker the shade of blue, the higher the effectiveness in terms of both nDCG@1k and Recall@1k. For the high quality corpus graph (ef_construction=40) for Proactive LADR with 64 neighbors, a lower number of seeds leads to higher effectiveness.

In Figure 2, we observe that on the high quality graph (i.e. ef_construction=40), seed size of 200 and 64 neighbors leads to nDCG@1k of 0.735 and Recall@1k of 0.858. In all other cases for both Proactive and Adaptive LADR, the general trend is that with increasing number of seeds and neighbors, the effectiveness increases. For example, in Adaptive LADR when we increase seed size from 200 to 1000 and neighbors from 8 to 64, nDCG@1k increases from 0.675 to 0.729 on high quality graph while it increases from 0.622 to 0.716 in poor quality graph (ef_construction=10). In a similar setting, for Adaptive LADR, Recall@1k increases from 0.760 to 0.849 for high quality graph while it increases from 0.674 to 0.832 in poor quality graph. From these results we infer that the trends across seed set size and number of neighbors remain the same for both poor and high quality graphs. We also note that the result

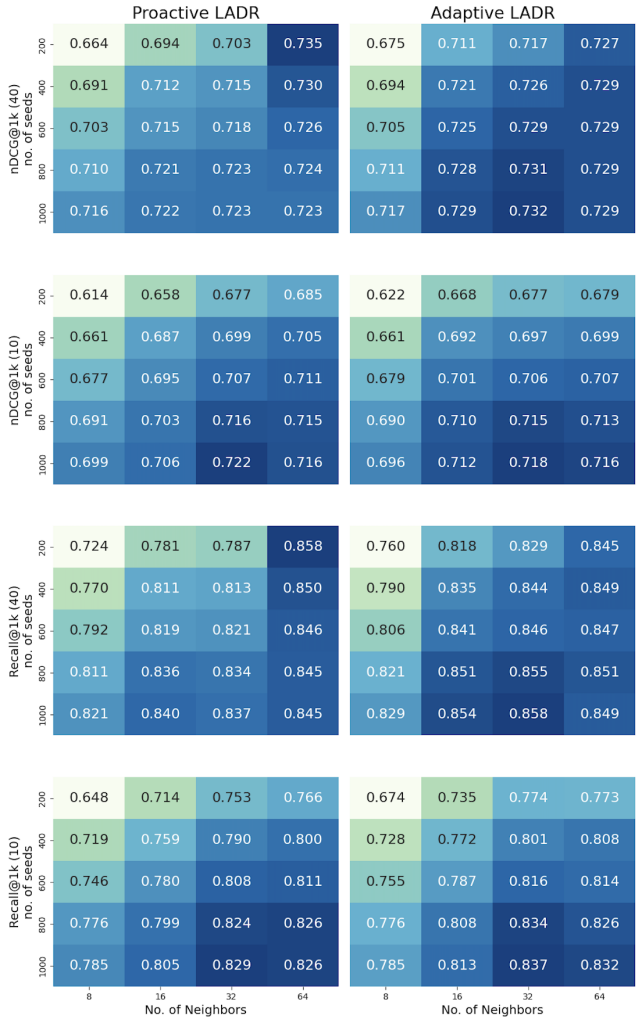


Figure 2: Comparison of Proactive and Adaptive LADR effectiveness on high quality (ef_construction=40) and poor quality (ef_construction=10) graphs across grid search with seed size in [200, 400, 600, 800, 1000] and number of neighbors in [8, 16, 32, 64]. We evaluate and analyze nDCG@1k and Recall@1k across parameter combinations on TREC DL 2019 query set. In the above heat map, we observe a general trend of improving effectiveness with increasing seed set size and number of neighbors. Further, we also note lower effectiveness for poor quality graphs.

values for poor quality graphs when compared with respective values from high quality graphs are lower as expected.

4.4 RQ4: Robustness of LADR as compared to HNSW across graph quality for top (nDCG@10) results

This RQ evaluates the change in nDCG@10 effectiveness across poor quality graphs and analyzes ANN method design aspects

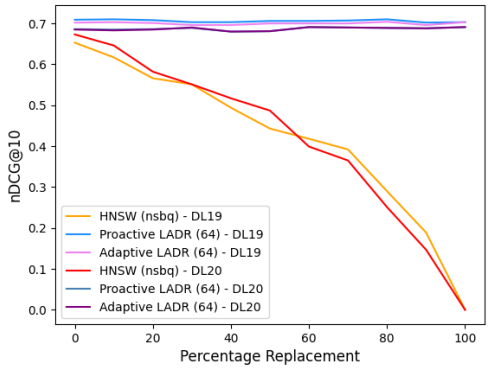


Figure 3: nDCG@10 vs Percentage Replacement for malicious node replacement to simulate poor quality graphs. DL19: TREC DL 2019 query set, DL20: TREC DL 2020 query set. Proactive and Adaptive LADR evaluated with 64 neighbors. HNSW evaluated with 'no space bounded queue' setting. Drop in nDCG@10 for HNSW and robustness of LADR variants with increasing percentage replacement are clearly evident.

contributing to the performance. We also analyze the impact of the number of neighbors considered in Proactive and Adaptive LADR, and ef_search parameter in HNSW on the top results. The quality of top results (nDCG@10) with respect to the percentage of nodes replaced is depicted in Figure 3 for HNSW, Proactive LADR, and Adaptive LADR on both TREC DL 2019 and 2020 query sets. For both LADR variants, regardless of the datasets, there is no performance compromise despite increasing percentage replacement. For HNSW however, there is a clear drop in performance with an increasing percentage of replacement. As evident in Figure 3, nDCG@10 even after 80% node replacement is 0.710 and 0.704 for Proactive and Adaptive LADR with 64 neighbors, respectively. Although it is 0.703 at 100% replacement for both LADR variants. On the other hand, nDCG@10 for 80% replacement for HNSW (nsbq) drops to 0.290 from 0.653. These results are for the TREC DL 2019 query set. This primarily sheds light on the minimal degradation in LADR performance regardless of poor graph quality.

Table 2 also shows nDCG@10 results for HNSW, Proactive LADR and Adaptive LADR for various parameter values. Across graphs of different qualities (ef_construction values from 40 to 10), Proactive and Adaptive LADR yields statistically equivalent results as shown in bold. On the other hand, for HNSW the drop in nDCG@10 is higher with a statistically significant drop for ef_construction value of 10 from 40 further supporting robustness of LADR variants over HNSW.

Table 3 shows the percentage of documents in the top ten results that are part of the BM25 seed set. In the case of Proactive LADR, irrespective of the graph construction strategy, approximately 85 to 89% of the documents in the top 10 results come from the seed nodes for TREC DL 2019 and approximately 88 to 92% of the documents in the top 10 results come from the seed nodes for TREC DL 2020. Similarly, for Adaptive LADR around 84 to 86% for TREC DL 2019 and 87 to 90% for TREC DL 2020 of the documents in the top 10

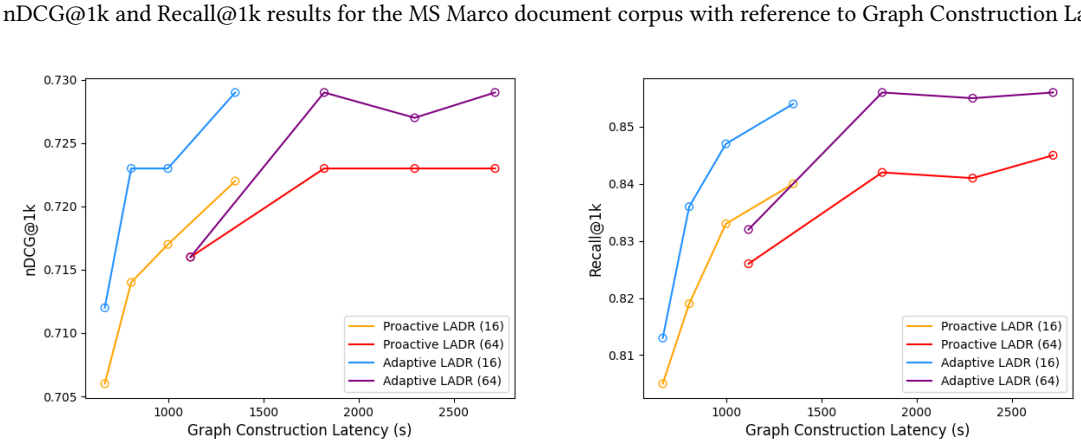


Figure 4: nDCG@1k (left) and Recall@1k (right) results for the MS Marco document corpus with reference to Graph Construction Latency (in seconds). Here, the nearest neighbor graphs of different qualities have been simulated using the popular HNSW based simulation with different ef_construction parameter values 10, 20, 30, 40. With increase in graph construction latency, better quality graphs are created leading to better nDCG@1k and Recall@1k results. Also, Adaptive LADR leads to better results with the same graph when compared to Proactive LADR as expected. From the plots, we can infer that graph construction can be tuned to make retrieval more efficient while maintaining acceptable loss in effectiveness.

results are from seed nodes. This shows that for LADR variants, the seed nodes play an important role for the top results, especially for lower values of the number of neighbors. These trends are observed in both the TREC DL 2019 and 2020 query sets. The robustness in the top 10 results of the LADR variants can be attributed to the mechanism of considering BM25 seed documents that have a prominent presence at the top, as evident in Table 3. This underlines the utility of LADR for precision-oriented applications where only the top results are consequential; thus, the robustness of LADR with reference to the nDCG@10 results addresses RQ4.

4.5 RQ5: The trade-off between ANN effectiveness and graph construction latency

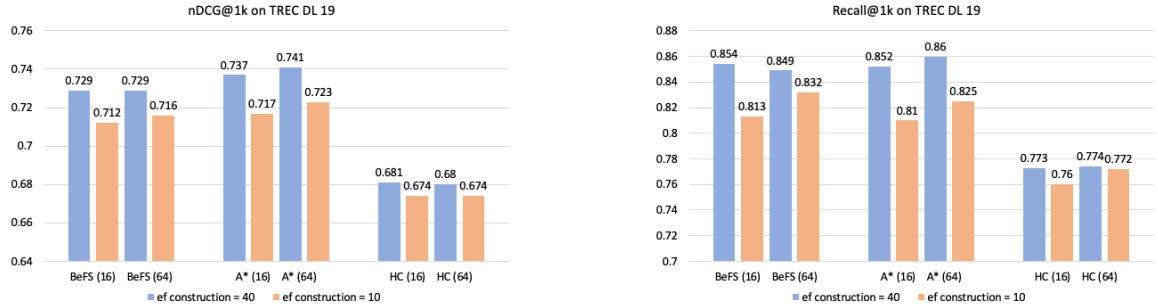
After establishing the robustness of the graph-based ANN IR methods Proactive and Adaptive LADR regardless of graph quality, we analyzed the savings in graph construction costs. This RQ focuses on analyzing the effectiveness and graph construction latency trade-off. Query-time latency remains unaffected across graphs of different qualities. Figure 4 shows the nDCG@1k and Recall@1k versus graph construction latency for Proactive and Adaptive LADR. As seen in Figure 4, there are significant savings in terms of graph construction time with a poor quality graph without greatly compromising effectiveness. The major leap of improvement in efficiency is observed while going from ef_construction value of 40 to 20. Especially, Adaptive LADR with 64 neighbors leads to no decrease in Recall@1k and nDCG@1k when ef_construction value is reduced from 40 to 20. This comes with a 899-second (33%) decrease in graph construction latency. Similar trends are observed for Proactive LADR (64 neighbors) with no decrease in nDCG@1k and decrease by 0.03 in Recall@1k when ef_construction value is reduced from 40 to 20 while saving 899 seconds (33%) in graph construction time. We recommend ef_construction values between 20 and 30 for efficient graph construction while maintaining robust performance.

Further reducing ef_construction value (20 to 10) leads to even more efficient graph construction but also results in a reduction in effectiveness. For example, Adaptive LADR with 64 neighbors saves an additional 703 seconds (25.9%) while suffering 0.013 and 0.024 reduction in nDCG@1k and Recall@1k, respectively. Thus, there is a clear trade-off between graph construction latency and effectiveness. The recommended parameters can help practitioners tune the construction methods for their specific applications. Optimal performance is observed for ef_construction values in the range of 20 to 30 addressing RQ5.

4.6 RQ6: Graph exploration strategies

RQ6 focuses on the analysis of the graph-based ANN search results with reference to different graph exploration policies. Three heuristic search and exploration approaches, namely Best First Search, Hill Climbing Search, and A* search, are adapted and implemented for exploration in the case of Adaptive LADR. Figure 5 shows the performance of LADR (nDCG@1k and Recall@1k) on the TREC DL 2019 and TREC DL 2020 query sets. The Best First Search method and the A* Search method have comparable results for nDCG@1k for both the TREC DL 2019 and TREC DL 2020 query sets. A similar trend is observed for Recall@1k. On TREC DL 2019, for Adaptive LADR with 64 neighbors, when the ef_construction value is reduced to 10 from 40, nDCG@1k reduces to 0.716 from 0.729 for the Best First Search strategy. While for A*, nDCG@1k reduces to 0.723 from 0.741 under the same settings. Similarly, for the Best First Search, Recall@1k reduces to 0.832 from 0.849 and for A*, Recall@1k reduces to 0.825 from 0.860. For both of these approaches, there is a slight reduction in performance for poor-quality graphs constructed when the ef_construction value is reduced to 10. However, the use of Hill Climbing Search results in lower performance

nDCG@1k and Recall@1k results on TREC DL 2019 dataset for Best First Search, A* and Hill Climbing heuristic search methods in Adaptive LADR across poor and high quality graphs



nDCG@1k and Recall@1k results on TREC DL 2020 dataset for Best First Search, A* and Hill Climbing heuristic search methods in Adaptive LADR across poor and high quality graphs

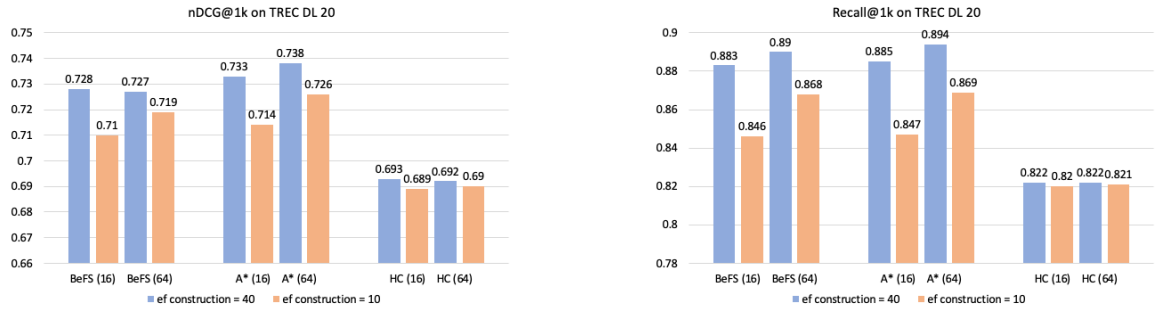


Figure 5: nDCG@1k and Recall@1k results on TREC DL 2019 and 2020 for Best First Search (BeFS), A* and Hill Climbing (HC) heuristic search methods in Adaptive LADR. ef_construction = 40 is used to simulate high quality graph while ef_construction = 10 is used to simulated poor quality graph. n =16 and 64 are parameters for respective heuristic search methods. We observe higher effectiveness of BeFS and A* when compared with HC. Further, we also note that HC is more robust to poor quality graphs than the other two heuristic search methods.

across query sets. This can be attributed to the depth-first nature of the Hill Climbing Search which is counterintuitive to the Clustering Hypothesis [12]. Interestingly, the impact on the performance due to the graph quality is negligible for Hill Climbing Search as compared to the two other search approaches under consideration. In the same setting as above, for the Hill Climbing search strategy, nDCG@1k reduces to 0.674 form 0.68 while Recall@1k reduces to 0.772 from 0.774. The robustness here is a result of depth first nature not making most out of the clustering hypothesis and hence is less affected by graph quality. Figure 5 provides a detailed performance analysis of poor graph qualities with reference to the impact of graph exploration strategies addressing RQ6.

5 Discussion and Conclusion

Graph-based ANN search methods in IR like HNSW, Proactive LADR and Adaptive LADR try to strike a balance between effectiveness and efficiency. The quality of the graph and its efficient exploration are the two pillars of the success of these approaches. We presented a detailed investigation and extensive evaluation on this front. We simulated graphs of varying qualities using different approaches. We demonstrated the use of different node replacement approaches and different ef_construction values (through HNSW) to simulate graphs with deteriorating qualities. The study was carried out to evaluate the robustness of graph-based ANN search methods. We conclude that Proactive LADR and Adaptive LADR are robust to a great extent, regardless of the quality of the graph. Using both LADR variants, we can achieve statistically equivalent effectiveness even in the case of compromised poor-quality graphs, especially for top results (nDCG@10) while saving up to 33% of graph construction time. However, that is not the case for HNSW, where effectiveness deteriorates significantly with a reduction in graph quality. Moreover, the effectiveness of LADR variants is not impacted much by the graph traversal method. A* and Best First Search deliver higher performance as compared to the Hill-Climbing Search. The depth-first nature of Hill Climbing with the constrained locality information restricting the selection of neighbors is the striking reason for its low performance.

Finally, our efforts also establish the utility of LADR in case of unavailability of good quality graphs in real-life scenarios with a predominance of faulty agents and poor signals.

References

[1] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2018. ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. arXiv:1807.05614 [cs.IR] https://arxiv.org/abs/1807.05614

[2] George H. Chen and Devavrat Shah. 2018. . doi:10.1561/2200000064

[3] Nick Craswell et al. 2020. Overview of the TREC 2019 deep learning track. doi:10.48550/ARXIV.2003.07820

[4] Nick Craswell et al. 2021. Overview of the TREC 2020 deep learning track. doi:10.48550/ARXIV.2102.07662

[5] Stefan Edelkamp and Stefan Schrödl. 2011. *Heuristic search: theory and applications.* Elsevier.

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

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2353–2359. doi:10.1145/3477495.3531857

[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. doi:10.1145/3404835.3463098

[9] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2018. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. arXiv:1707.00143 [cs.LG] https://arxiv.org/abs/1707.00143

[10] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2, Article 137 (June 2023), 27 pages. doi:10.1145/3589282

[11] Sebastian Hofstätter et al. 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)*. 113–122.

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

[13] Zhuoran Ji and Cho-Li Wang. 2022. Efficient exact K-nearest neighbor graph construction for billion-scale datasets using GPUs with tensor cores. In *Proceedings of the 36th ACM International Conference on Supercomputing* (Virtual Event) *(ICS '22)*. Association for Computing Machinery, New York, NY, USA, Article 10, 12 pages. doi:10.1145/3524059.3532368

[14] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: re-rank with source coding. arXiv:1102.3828 [cs.IR] https://arxiv.org/abs/1102.3828

[15] Hrishikesh Kulkarni, Nazli Goharian, Ophir Frieder, and Sean MacAvaney. 2024. LexBoost: Improving Lexical Document Retrieval with Nearest Neighbors. In *Proceedings of the ACM Symposium on Document Engineering 2024* (San Jose, CA, USA) *(DocEng '24)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. doi:10.1145/3685650.3685658

[16] 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* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 152–162. doi:10.1145/3539618.3591715

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

[18] Jaewoo Lee, Joonho Ko, Jinheon Baek, Soyeong Jeong, and Sung Ju Hwang. 2024. Unified Multi-Modal Interleaved Document Representation for Information Retrieval. *ArXiv* abs/2410.02729 (2024). https://api.semanticscholar.org/CorpusID:273098663

[19] 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. doi:10.1145/3485447.3511955

[20] Cheng Li, Marc Najork, Mike Bendersky, Mingyang Zhang, and Saar Kuzi. 2020. Leveraging Semantic and Lexical Matching to Improve the Recall of Retrieval Systems: A Hybrid Approach.

[21] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[22] Jun Liu, Zhenhua Zhu, Jingbo Hu, Hanbo Sun, Li Liu, Lingzhi Liu, Guohao Dai, Huazhong Yang, and Yu Wang. 2022. Optimizing Graph-based Approximate Nearest Neighbor Search: Stronger and Smarter. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. 179–184. doi:10.1109/MDM55031.2022.00045

[23] 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. doi:10.1145/3397271.3401262

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

[25] 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).

[26] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. 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, 1723–1727. doi:10.1145/3404835.3463030

[27] Tomas Mikolov et al. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[28] Hiroyuki Ootomo, Akira Naruse, Corey Nolet, Ray Wang, Tamas Feher, and Yong Wang. 2024. CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search for GPUs. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 4236–4247. doi:10.1109/ICDE60146.2024.00323

[29] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. In *Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval*. https://arxiv.org/abs/1606.04648

[30] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3 (2009), 333–389.

[31] Luo Si and Rong Jin. 2011. Machine learning for information retrieval. 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, 1293–1294. doi:10.1145/2009916.2010167

[32] Nicola Tonellotto. 2022. Lecture Notes on Neural Information Retrieval. arXiv:2207.13443 [cs.IR] https://arxiv.org/abs/2207.13443

[33] 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.

[34] Hui Wang, Yong Wang, and Wan-Lei Zhao. 2022. Graph-based Approximate NN Search: A Revisit. arXiv:2204.00824 [cs.IR] https://arxiv.org/abs/2204.00824

[35] Xiao Wang et al. 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. doi:10.1145/3477495.3531721

[36] Ryen W. White. 2024. Tasks, Copilots, and the Future of Search: A Keynote at SIGIR 2023. *SIGIR Forum* 57, 2, Article 4 (Jan. 2024), 8 pages. doi:10.1145/3642979.3642985