Down with the Hierarchy: The 'H' in HNSW Stands for "Hubs"

Blaise Munyampirwa [1] **Vihan Lakshman** [2] **Benjamin Coleman** [3]

Abstract

Driven by recent breakthrough advances in neural representation learning, approximate near-neighbor (ANN) search over vector embeddings has emerged as a critical computational workload. With the introduction of the seminal Hierarchical Navigable Small World (HNSW) algorithm, graph-based indexes have established themselves as the overwhelmingly dominant paradigm for efficient and scalable ANN search. As the name suggests, HNSW searches a layered hierarchical graph to quickly identify neighborhoods of similar points to a given query vector. But is this hierarchy even necessary? A rigorous experimental analysis to answer this question would provide valuable insights into the nature of algorithm design for ANN search and motivate directions for future work in this increasingly crucial domain. We conduct an extensive benchmarking study covering more large-scale datasets than prior investigations of this question. We ultimately find that a flat navigable small world graph graph retains all of the benefits of HNSW on high-dimensional datasets, with latency and recall performance essentially *identical* to the original algorithm but with less memory overhead. Furthermore, we go a step further and study *why* the hierarchy of HNSW provides no benefit in high dimensions, hypothesizing that navigable small world graphs contain a well-connected, frequently traversed "highway" of hub nodes that maintain the same purported function as the hierarchical layers. We present compelling empirical evidence that the *Hub Highway Hypothesis* holds for real datasets and investigate the mechanisms by which the highway forms. The implications of this hypothesis may also provide future research directions in developing enhancements to graph-based ANN search.

[1]Argmax Inc., Mountain View, CA [2]MIT CSAIL, Cambridge, MA [3]Google DeepMind, Mountain View, CA. Correspondence to: Blaise Munyampirwa <blaisemunyampirwa@gmail.com>.

Proceedings of the 1st Workshop on Vector Databases at International Conference on Machine Learning, 2025. Copyright 2025 by the author(s).

1. Introduction

Near neighbor search is a fundamental problem in computational geometry that lies at the heart of countless practical applications. From industrial-scale recommendation (Feng et al., 2022) to retrieval-augmented generation (Lewis et al., 2020) and even to computational biology (Zhao et al., 2024), numerous data-intensive tasks utilize similarity search at some location in the stack. As a result, similarity indexes are very well-studied (Guo et al., 2019; Malkov & Yashunin, 2016; Johnson et al., 2017; Aguerrebere et al., 2023; Jayaram Subramanya et al., 2019) with multiple large-scale benchmarks and leaderboards to compare techniques (Aumüller et al., 2018; Simhadri et al., 2022).

Historically, the state-of-the-art for near neighbor search involved constructing sophisticated tree-based data structures, such as kd-trees (Bentley, 1975) and cover trees (Beygelzimer et al., 2006), that guaranteed exact solutions while avoiding a brute-force examination of all points. However, the recent advent of large-scale neural representation learning, including large language models (LLMs), places a significant strain on these classical methods that were developed to target a much lower-dimensional search space. In response, the community has turned to approximate search methods. While alternative approximate indexing methods such as locality-sensitive hashing (Indyk & Motwani, 1998) and product quantization (Jegou et al., 2010), have garnered significant interest, graph-based approaches generally achieve the strongest performance on established ANN benchmarks (Aumüller et al., 2018; Simhadri et al., 2022). Introduced in 2016, the Hierarchical Navigable Small World (HNSW) algorithm (Malkov & Yashunin, 2016), emerged as one of the first high-performance graph-based search indexes at scale and still enjoys immense popularity to this day with over 4300 Github stars[1] and deployments in major commercial search systems such as Apache Lucene (Xian et al., 2024) and Pinterest (Pinterest, 2021).

As the name implies, a core feature of the HNSW index is its hierarchically layered graph akin to a skip list (Pugh, 1990) where the search process iteratively traverses through graphs of increasing density before converging to a neighborhood of similar points in the final graph layer. By drawing intuition from skip lists, the HNSW authors argue that the

[1]https://github.com/nmslib/hnswlib

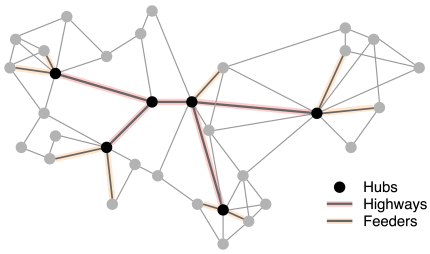


Figure 1. We hypothesize that, in high dimensions, graph-based ANN indexes naturally form a "highway-feeder" structure, where a small subset of nodes and edges are easily reached, well-connected, and heavily traversed.

initial coarse graph layers allow for efficiently identifying the neighborhood of similar points in the collection through fewer overall comparisons.

Despite its popularity, HNSW has notable scalability issues; the hierarchical structure adds significant memory overhead and, as noted in (Malkov & Yashunin, 2016), can reduce throughput in distributed settings compared to flat NSW graphs. Although this overhead is often justified by the latency benefits of graph-based indexes, recent work questions whether the hierarchy is still necessary. (Lin & Zhao, 2019) report that the hierarchy only improves performance for low-dimensional data ($d < 32$), and (Coleman et al., 2022) reach similar conclusions in their ablation study. These observations highlight the need for a more rigorous investigation into whether the hierarchy remains useful in modern, high-dimensional workloads.

Perhaps most importantly, we still have no satisfactory understanding of *why hierarchy does not help*. Hierarchical structures are a mainstay of algorithm design, where a common trick is to reduce an $O(n)$ search process to a sublinear one by traversing a (balanced) hierarchy (Pugh, 1990; Guibas & Sedgewick, 1978; Mikolov et al., 2013; Cormen et al., 2022). Arguably, it is counterintuitive for this idea to fail to hold in the context of high-dimensional similarity search – especially when we have strong positive results that hierarchy *helps* in low dimensions (Beygelzimer et al., 2006; Dolatshah et al., 2015; Ram & Sinha, 2019; Lin & Zhao, 2019). Thus, an exhaustive benchmark and deeper analysis into the necessity of the hierarchy in HNSW would shed further light on the nature of algorithm design in high-dimensional spaces and thus may be of independent interest to the community as well.

1.1. Contributions

In this paper, we study whether the hierarchical component of HNSW is truly necessary. Our central research question is, *"Can we achieve the same performance on large-scale benchmarks with simply a flat navigable small world graph?"* To that end, we organize the paper into two parts:

Benchmarking the hierarchy: We rigorously benchmark HNSW to understand whether the hierarchy is necessary. To do so, we reproduce and extend the hierarchy ablations of previous studies, finding that, on high-dimensional vector datasets, it is indeed beneficial to remove the 'H' from HNSW.

Why does hierarchy not help? We hypothesize that the hierarchy benefits decrease in high-dimensions due to *hubness*. Hubness is a high-dimensional phenomenon that causes a skewed distribution in the near-neighbor lists of search queries (Radovanovic et al., 2010). We hypothesize that hubness leads to preferential attachment in the similarity search graph, inducing the formation of *easily-traversed highways* that connect disparate regions of the graph. This hypothesis, which we call the *Hub Highway Hypothesis*, explains why we no longer need the hierarchy in high dimensions; we can simply traverse the intrinsic highway structure that naturally forms in high-dimensional spaces. Our results ultimately show that hubness is responsible for driving the connectivity of similarity search graphs. This insight opens up exciting new research directions in graph construction, link pruning, and graph traversal.

Our specific contributions are as follows.

- We release an implementation for a flattened version of HNSW, called FlatNav[2], that reaches performance parity with the original hierarchical version with considerable memory savings. To our knowledge, `flatnav` is the only actively maintained, high-performance NSW search library in the open-source ecosystem and thus fills a crucial void in the similarity search community.
- We demonstrate that hierarchy does not improve performance in either the median or tail latency case by building HNSW and FlatNav indexes for 13 popular benchmark datasets ranging in size from 1 million to 100 million vectors.
- We present strong scientific evidence for the hub-highway hypothesis, drawing empirical support from analysis of hubness phenomena in high-dimensional metric spaces and the resulting HNSW graphs.

Practical implications: Our benchmarks reveal that HNSW can be significantly optimized for modern high-dimensional embedding workloads. For instance, as we show in Table 6 in the appendix, our implementation saves roughly 38% and 39% of peak memory consumption during index construction on two Big-ANN benchmark datasets compared to `hnswlib` (and sizable further headroom is likely). Our

[2] https://github.com/BlaiseMuhirwa/flatnav

results confirm the folklore of the similarity search community, conclusively demonstrating that we can remove the hierarchy on high-dimensional inputs with impunity.

2. Related Work

2.1. Near-Neighbor Benchmarks

Recent years have seen the emergence of large-scale benchmarks for the k-NNS problem. ANN Benchmarks (Aumüller et al., 2018) established the first standard evaluation framework, expanding over time to cover 30+ methods across 9 datasets. However, the ANN Benchmark datasets are relatively small, with around one million points. To better reflect real-world scale, Big ANN Benchmarks (Simhadri et al., 2022) introduced five billion-scale datasets. Nevertheless, these benchmarks emphasize overall throughput and omit tail latency metrics like the 99th percentile, which are essential for evaluating hierarchical components such as those in HNSW.

Hierarchy Studies: Many top-performing graph-based ANN algorithms, including HNSW (Malkov & Yashunin, 2016), ONNG (Iwasaki & Miyazaki, 2018), PANNG (Iwasaki, 2016), and HCNNG (Munoz et al., 2019), rely on hierarchical structures. However, recent work has questioned this design. (Dobson et al., 2023) show that HNSW can underperform both HCNNG (with a shallower hierarchy) and DiskANN (Jayaram Subramanya et al., 2019) (which lacks one). (Lin & Zhao, 2019) find hierarchy helpful only in low-dimensional synthetic settings ($d < 32$), but study few real-world datasets and do not explain the observed failure modes. We aim to address these gaps by reproducing prior results (Malkov & Yashunin, 2016; Lin & Zhao, 2019) with our own implementation and extending the analysis to larger datasets with a focus on when and why hierarchy matters.

2.2. Hubness in High Dimensional Spaces

Astute readers might observe that the HNSW graph construction algorithm does not explicitly enforce the small world property and instead adds edges between nodes based on their proximity in the metric space. The connection between proximity and the small world property arises due to *hubness*.

Hubness is a property of high-dimensional metric spaces where a small subset of points (the "hubs") occur a disproportionate number of times in the near-neighbor lists of other points in the dataset (Radovanovic et al., 2010). In other words, a small fraction of nodes are highly connected to other points in the near-neighbor graph. The concentration of distance and measure in high-dimensional spaces provide good intuition for how hubness can arise in a datasets. The concentration of distances is a well-studied phenomenon where the expected ℓ_2 distance between independent and identically distributed (i.i.d) vectors grows with $\sqrt{d}$ while the variance tends to a constant as d approaches infinity (Talagrand, 1994). As a result, the ℓ_2 distance loses its discriminative power as d increases, a fact which also holds for ℓ_p and fractional norms (François et al., 2007). Concentration of measure is a high-dimensional property where random distributions have most of their mass near the boundary of their domain. Taken together, these facts suggest that hubs will form at extrema of high-dimensional datasets - a result which holds true empirically (Low et al., 2013).

Due to undesirable consequences of the hubness phenomenon, such as poor clustering quality, a large body of work has focused on hubness reduction strategies. For instance, (Zelnik-Manor & Perona, 2004) introduced local scaling which scales distances $d(\mathbf{x}, \mathbf{y})$ by accounting for local neighborhood information. Interestingly, our work stands in contrast to this literature on hubness reduction by presenting a case study where hubs provide tangible value in an algorithmic setting, namely in accelerating greedy traversal in near neighbor proximity graphs. This result may be of independent interest to machine learning and algorithms researchers as well.

3. FlatNav Benchmarking Experiments

In this section, we report the results of our benchmarking study comparing the performance of flat HNSW search to hierarchical search on a suite of standard high-dimensional benchmark datasets drawn from real machine learning models. We fix the implementation in our experimental design such that the *same code* is used to construct the indexes. In particular, we use the `hnswlib` library as our baseline HNSW implementation. To benchmark the flat NSW index performance, we extract the bottom layer from `hnswlib` after constructing the full hierarchical graph and reimplement HNSW's greedy search heuristic over the flat graph via `flatnav`.

One natural question that arises from this experimental design is whether the hierarchical component of HNSW might still be useful during *construction* even if the base layer suffices for *search*. We find that there is no difference in performance between these settings. Specifically, we include additional results comparing HNSW to flat graphs constructed from scratch (without the hierarchy) in Appendix E where we see identical results to those reported in this section of the paper. In this section, we focus on extracting the base layer from the `hnswlib` graph to reduce any potential confounding effects from implementation differences. Nevertheless, we obtain identical results regardless of how the base graph is constructed.

3.1. Datasets and Compute

We utilize the benchmark datasets released through the popular leaderboards ANN Benchmarks (Aumüller et al., 2018) (MIT Licensed) and Big ANN Benchmarks (MIT Licensed) (Simhadri et al., 2022). The specific datasets and their associated statistics are presented in Table 4. For the Big ANN Benchmark datasets, we consider both the 10M and 100M collection of vectors, for which the ground truth near neighbors have previously been computed and released. We did not experiment with the largest Big ANN datasets with 1 billion vectors since constructing HNSW indexes at this scale requires over 1.5TB of RAM, which exceeded our compute resources. In this section, we include our benchmarking results for the four 100M-scale datasets available through Big ANN Benchmarks. We see that our flat HNSW implementation achieves performance parity with the hierarchical HNSW implementation.

For our benchmarks on datasets consisting of fewer than 100M vectors in the collection, we use an AWS c6i.8xlarge instance with an Intel Ice Lake processor and 64GB of RAM. We selected this particular public cloud instance to facilitate accessible reproducibility of our experiments. For the 100M-sized large-scale experiments, we use a cloud server equipped with an AMD EPYC 9J14 96-Core Processor and 1 TB of RAM.

3.2. Latency Results

3.2.1. BigANN Benchmarks (Simhadri et al., 2022)

In Figures 2 and 3, we compare latency metrics for HNSW and FlatNav at the 50th and 99th percentile for the four 100M datasets from BigANN benchmarks listed in Table 4. All of our results support the conclusion that `flatnav` achieves nearly identical performance to `hnswlib`.

From the results in Figures 2 and 3, we observe that there is no consistent and discernable gap between FlatNav and HNSW in both the median and tail latency cases. These results suggest that the hierarchical structure of HNSW provides no tangible benefit on practical high-dimensional embedding datasets.

3.2.2. ANN Benchmarks (Aumüller et al., 2018)

We repeat the same experimental setup comparing HNSW and FlatNav on the ANN Benchmark datasets listed in Table 4. In Figures 4 and 5, we report the p50 and p99 latency of all of the non-GloVe ANN Benchmarks. Although these datasets are smaller in scale than the BigANN Benchmarks, we still see no discernible difference in latency between HNSW and FlatNav which supports our hypothesis that the vector dimensionality and not the size of the collection is the main driver of eliminating the need for hierarchical

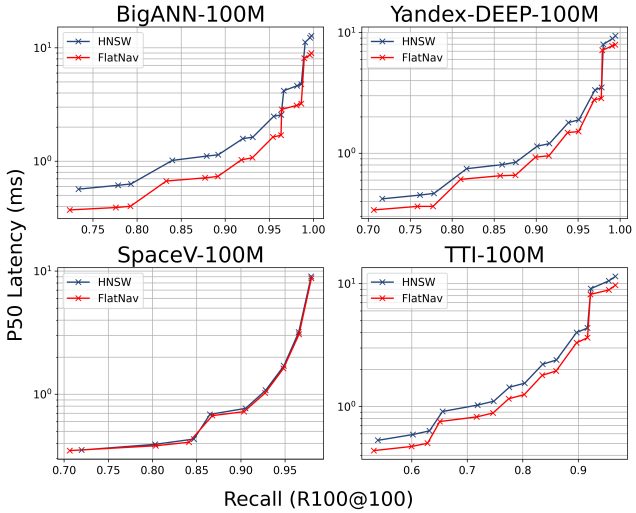


Figure 2. p50 Latency vs. Recall. FlatNav performs nearly identically to HNSW.

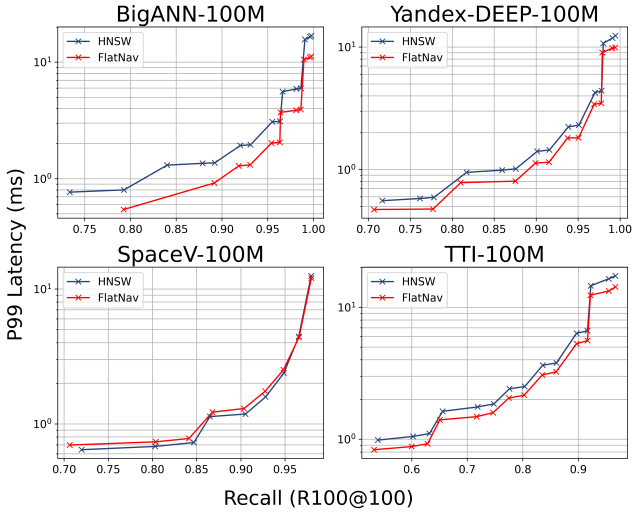


Figure 3. p99 Latency vs. Recall. FlatNav performs nearly identically to HNSW.

search in small world graphs. We see further evidence of this idea in Figure 6, which confirms that there is no clear performance benefit provided by the hierarchy of HNSW.

4. The Hub Highway Hypothesis

We now turn our attention to studying why the hierarchy appears to provide no benefit in the search process. In our experiments, we observed that a small fraction of nodes appear in the set of near neighbors for a disproportionate number of other vectors. We thus conjecture that the hub structure prevalent in high-dimensional data performs the same functional role as the hierarchy.

HYPOTHESIS (Hub Highways). *In high-dimensional metric spaces, k-NN proximity graphs form a highway routing structure where a small subset of nodes are well-connected and heavily traversed, particularly in the early stages of graph search.*

We remark that the existence of hub nodes in high-dimensional space is not a new observation (Radovanovic et al., 2010). The novelty of our hypothesis lies in connecting the idea of hubness to the notion of accelerating near neighbor search in ANN proximity graphs. In particular, we conjecture that near neighbor queries over proximity graphs in high dimensions often spend the majority of their time visiting hub nodes early on in the search process before converging to a local neighborhood of near neighbors. This procedure succeeds because hub nodes are very well connected to other parts of the graph and thereby efficiently route queries to the appropriate neighborhood in much the same manner that the layered hierarchy purports to do.

In the remainder of this section, we present an experimental design and a series of results that provide empirical evidence in the affirmative for the existence of such a highway routing mechanism amongst hubs in navigable small world graphs.

4.1. Methodology

Argument sketch: We will demonstrate the *Hub Highway Hypothesis* by providing empirical evidence for the following claims.

1. Some nodes are visited by queries much more frequently than others. The relative popularity of these *hub nodes* is explained by the hubness phenomenon that arises in high dimensions.
2. The hub nodes form a well-connected subgraph of hubs (the *highway* network).
3. Queries visit many hub nodes early in the search process, before visiting less well-traversed neighborhoods.

Empirical measures of hubness: To support the first

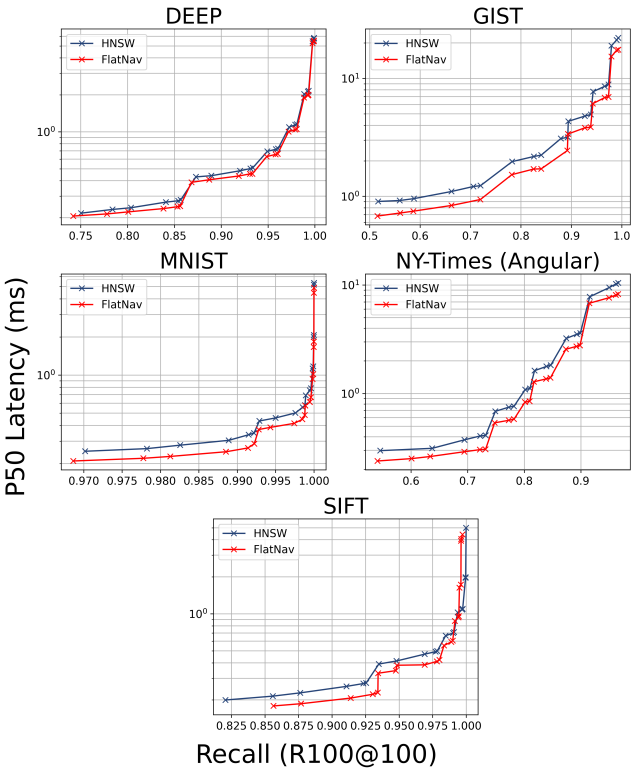


Figure 4. p50 Latency vs. Recall. FlatNav performs nearly identically to HNSW.

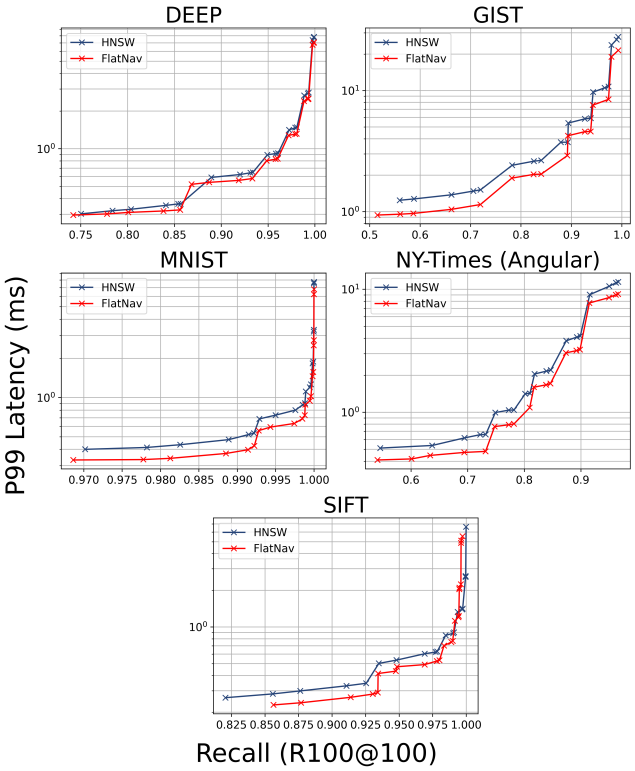


Figure 5. p99 Latency vs. Recall. FlatNav performs nearly identically to HNSW.

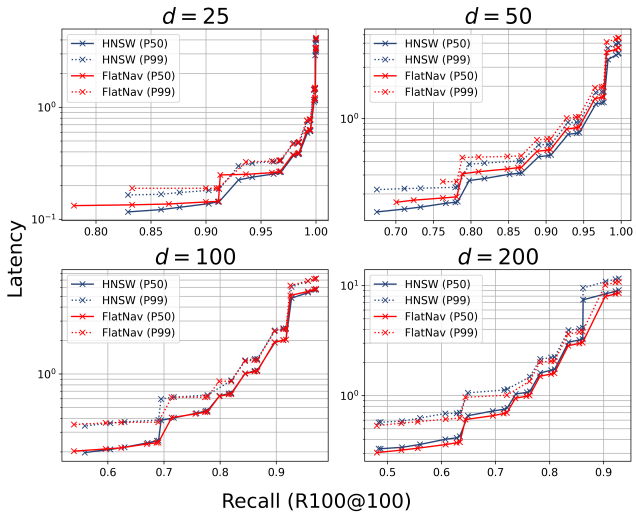


Figure 6. p50 and p99 Latency vs. Recall for HNSW and FlatNav over GloVe datasets.

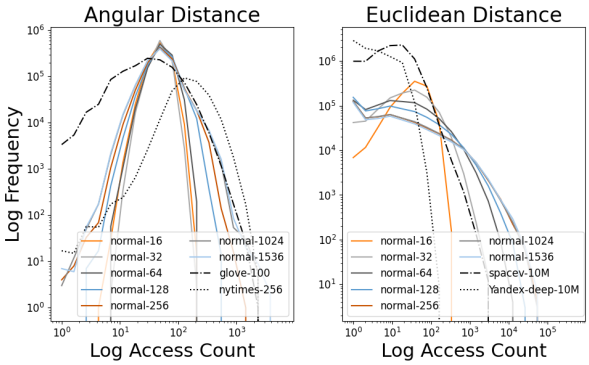


Figure 7. Log-normalized Node access count distribution $P_m(\mathbf{x}_i)$ for datasets using angular (left) and l_2 (right) distances.

part of our argument, we require a formal characterization of hubness. Following (Radovanovic et al., 2010), let $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n$ be vectors drawn from the same probability distribution supported on $\mathcal{S} \subseteq \mathbb{R}^d$, and let $\phi : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be a distance function. For $1 \leq i, k \leq n$, define $p_{i,k}(\mathbf{x}) = 1$ if $\mathbf{x}$ is among the k-NN set of $\mathbf{x}_i$ under ϕ, and 0 otherwise. Define $N(\mathbf{x}) := \sum_{i=1}^{n} p_{i,k}(\mathbf{x})$, the number of vectors $\mathbf{x}_i$ for which $\mathbf{x}$ appears in their k-NN set.

Given a dataset $\mathcal{D}$, the values $\{N(\mathbf{x})\}_{\mathbf{x}\in\mathcal{D}}$ form a distribution N_k, whose skewness is given by $S_{N_k} = \mathbb{E}[(N_k - \mu_{N_k})^3] \,/\, \sigma^3_{N_k}$. We use S_{N_k} to measure the hubness of a dataset.

This measure characterizes the asymmetry of the k-occurrence distribution N_k, and it is the metric most often used to estimate the presence of hubs. The more skewed the distribution of N_k, the greater the chance that a small number of vectors (hubs) will occur in the k-NN sets of other vectors.

Synthetic and ANN Benchmark Datasets: We use real and synthetic datasets as shown in Table 5 to study the Hub-Highway Hypothesis. In addition to a subset of ANN Benchmark datasets, we generate synthetic datasets by drawing vectors from the standard normal distribution.

4.2. Skewness of the Node Access Distribution

The goal of this study is to support our claim that some nodes are visited far more frequently, and that this process is driven by hubs in the metric space. We examine the discrete distribution of the number of times each node in the index is visited during search given a fixed number of queries, which we write as $P_m(\mathbf{x}_i)$ (i.e., node $\mathbf{x}_i$ is visited m times). If $P_m(\mathbf{x}_i)$ is right-skewed, it means that some nodes are very frequently visited.

Table 1. Similarity search index parameters

m	ef-construction	ef-search	k
32	100	200	100

Figure 7 shows the log-normalized node access count distribution for different datasets. We observe that as the dimension d increases, this distribution becomes right-skewed for ℓ_2 distance-based datasets. While this is strong evidence for the first part of our claim, it leaves open the possibility that some mechanism other than hubness is driving our observations. To control for this possibility, we also study the cosine distance, which is known to have anti-hub properties that prevent hub formation (Radovanovic et al., 2010). We find that the cosine distance does not have a dramatic skew, even for $d \in \{1024, 1536\}$. The increased skewness of the $P_m(\mathbf{x}_i)$ distribution as d increases demonstrates that highway nodes become increasingly prevalent as the dimension increases, and the differences between the ℓ_2 and cosine results suggest that the hubness phenomenon is responsible for the formation of the highway nodes.

4.3. Subgraph Connectivity of the Hub-Highway Nodes

In this section we present empirical evidence confirming that hub-highway nodes exhibit strong connectivity in the graph. We begin by explaining our procedure to identify hubs. Let $\{\mathbf{x}_i\}_{i=1}^n$ be the vectors in a similarity search index for a dataset $\mathcal{D}$.

- We identify hub nodes as those that fall into the top percentile of the empirical node access distribution $P_m(\mathbf{x}_i)$. We use 95th and 99th percentile of node access counts as our threshold. We assign a binary label to each node to indicate whether it is a hub. Let $h : \mathcal{D} \to \{0, 1\}$ be this assignment function with $h(\mathbf{x}_i) = 1$ for

nodes identified as hubs.

- We wish to estimate the likelihood that a randomly-chosen out-neighbor of a hub node is, itself, a hub. To do so, we examine the 1-hop out-degree expansion of each hub and count the number of adjacent hub nodes. This yields a discrete distribution for the number of hubs to which each hub node is connected.

- Similarly, we select a set of random non-hub nodes from $V := \mathcal{D} \backslash \bigcup_{i=1}^{n} h(\mathbf{x}_i) = 1$. For each node $\mathbf{x}_i \in V$, we compute the same quantity to find the number of hubs with which non-hubs are connected, allowing us to construct the equivalent distribution for non-hubs.

We test whether hubs differ from non-hubs in connectivity behavior by using statistical tests on the two distributions. Under the null hypothesis, both groups are equally likely to connect to other hubs; the alternative asserts that hubs preferentially attach to hubs.

We apply both a two-sample t-test and the Mann-Whitney U-test (Mann & Whitney, 1947), the latter of which is more suitable for ANN datasets since it makes no normality assumption. With dataset sizes $n > 10^3$, Table 2 reports results using the top 5% of nodes as hubs. At a 0.05 significance threshold, we reject the null in all but five cases using the U-test (and all but six with the t-test). Effect sizes are largest in synthetic ℓ_2 datasets, likely due to their stronger hubness under this metric.

Using a stricter top-1% threshold (Table 3), we reject the null in all but one case for both tests, with substantially larger effect sizes. These findings indicate that the most prominent hubs tend to form tightly connected subgraphs, consistent with our Hub Highway Hypothesis.

Table 2. Two-sample t-test and Mann-Whitney U-test results. Hub nodes are selected using the P95 threshold of the node access distribution.

Dataset	Dim	Mann-Whitney	Two-Sample t-Test	Effect Size
IID Normal (Angular)	16	0.3629	0.3090	0.0267
IID Normal (L2)	16	$<10^{-5}$	$<10^{-5}$	0.3737
IID Normal (Angular)	32	0.0335	0.0516	0.0872
IID Normal (L2)	32	$<10^{-5}$	$<10^{-5}$	0.4275
IID Normal (Angular)	64	0.0216	0.0148	0.1165
IID Normal (L2)	64	$<10^{-5}$	$<10^{-5}$	0.3965
IID Normal (Angular)	128	0.0083	0.0083	0.1284
IID Normal (L2)	128	$<10^{-5}$	$<10^{-5}$	0.3773
IID Normal (Angular)	256	0.0009	0.0007	0.1723
IID Normal (L2)	256	$<10^{-5}$	$<10^{-5}$	0.2620
IID Normal (Angular)	1024	0.1000	0.1114	0.0652
IID Normal (L2)	1024	$<10^{-5}$	$<10^{-5}$	0.2361
IID Normal (Angular)	1536	0.0957	0.1141	0.0645
IID Normal (L2)	1536	$<10^{-5}$	$<10^{-5}$	0.2512
GloVe	100	$<10^{-5}$	$<10^{-5}$	0.2550
NYTimes	256	$<10^{-5}$	$<10^{-5}$	0.4488
GIST	960	$<10^{-5}$	$<10^{-5}$	0.3645
Yandex-DEEP	96	0.5002	0.5000	0.0000
Microsoft-SpaceV	100	0.1586	0.1585	0.0535

Table 3. Two-sample t-test and Mann-Whitney U-test results. Hub nodes are selected using the P99 threshold of the node access distribution.

Dataset	Dim	Mann-Whitney	Two-Sample t-Test	Effect Size
IID Normal (Angular)	16	0.0006	0.0006	0.1745
IID Normal (L2)	16	$<10^{-5}$	$<10^{-5}$	0.6621
IID Normal (Angular)	32	0.0347	0.0347	0.0972
IID Normal (L2)	32	$<10^{-5}$	$<10^{-5}$	0.8173
IID Normal (Angular)	64	0.0359	0.0417	0.0927
IID Normal (L2)	64	$<10^{-5}$	$<10^{-5}$	0.8725
IID Normal (Angular)	128	0.0093	0.0070	0.1316
IID Normal (L2)	128	$<10^{-5}$	$<10^{-5}$	0.8428
IID Normal (Angular)	256	$<10^{-5}$	$<10^{-5}$	0.3110
IID Normal (L2)	256	$<10^{-5}$	$<10^{-5}$	0.8582
IID Normal (Angular)	1024	0.1472	0.1318	0.0598
IID Normal (L2)	1024	$<10^{-5}$	$<10^{-5}$	0.8314
IID Normal (Angular)	1536	$<10^{-5}$	$<10^{-5}$	0.2356
IID Normal (L2)	1536	$<10^{-5}$	$<10^{-5}$	0.8568
GloVe	100	$<10^{-5}$	$<10^{-5}$	0.7642
NYTimes	256	$<10^{-5}$	$<10^{-5}$	0.9305
GIST	960	$<10^{-5}$	$<10^{-5}$	0.6829
Yandex-DEEP	96	0.0013	0.0013	0.1614
Microsoft-SpaceV	100	0.0011	0.0011	0.1644

4.4. Hub-Highway Nodes Enable Fast Traversal

Our final question is whether the highway nodes allow queries to quickly traverse the similarity search graph. While it is not surprising that a well-connected subgraph of frequently visited nodes would enable this behavior, it is not necessarily the case that queries would use the highway in the way predicted by our hypothesis, namely to quickly identify a neighborhood for deep exploration. To investigate this question, we track the sequence of nodes visited during beam search for several thousand queries. This allows us to determine the fraction of time spent within hub nodes in different phases of search.

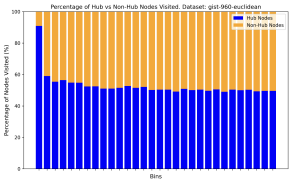
Since beam search takes a variable number of steps for each query, we normalize by the total search length when presenting the results. More formally, suppose that $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l$ is a length-l sequence of such nodes visited by a query. We use the hub node assignment heuristic discussed in section 4.3 to label $h(\mathbf{x}_i)$ each of these nodes as hubs / non-hubs. We then split the sequence into bins and compute the prevalence of hubs in each bin. For bin B_i, this is given by

$$\left(\frac{1}{|B_i|}\right) \sum_{\mathbf{x}_j \in B_i} h(\mathbf{x}_j)$$

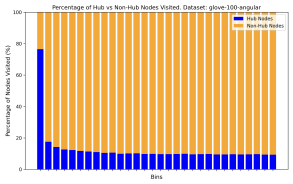
where $|B_i|$ is the bin size (fixed to 30 in our analysis). By averaging this value over all queries, we can plot the likelihood of visiting a hub as the search progresses.

Figure 8 shows results for the Gist, GloVe, Microsoft SpaceV and Yandex-DEEP benchmark datasets. We observe that queries tend to concentrate in the highway structures early in search, shown by the high percentage of hub nodes visited in the first 5-10% of the search steps. This result suggests that the highway allows queries to quickly navigate the similarity search graph until they find the region of the graph best suited for deep exploration. The propensity of the query to visit hubs appears to be tied to the hubness properties of the dataset. For example, the GloVe dataset uses

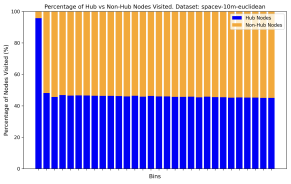
the angular distance, has less pronounced hubs (Figure 7), and queries spend a lower percentage of their time in hub nodes in this dataset (Figure 8(b)). On the other hand, the GIST dataset has some of the highest highway utilization rates and is also our highest-dimensional ℓ_2 dataset.



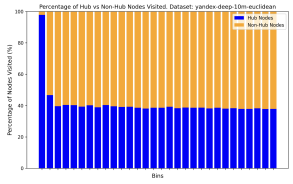
((a)) Gist



((b)) GloVe



((c)) Microsoft SpaceV



((d)) Yandex Deep

Figure 8. Highway nodes allow queries to traverse the graph faster.

4.5. Discussion

Our sequence of experiments provide substantial evidence supporting the *Hub Highway Hypothesis*. Although it has long been established that the hubness phenomenon negatively affect common applications, such as clustering and even near neighbor search recall, existing ANNS methods, such as HNSW and NSG (Zhao et al., 2023) mostly emphasize algorithmic improvements and performance optimizations only. We have shown that leveraging the inherent structures in the data, particularly hub-highway occurrences, should be central to the design of new scalable similarity search indexes.

Scientific Implications: Our work reveals that the hub highway is an intrinsic and naturally-forming structure in high-dimensional proximity graphs. We believe this observation is both novel and has important implications for the scaling potential of many traversal heuristics. Specifically, we expect the benefits of sophisticated search initialization methods to decay under hub-style preferential attachment.

Initialization is a recurring and popular research direction for graph-based near-neighbor search, and leading algorithms vary greatly in their initialization techniques. The research question dates back to the seminal 1993 paper by Arya and Mount (Arya & Mount, 1993), which conjectured that clever search initialization – in their case, via kd-tree – could improve performance over random initialization. Over the following three decades, the research community has investigated diverse stratified sampling based on clusters (Sebastian & Kimia, 2002), vantage-point trees (Iwasaki, 2010), seeds formed from the graph expansion of kd-trees (Iwasaki & Miyazaki, 2018) and previously-visited nodes (Wang & Li, 2012), hierarchical graphs (Malkov & Yashunin, 2016), hierarchical clustering (Munoz et al., 2019), dataset medoids (Jayaram Subramanya et al., 2019), and several other methods before finally returning to cluster-stratified candidates (Oguri & Matsui, 2024; Ni et al., 2023) and random entry nodes (Jaiswal et al., 2022).

How is it that these early papers on graph search show performance gains, even as today's best vector databases return to simple initializations without performance loss? Our hub highway hypothesis offers a clear explanation for this apparent contradiction: In the early 2000s and 2010s, datasets were low-dimensional and initialization was important to avoid local minima and long graph detours. However, modern vector databases contain data that is sufficiently high-dimensional to naturally form a fast-routing structure, explaining why initialization no longer drives performance. Based on our results, we conjecture that the largest algorithmic improvements to graph-based ANNS should come from optimizations that affect the connectivity and cost of traversal in the base graph, such as link pruning and search algorithm design.

5. Conclusion

Approximate near neighbor search has become an increasingly crucial computational workload in recent years with the seminal Hierarchical Navigable Small World (HNSW) algorithm continuing to garner significant interest and adoption from practitioners. We present the first comprehensive study on the utility of the hierarchical component of HNSW over numerous large-scale datasets and performance metrics. Ultimately, we find that the hierarchy of HNSW provides no clear benefit on high-dimensional datasets and can be **removed** without any discernible loss in performance while providing memory savings.

While similar observations have been made before in the literature (Lin & Zhao, 2019; Coleman et al., 2022; Dobson et al., 2023), we are, to our knowledge, the first to conduct an exhaustive study over modern benchmark datasets and taking extensive care to compare implementations with performance engineering parity. Furthermore, we go beyond prior works and study *why* the hierarchy does not help, culminating in our introduction of the *Hub Highway Hypothesis*, an empirical result on how proximity graphs built over high-dimensional metric spaces leverage a small subset of well-connected nodes to traverse the network quickly. We believe that our results provide immediate implications for practitioners seeking to save memory or simplify their vector database implementations and we look forward to further partnering with the community on these endeavors.

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under Grant No. 2313998. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation.

References

Aguerrebere, C., Bhati, I., Hildebrand, M., Tepper, M., and Willke, T. Similarity search in the blink of an eye with compressed indices. *Proceedings of the VLDB Endowment*, 16:3433–3446, 08 2023. doi: 10.14778/3611479.3611537.

Arya, S. and Mount, D. M. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, volume 93, pp. 271–280. Citeseer, 1993.

Aumüller, M., Bernhardsson, E., and Faithfull, A. J. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.*, 87, 2018. URL https://api.semanticscholar.org/CorpusID:36089988.

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18 (9):509–517, 1975.

Beygelzimer, A., Kakade, S., and Langford, J. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pp. 97–104, 2006.

Boytsov, L. and Naidan, B. Engineering efficient and effective non-metric space library. In *Similarity Search and Applications: 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings 6*, pp. 280–293. Springer, 2013.

Campos, D. F., Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L., and Mitra, B. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268, 2016. URL https://api.semanticscholar.org/CorpusID:1289517.

Coleman, B., Segarra, S., Smola, A. J., and Shrivastava, A. Graph reordering for cache-efficient near neighbor search. *Advances in Neural Information Processing Systems*, 35: 38488–38500, 2022.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022.

Dobson, M., Shen, Z., Blelloch, G. E., Dhulipala, L., Gu, Y., Simhadri, H. V., and Sun, Y. Scaling graph-based anns algorithms to billion-size datasets: A comparative analysis. *arXiv preprint arXiv:2305.04359*, 2023.

Dolatshah, M., Hadian, A., and Minaei-Bidgoli, B. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. *arXiv preprint arXiv:1511.00628*, 2015.

Feng, C., Li, W., Lian, D., Liu, Z., and Chen, E. Recommender forest for efficient retrieval. *Advances in Neural Information Processing Systems*, 35:38912–38924, 2022.

François, D., Wertz, V., and Verleysen, M. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19:873–886, 2007. URL https://api.semanticscholar.org/CorpusID:13220558.

Guibas, L. J. and Sedgewick, R. A dichromatic framework for balanced trees. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pp. 8–21. IEEE, 1978.

Guo, R., Sun, P., Lindgren, E. M., Geng, Q., Simcha, D., Chern, F., and Kumar, S. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:218614141.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

Iwasaki, M. Proximity search in metric spaces using approximate k nearest neighbor graph. *IPSJ Trans. Database*, 3 (1):18–28, 2010.

Iwasaki, M. Pruned bi-directed k-nearest neighbor graph for proximity search. In *International Conference on Similarity Search and Applications*, pp. 20–33. Springer, 2016.

Iwasaki, M. and Miyazaki, D. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. *arXiv preprint arXiv:1810.07355*, 2018.

Jaiswal, S., Krishnaswamy, R., Garg, A., Simhadri, H. V., and Agrawal, S. Ood-diskann: Efficient and scalable graph anns for out-of-distribution queries. *arXiv preprint arXiv:2211.12850*, 2022.

Jayaram Subramanya, S., Devvrit, F., Simhadri, H. V., Krishnawamy, R., and Kadekodi, R. Diskann: Fast accurate

billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32, 2019.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7:535–547, 2017. URL https://api.semanticscholar.org/CorpusID:926364.

Kleinberg, J. M. Navigation in a small world. *Nature*, 406 (6798):845–845, 2000.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401, 2020. URL https://api.semanticscholar.org/CorpusID:218869575.

Lin, P.-C. and Zhao, W.-L. Graph based nearest neighbor search: Promises and failures. *arXiv preprint arXiv:1904.02077*, 2019.

Low, T., Borgelt, C., Stober, S., and Nürnberger, A. The hubness phenomenon: Fact or artifact? *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*, pp. 267–278, 2013.

Malkov, Y. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:824–836, 2016. URL https://api.semanticscholar.org/CorpusID:8915893.

Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. In *Similarity Search and Applications: 5th International Conference, SISAP 2012, Toronto, ON, Canada, August 9-10, 2012. Proceedings 5*, pp. 132–147. Springer, 2012.

Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45: 61–68, 2014.

Mann, H. B. and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pp. 50–60, 1947.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Munoz, J. V., Gonçalves, M. A., Dias, Z., and Torres, R. d. S. Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. *Pattern Recognition*, 96:106970, 2019.

Ni, J., Xu, X., Wang, Y., Li, C., Yao, J., Xiao, S., and Zhang, X. Diskann++: Efficient page-based search over isomorphic mapped graph index using query-sensitivity entry vertex. *arXiv preprint arXiv:2310.00402*, 2023.

Oguri, Y. and Matsui, Y. Theoretical and empirical analysis of adaptive entry point selection for graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2402.04713*, 2024.

Pinterest. Manas hnsw realtime: Powering realtime embedding-based retrieval, Jan 2021. URL https://medium.com/pinterest-engineering/manas-hnsw-realtime-powering-realtime-embedding-based-retrieval-dc71dfd6afdd.

Pugh, W. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

Radovanovic, M., Nanopoulos, A., and Ivanovic, M. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(sept): 2487–2531, 2010.

Ram, P. and Sinha, K. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*, pp. 1378–1388, 2019.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://api.semanticscholar.org/CorpusID:201646309.

Sebastian, T. B. and Kimia, B. B. Metric-based shape retrieval in large databases. In *2002 International Conference on Pattern Recognition*, volume 3, pp. 291–296. IEEE, 2002.

Simhadri, H., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S., and Wang, J. Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search, 05 2022.

Talagrand, M. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 81:73–205, 1994. URL https://api.semanticscholar.org/CorpusID:119668709.

Travers, J. and Milgram, S. An experimental study of the small world problem. In *Social networks*, pp. 179–197. Elsevier, 1977.

Wang, J. and Li, S. Query-driven iterated neighborhood graph search for large scale indexing. In *Proceedings of the 20th ACM international conference on Multimedia*, pp. 179–188, 2012.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

Xian, J., Teofili, T., Pradeep, R., and Lin, J. Vector search with openai embeddings: Lucene is all you need. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 1090–1093, 2024.

Zelnik-Manor, L. and Perona, P. Self-tuning spectral clustering. In *Neural Information Processing Systems*, 2004. URL https://api.semanticscholar.org/CorpusID:17066951.

Zhao, J., Both, J. P., Rodriguez-R, L. M., and Konstantinidis, K. T. Gsearch: ultra-fast and scalable genome search by combining k-mer hashing with hierarchical navigable small world graphs. *Nucleic Acids Research*, 52(16): e74–e74, 2024.

Zhao, X., Tian, Y., Huang, K., Zheng, B., and Zhou, X. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. *Proc. VLDB Endow.*, 16:1979–1991, 2023. URL https://api.semanticscholar.org/CorpusID:258718568.

Appendix

A. Background: Similarity Search & HNSW

A.1. Similarity Search

In the similarity search (or k-NNS) problem, we are interested in retrieving k elements from a dataset $\mathcal{D} = \{\mathbf{x}_i, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ that minimize the distance to a given query $q \in \mathbb{R}^d$ (or, equivalently, maximize the vector similarity). More precisely, given a similarity function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the nearest neighbor $\mathbf{x}^* \in \mathcal{X}$ of q is defined as

$$\mathbf{x}^* \coloneqq \arg\max_{\mathbf{x}_i \in \mathcal{D}} \phi(\mathbf{x}_i, q)$$

where ϕ is usually the ℓ_2 or cosine similarity. With the enormity of modern data workloads and the underlying vector dimensionality, it becomes computationally infeasible to exhaustively search for the true top-k neighbors for any query q. Thus, approximate search algorithms trade-off quality of the search for lower latency.

In the approximate nearest neighbor search (ANNS) regime, we evaluate the quality of the search procedure typically by the Recall@k metric. More formally, suppose a given ANNS search algorithm outputs a subset $\mathcal{O} \subseteq \mathcal{D}, |\mathcal{O}| = k$, and let $G \subseteq \mathcal{D}$ be the true k nearest neighbors of a query q. We define Recall@k by $\frac{|\mathcal{O} \cap G|}{k}$. ANNS algorithms seek to maximize this metric while retrieving results as quickly as possible.

A.2. HNSW Overview

With this formalization of the ANNS problem, we will now briefly review the key elements of the HNSW algorithm, which is the central focus of our benchmarking study. As we alluded to previously, HNSW builds off of prior work in *navigable small world graph* indexes introduced in (Malkov et al., 2014). Small world graphs are a well-studied phenomenon in both computing and the social sciences and are primarily defined by the fact that the average length of a shortest path between two vertices is small (typically scaling logarithmically with the number of nodes in the network) (Travers & Milgram, 1977; Watts & Strogatz, 1998; Kleinberg, 2000). Small world graphs are also often characterized by the presence of well-connected *hub nodes* which we discuss further in the next section.

While small world graphs are, by construction, suited for efficient greedy graph traversal, the HNSW authors argue that the polylogarithmic scaling of the search process is still too inefficient for the demands of near neighbor search on large datasets. This claim motivates the design of HNSW where the hierarchy allows for computing a fixed number of distances in each graph layer independent of the network size.

Specifically, the HNSW index is constructed in an iterative fashion. For a newly inserted element x, the algorithm will randomly select a maximum layer l and then insert the new point into every layer up to l. This randomized process is executed with an exponentially decaying probability distribution such that, in expectation, each subsequent layer has exponentially more nodes than its predecessor. Within a layer, HNSW greedily adds edges between x and its M closest neighbors (where M is a hyperparameter) where the neighbors consist of previously inserted points. This process then repeats in the subsequent layer below using the closest neighbors found in the prior graph as entry points. Through this process, the top layer of the hierarchy will be the coarsest directed graph, consisting of the fewest nodes and edges, and the bottom layer will be the densest and contain all of the nodes, each with connections to (up to) M neighbors. As an additional, and important, optimization, HNSW also implements the pruning heuristic of (Arya & Mount, 1993) that will prune an edge from u to v if there exists another edge from u to a neighbor w of v such that the distance from u to w is less than that of u to v.

The search procedure of HNSW, described in Algorithm 2 also executes iteratively where the algorithm maintains a list of candidate points at each layer of the hierarchical graph before returning the final list k nearest neighbor candidates after traversing the base graph layer.

B. Reproduction of Prior Studies

In this section, we present a replicability study using four `flatnav` NSW implementation. In particular, we revisit the experimental design of two prior works in the literature: the original 2016 HNSW paper of (Malkov & Yashunin, 2016) and a subsequent 2019 paper from (Lin & Zhao, 2019) that found limitations with the hierarchical component of HNSW. As we discussed in the previous section, these prior works possess limitations in experimental design, scope of benchmarking datasets, and a lack of analysis into understanding the results, which motivates our work in this paper. Nevertheless, we use these prior studies as a starting point to see if we can independently replicate these results via our own FlatNav implementation. Such a reproduction would both further validate the soundness of these previous experiments over the test of time as well as provide confirmation of the correctness of FlatNav before we proceed to new, larger-scale benchmarks.

Following the same setups as (Malkov & Yashunin, 2016) and (Lin & Zhao, 2019) we generate a series of random vector datasets of varying dimensionality where each vector component is sampled uniformly at random from the range $[0, 1)$. In particular, we consider dimensionalities of $d = 4, 8, 16$ and 32. As in (Lin & Zhao, 2019), we set the number of near neighbors to retrieve to $k = 1$ (departing

Algorithm 1 HNSW Construction

```
Input: Set of data points D, max layer L_max, max connections per layer M, layer insertion probability m_l, size of dynamic candidate list efc
Output: HNSW graph with hierarchical layers
procedure CONSTRUCT(D, L_max, M, m_l, efc)
    Initialize empty hierarchical graph G
    Initialize entry point ep ← None
    for each p ∈ D do
        L_p ← GeometricDistribution(m_l)
        if ep = None then
            Set p as entry point ep
            Insert p into all levels ≤ L_p
        end if
        for l = L_max to L_p do
            ep ← SearchLayer(G, l, p, ep, efc)        ▷ Algorithm 2
        end for
        for l = 0 to L_p do
            N ← SelectNeighbors(p, G, l, M)
            Add edges from q to each neighbor n ∈ N at layer l
            if n ∈ N has < M edges then
                Add back-connections to q to node n.
            else
                Run SelectNeighbors on {q, edges of n}.
            end if
        end for
    end for
    if L_p > L_ep then
        ep ← p
    end if
end procedure

function SELECTNEIGHBORS(p, G, l, M)
    Compute distances from p to all nodes in G[l]
    Return M nodes based on selection heuristic in (Arya & Mount, 1993)
end function
```

Algorithm 2 HNSW Query

```
Input: Graph G, layer l, query q, starting point p, number of nearest neighbors to return efs
procedure SEARCHLAYER(G, l, q, p, efs)
    Candidate queue C = p, currently top results queue T = p, visited list V = p
    while C is not empty do
        c ← nearest element from C to q
        f ← furthest element from T to q
        if dist(c, q) ¿ dist(f, q) then return T
        end if
        for e ∈ neighbourhood(c) at layer l do
            if e ∈ V then
                continue
            end if
            V.add(e)
            if dist(e, q) ≤ dist(f, q) or |T| ≤ efs then
                C.add(e)
                T.add(e)
            end if
            if |T| ≥ efs then
                Remove furthest point to q from T
            end if
            f ← furthest element from T to q
        end for
        return T
    end while
end procedure
```

from the default of $k = 100$ we use elsewhere in this paper). We also tried including the `sw-graph` NSW baseline (Boytsov & Naidan, 2013) that (Malkov & Yashunin, 2016) used in their evaluation to benchmark against HNSW, but we were not able to run this older library successfully. However, we were able to replicate these prior findings using our own `flatnav` implementation which is conceptually identical to `sw-graph` but with more software optimizations to achieve engineering parity with `hnswlib`.

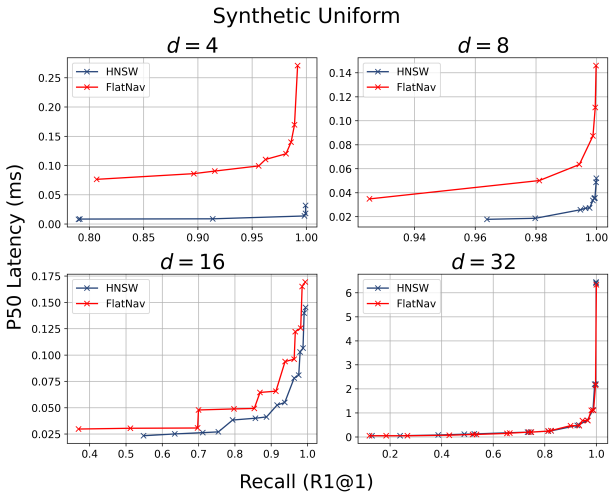


Figure 9. Median Latency vs. Recall of HNSW and FlatNav across dimensions $d = 4, 8, 16, 32$. We observe that the hierarchical structure accelerates search only when $d < 32$, matching the findings of (Lin & Zhao, 2019). Our results demonstrating a significant advantage with HNSW on synthetic datasets with dimensionality $d = 4$ and $d = 8$ also match the findings of the original HNSW paper (Malkov & Yashunin, 2016)

.

As shown in Figure 9, we successfully replicated the experiments benchmarking HNSW versus a flat NSW graph from two prior research papers. Notably, both of these previous works primarily experiment with randomly generated vector data with very low dimensionality by the standards of modern machine learning. Coupled with our findings in the next section where we find no discernible difference between HSNW and a flat graph index on high-dimensional datasets, our results suggest a simple decision criterion for selecting a search index: **For dimensionality $d < 32$, HNSW and the hierarchy provide a speedup. Otherwise, the simplicity and memory savings of a flat NSW index provide more benefit.**

We also had the opportunity to discuss our findings with the lead author of (Malkov & Yashunin, 2016) who confirmed that the hierarchy provides a robust speedup on these low-dimensional datasets but noted the performance on higher dimensional vectors remained less clear, which further motivated us to take up the benchmarking study in the next section.

C. Dataset Statistics

C.1. Benchmark Datasets for the Latency-Recall Tradeoff

Table 4 shows the different benchmark datasets used in Section 3.1.

Dataset	Dimensionality	# Points	# Queries
BigANN†	128	100M	10K
Microsoft SpaceV†	100	100M	29.3K
Yandex DEEP†	96	100M	10K
Yandex Text-to-Image†	200	100M	100K
GloVe	{25, 50, 100, 200}	1.2M	10K
NYTimes	256	290K	10K
GIST	960	1M	1K
SIFT	128	1M	10K
MNIST	784	60K	10K
DEEP1B	96	10M	10K

Table 4. Dataset Statistics. The datasets marked by † are from the BigANN benchmarks (Simhadri et al., 2022). The remaining are taken from ANN Benchmarks (Aumüller et al., 2018).

C.2. Benchmark Datasets for the Hub-Highway Hypothesis Experiments

Table 5 details the various ANN and Big-ANN benchmark datasets as well as the synthetic datasets used in Section 4 for illustrating the empirical evidence of the hub-highway hypothesis.

Table 5. Hub-Highway Experimental Datasets

Dataset	Dimensionality	# Points	# Queries
GIST	960	1M	1k
GloVe	100	1.2M	10k
NYTimes	256	290K	10k
Yandex-DEEP	96	10M	10k
Microsoft-SpaceV	100	10M	29.3k
IID Normal	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^{10}, 1.5 \cdot 2^{10}\}$	1M	10k
IID Normal	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^{10}, 1.5 \cdot 2^{10}\}$	1M	10k

D. Extended Discussion and Limitations

D.1. Extended Discussion

Small-World Graphs: The network science research community has known for decades that long-range connections and hubs induce the formation of "small-world" graphs that are easily traversed (Watts & Strogatz, 1998). This idea has been enormously influential in ANNS, providing the motivation for both NSW and HNSW (Malkov et al., 2012; 2014), but our results suggest that ANN graphs constructed over low-dimensional datasets may not in fact exhibit small-world properties. Because the k-occurrence distribution is near-uniform for intrinsically low-dimensional data distributions, a pure kNN graph (without pruning or long-range

links) will not create hubs with a high in-degree. We believe that hierarchical structures are helpful in low dimensions because they help to induce hub behavior, by ensuring that search always begins from a small set of nodes. However, this is not necessary to produce hubs and induce small-world properties in high dimensions. The kNN graph construction process is sufficient on its own, because the hub highway emerges in high dimensions.

D.2. Limitations

Despite our notable empirical evidence for the hug-highway hypothesis and its significance in understanding graph-based ANN search, a principled understanding of this phenomenon from theoretical grounds is still lacking. Future research efforts can be directed towards explicit bounds on the probability of a query q reaching high latency on a proximity graph $G = (V, E)$ consisting of hub nodes H.

There are several reasons why such a theoretical formulation is hard to attain. First, query latency is a metric that is the most perturbed by both engineering optimizations, such as SIMD operations, as well as the underlying hardware. Therefore, even if one attains a theoretical framework for bounding the expected query latency on a proximity graph, it is still very plausible that pure engineering optimizations could realize better performance in practice. Second, even if we control for engineering optimizations and the underlying hardware, it is not sufficient to consider the most obvious factors including graph complexity (i.e., $|V|$ and $|E|$) and the dimensionality of the underlying vector space d. Different graph construction procedures and pruning algorithms will induce different expected latency bounds. Future research directed to this theoretical understanding of the hub-highway hypothesis will prove to be invaluable to both theoreticians and practitioners alike.

E. Extended Benchmarks

E.1. Building FlatNav from Scratch

In the main body of the paper, we present a series of results demonstrating that there is essentially no difference in performance between search over HNSW with a full hierarchy and search over the flat NSW base graph. As we discuss in Section 3, we fix the experimental design in the main body of the paper to extract the base NSW graph from the *same* `hnswlib` code that constructs the full HSNW index and use our `flatnav` implementation of the search algorithm to traverse the graph. In other words, we construct the full hierarchical graph with `hnswlib` and then extract the base layer as our separate flat graph index. We designed the benchmarking experiments in this manner to avoid any potential confounding effects from differences in code between the baseline HNSW graph construction and our own version.

However, this experimental setup raises a separate concern over whether the hierarchical component of HNSW might still be useful to *construct* the base graph index even if the hierarchy is not used during search. In this extended benchmark, we provide additional results to demonstrate that this is not the case. In particular, when we benchmark the full hierarchical HNSW index from `hnswlib` against a flat graph built from scratch with no hierarchy at all via our `flatnav` library, we observe identical results to what we report in Section 3. Thus, we can conclude that the hierarchical component of HNSW does not seem to be useful for high-dimensional workloads for either construction or search.

In Figures 10 and 11 we show the benchmarking results of HNSW and FlatNav with the latter built from scratch with no hierarchy even in construction. The results are identical to our findings in Figures 4 and 5 in Section 3 in the main body of the paper. For these extended benchmarks, we use a cloud server equipped with an AMD EPYC 9J14 96-Core Processor and 1 TB of RAM.

F. Extended Hubness Experiments

F.1. Extending the Hub-Highway Hypothesis to LLM Embeddings

To evaluate whether the Hub-Highway Hypothesis generalizes beyond sythetic gaussian-distributed and ANN benchmark datasets, we apply the same analysis to large language model (LLM) embeddings used in information retrieval. In particular, we examine the MSMARCO (Campos et al., 2016) dataset, a widely-used retrieval benchmark comprising millions of real-world queries and documents.

Data generation. We encode all training split queries in MSMARCO using the all-MiniLM-L6-v2 model from the `SentenceTransformers` (Reimers & Gurevych, 2019) library. This yields 384-dimensional vector representations. Using these embeddings, we construct a k-NN proximity graph (using $k = 100$) and compute the node access distribution $P_m(\mathbf{x}_i)$, defined as the number of times node $\mathbf{x}_i$ is visited across a fixed set of queries using HNSW beam search heuristic. As with all experiments, we fix the number of queries to be 10,000.

Results. Figure 12 shows the log-normalized node access frequency distribution for MSMARCO. The distribution exhibits a long-tail behavior similar to what we observed in synthetic ℓ_2 datasets and the Big-ANN benchmarks, with a clear skew indicating that a small subset of nodes are accessed orders of magnitude more frequently than others. This supports our hypothesis that highway-like structures emerge naturally even in real-world retrieval embeddings

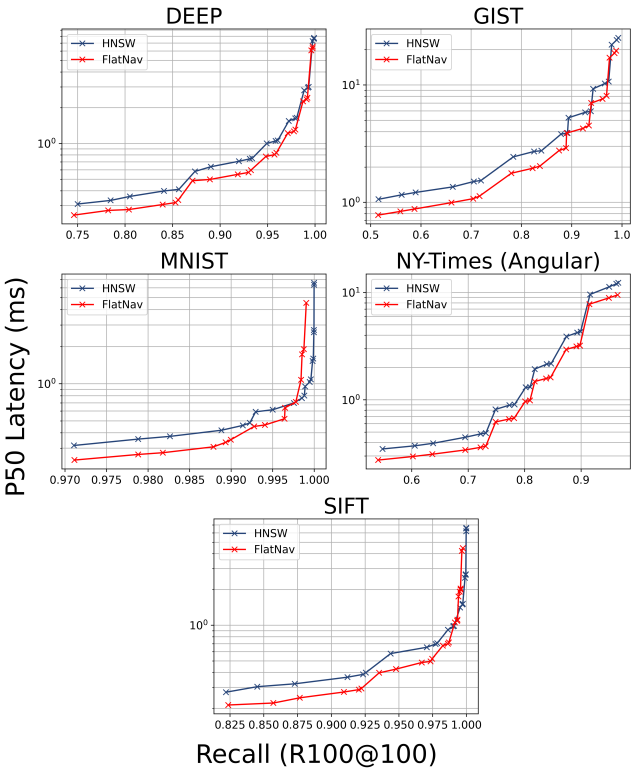


Figure 10. The p50 Latency vs Recall relationship between HNSW and FlatNav (constructed from scratch) is identical to the relationship between HNSW and FlatNav (base layer extracted from HNSW) shown in Figure 4.

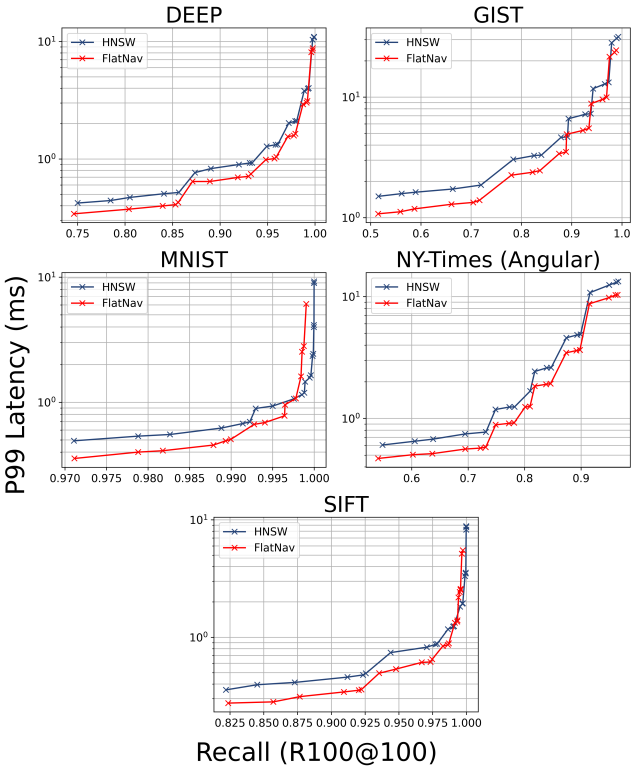


Figure 11. The p50 Latency vs Recall relationship between HNSW and FlatNav (constructed from scratch) is identical to the relationship between HNSW and FlatNav (base layer extracted from HNSW) shown in Figure 5.

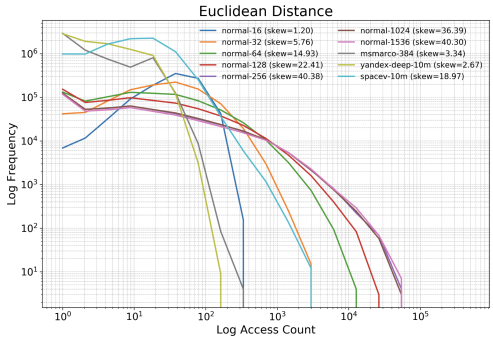


Figure 12. Log-normalized node access count distribution $P_m(\mathbf{x}_i)$ for ℓ_2 datasets along with MSMARCO embeddings.

generated by LLMs.

These findings suggest that the routing behavior of hub nodes is not an artifact of synthetic data or benchmark construction, but a general phenomenon in high-dimensional embedding spaces. This reinforces our claim that such hubs can effectively replace the hierarchy in modern ANN graph indexes.

F.2. No Hierarchy Memory Savings

Dataset	Dataset Size	`hnswlib`	`flatnav`
BigANN	100M	183	113
Microsoft SpaceV	100M	104	85.5
Yandex DEEP	100M	100	60.7

Table 6. Peak Index Construction Memory in GBs. We observe that `flatnav` requires considerably less memory during construction compared to `hnswlib`.

Section 3 focused on demonstrating that we can remove the hierarchy in HNSW with impunity for latency benchmarks. Here we turn our attention to memory consumption and measure the memory savings from removing the hierarchy by running a memory profiler during index construction. In terms of memory allocation, similarly to `flatnav`, `hnswlib` allocates static memory during index construction comprising base layer node allocation, a visited node list and a list of mutexes in the multi-threaded setting. Additionally, it also incurs memory cost attributable to the hierarchy, particularly maintaning the dynamically allocated links between nodes at each layer.

We benchmarked both libraries against a subset of the BigANN benchmarks. Table 6 shows the peak memory allocated by the two implementations during index construction for the BigANN, Microsoft SpaceV and Yandex DEEP benchmarks. Since multithreading has a runtime overhead, we fix the number of cores to 32 in each one of the stated benchmark. For BigANN, we observe a 38% reduction in

peak memory, a 39% reduction for Yandex DEEP, and an 18% reduction for the Microsoft SpaceV benchmark. This shows that we are able to save significant memory by removing the hierarchy, and it is likely that we can optimize `flatnav` implementation to save memory further.

One caveat of these reported memory savings is that we are comparing different two software implementations in `hnswlib` and `flatnav`. Since `hnswlib` is a mature library widely used by practitioners as well as researchers, it supports more features than `flatnav` and thus must maintain additional complexity whereas our implementation, while performant, is more of a research prototype. Therefore, differences in code may account for a significant part of the peak memory usage differences. Nevertheless, we believe our findings are relevant and still noteworthy given that `hnswlib` is so widely adopted. By demonstrating that we can considerably reduce the memory overhead of `hnswlib` without sacrificing performance, we hope to bring the community's attention to the opportunities for further optimization in this direction.

F.3. Hub formation: Discerning Metric Space Hubness from Preferential Attachment

While Figure 7 confirms the existence of hub nodes in the dataset, it does not distinguish between hubs that arise from the properties of the underlying metric space and hubs that form through some other mechanism. It is possible that *preferential attachment* explains the formation of hubs, since NSW graphs are built incrementally by sequentially adding points to an existing graph. Nodes that are added early in graph construction may become hubs by accumulating a greater-than-average share of inbound graph links, rather than by being popular neighbors in the metric space.

To investigate the effects of preferential attachment, we computed the variance (R^2 of a linear model) in the empirical node access distribution explained by the insertion ordering(Table 7). We log-transformed both the node access count and the insertion order before running the linear model and confirmed that the residuals are approximately normal after examining the QQ plots ($p < 10^{-6}$ for all models).

We observe a modest effect from the node insertion order in our synthetic data. This is particularly true for the angular datasets, which we believe to be due to the weaker hubness phenomena produced by the angular distance metric. Preferential attachment may account for a relatively greater share of the node access distribution when metric hubs are not present to heavily skew the distribution.

Ideally, we would repeat this analysis using the K-occurrence distribution, to show that the hubness of the metric space is more strongly predictive of the node access count than the insertion order. Unfortunately, it is not feasible to compute the K-occurrence distribution due to the $O(n^2)$ brute-force computation cost. However, we believe that the results in Table 7 still support the idea that the dimensionality of the metric space strongly contributes to the formation of hubs in the *Hub-Highway Hypothesis*, especially when combined with the evidence in Figure 7.

Table 7. Correlation analysis for node insertion order and node access count, to determine effects of preferential attachment.

Dataset	Dimension	Explained Variance (%)
IID Normal (Angular)	16	3.0%
IID Normal (Angular)	32	8.7%
IID Normal (Angular)	64	16.6%
IID Normal (Angular)	128	23.3%
IID Normal (Angular)	256	24.6%
IID Normal (Angular)	1024	24.1%
IID Normal (Angular)	1536	23.9%
IID Normal (L2)	16	$< 0.1\%$
IID Normal (L2)	32	3.1%
IID Normal (L2)	64	7.6%
IID Normal (L2)	128	8.7%
IID Normal (L2)	256	7.1%
IID Normal (L2)	1024	7.1%
IID Normal (L2)	1536	6.6%
GloVe (Angular)	100	0.2%
NYTimes (Angular)	256	0.3%
GIST (L2)	960	$< 0.1\%$
Yandex-DEEP (L2)	96	0.3%
Microsoft-SpaceV (L2)	100	0.5%