DS SERVE: A Framework for Efficient and Scalable Neural Retrieval

Jinjian Liu[1*], **Yichuan Wang**[1*], **Xinxi Lyu**[2], **Rulin Shao**[3],
Joseph E. Gonzalez[1], **Matei Zaharia**[1], **Sewon Min**[1]

[1] University of California, Berkeley
[2] University of Illinois Urbana–Champaign
[3] University of Washington
leifmax@berkeley.edu, yichuan_wang@berkeley.edu, seanlyu2@illinois.edu, rulins@cs.washington.edu,
jegonzal@cs.berkeley.edu, matei@berkeley.edu, sewonm@berkeley.edu

Abstract

We present DS SERVE, a framework that transforms large-scale text datasets—comprising half a trillion tokens—into a high-performance neural retrieval system. DS SERVE offers both a web interface and API endpoints, achieving low latency with modest memory overhead on a single node. The framework also supports inference-time tradeoffs between latency, accuracy, and result diversity. We anticipate that DS SERVE will be broadly useful for a range of applications such as large-scale retrieval-augmented generation (RAG), training data attribution, training a search agent, and beyond.

Code — github.com/Berkeley-Large-RAG/RAG-DS-Serve
Demo URL — http://api.ds-serve.org:30888/ui

Introduction

Neural retrieval over large-scale text datasets comprising nearly a trillion tokens has become central to modern machine learning, powering applications from retrieval-augmented generation (RAG) to training data attribution and curation. Yet deploying such systems at this scale remains difficult: high latency and memory requirements make them costly and often impractical for fast inference or interactive use. For example, a dataset from CompactDS—a pre-training dataset with half a trillion tokens (Lyu et al. 2025)—produces two billion 768-dimensional vectors, with raw embeddings exceeding 5TB, posing a significant challenge for existing retrieval frameworks. Current systems typically optimize for either accuracy or efficiency, but rarely both, and often require distributed infrastructure, creating barriers for researchers and practitioners with limited resources.

To this end, we present DS SERVE, a framework that transforms massive text datasets into a neural retrieval system designed to run efficiently on a single node. DS SERVE leverages approximate nearest neighbor search, and is capable of handling billions of vectors (e.g., 2B in our deployment). It further supports exact and diversity-based reranking, enabling inference-time tradeoffs among accuracy, latency, and diversity. On CompactDS data, DS SERVE delivers subsecond latency with modest memory overhead (≈200GB RAM), demonstrating scalability without distributed infrastructure. DS SERVE provides both a web interface and API endpoints, easing integration across diverse workflows.

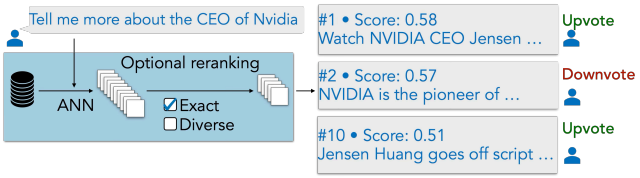

Figure 1: DS SERVE converts a large dataset into a neural retrieval system: a query q retrieves relevant text via ANN (DiskANN or IVFPQ), optionally reranks with Exact and/or Diverse Search, and returns the top-k chunks with voting for feedback.

In summary, DS SERVE enables building *fully controllable* in-house retrieval systems over arbitrary *large-scale* datasets on a single node, with accessible interaction via a web interface and API endpoints. DS SERVE is broadly applicable to large-scale RAG, training data attribution, search agent training, and other retrieval-intensive applications.

Description of DS SERVE

DS SERVE is a framework that transforms a large-scale text corpus $\mathcal{D}$ into a high-performance neural retrieval system. This retrieval system takes a user query q as input and returns a set of text chunks $\{d_1, \ldots, d_k\} \subset \mathcal{D}$, where k is a user-specified hyperparameter. The retrieval system is primarily based on the DiskANN backbone, with two optional additional modes: *Exact Search*, which improves accuracy, and *Diverse Search*, which enhances result diversity.

DS SERVE Backend

Datastore. DS SERVE can process arbitrary in-house datasets at massive scale. While prior work (Shao et al. 2024; Lyu et al. 2025) has shown that retrieval over large pre-training corpora can improve RAG accuracy, such efforts lacked accessible frameworks that allow non-experts to build and interact with indexes. In this paper, We demonstrate DS SERVE on CompactDS (Lyu et al. 2025), a 380B-word corpus (2B vectors) across high-quality sources at a

*These authors contributed equally.

scale far exceeding prior studies (< 50M vectors (Jin et al. 2024, 2025; Hu et al. 2025)) and typical commercial services (< 500M per namespace (Turbopuffer 2025)).

Approximate Nearest Neighbor (ANN) Search. Neural retrieval is formulated as a nearest neighbor search that selects the top-k chunks with the highest similarity scores $\mathrm{sim}(\mathbf{q}, \mathbf{d}_i)$, where $\mathbf{q}, \mathbf{d}_i \in \mathbb{R}^h$ are the vector representations of the query q and a candidate chunk $d_i \in \mathcal{D}$, respectively.[1]

As the datastore $\mathcal{D}$ scales to billions of vectors, efficient nearest neighbor search becomes a central challenge, since linear scans become infeasible. To address this, we adopt DiskANN (Subramanya et al. 2019), a graph-based ANN method that expands a beam of W neighbors while traversing a navigable graph stored primarily on disk. This search implicitly re-ranks candidates in full precision without re-embedding and exposes tunable search complexity L and beam width W so users can explore the accuracy-latency tradeoff. Empirically, DiskANN achieves higher accuracy than IVFPQ (Jégou, Douze, and Schmid 2011) and over 200 end-to-end QPS even with high search complexity. Thus, DiskANN is adopted as the default ANN backend in our framework; users can optionally switch to IVFPQ at their own discretion.

Exact Search. While ANN is fast and efficient, it inevitably sacrifices retrieval accuracy, especially at large $|\mathcal{D}|$ where aggressive quantization is required. To mitigate this limitation, DS SERVE provides an optional reranking mode based on exact search. In this mode, ANN first retrieves the top-K candidates ($K > k$), which are then reranked using exact similarity scores $\mathrm{sim}(\mathbf{q}, \mathbf{d}_i)$. We use GritLM (Muennighoff et al. 2024) to compute exact similarity between passages and queries, after which the true top-k passages are returned. Passage vectors are recomputed on the fly during cold start but cached for subsequent queries, typically reducing the latency to below 0.5s when similar queries are posed. As shown in our evaluation, exact search consistently improves accuracy across all tasks (Table 1).

Diverse Search. Search results often contain substantial overlap, returning nearly identical passages, limiting information breadth. We introduce a *Diverse Search* option , explicitly discouraging redundancy to improve coverage. Concretely, we apply maximal marginal relevance (MMR) (Carbonell and Goldstein 1998): at step t, given the already selected set $\mathcal{S}$, each remaining candidate i receives a score:

$$\lambda \, \mathrm{sim}\big(\mathbf{q}, \mathbf{d}_i\big) \; - \; (1-\lambda) \, \max_{j \in \mathcal{S}} \mathrm{sim}\big(\mathbf{d}_i, \mathbf{d}_j\big).$$

We find that diverse search substantially improves user experience, though it may not necessarily improve RAG performance. We leave its further evaluation to future work.

Interface Design

DS SERVE provide a web interface and API endpoints, with inference-time tunable parameters: k, two optional post-ANN search modes (Exact and Diverse Search), n_{probe} and λ, controlling the number of chunks retrieved and the tradeoffs among accuracy, latency, and diversity. Users can cast a one-click relevance vote for each chunk, with labels stored for system development and evaluation (Figure 1).

[1] $\mathbf{q} = \mathrm{enc}(q)$ and $\mathbf{d}_i = \mathrm{enc}(d_i)$, where we use Contriever (Izacard et al. 2021) as the encoder. The function $\mathrm{sim}(\cdot, \cdot)$ denotes the cosine similarity between two embeddings.

	No DS SERVE	DS SERVE		DS SERVE w/ Exact		
Task	Acc	Acc	t	Acc	t	t_{cache}
MMLU	68.9	73.5	0.17	73.7	16.44	0.30
MMLU Pro	39.8	47.5	0.19	49.4	16.54	0.32
AGI Eval	56.2	56.2	0.21	58.3	15.03	0.34
MATH	46.9	50.0	0.18	53.1	16.51	0.33
GPQA	29.9	31.7	0.17	36.6	16.57	0.32

Table 1: Evaluation of DS SERVE on five established benchmarks. '*Acc*' is accuracy (%), and t is end-to-end *retrieval* latency (s). For Exact Search, we report t without cache and t_{cache} with cache. We use $K = 1000$, $k = 10$, and $n_{\text{probe}} = 256$ for all tasks.

Evaluation and Application

We evaluate DS SERVE for RAG applications and discuss other potential use cases.

RAG. We evaluate a RAG model based on DS SERVE on five established benchmarks (Table 1). DS SERVE substantially improves accuracy over the baseline with negligible latency overhead, with further gains achieved through exact search. We report the upper bound of exact search latency, but it is often much lower in practice when the same or similar queries were issued previously and benefit from caching.

Data Attribution and Curation. DS SERVE can readily be used for training data attribution by indexing the entire pretraining corpus. The closest prior system, OLMoTrace (Liu et al. 2025), relies on n-gram matching, whereas DS SERVE considers semantic similarity, making it complementary to or more accurate than OLMoTrace. In addition, DS SERVE enables improved data curation through semantic deduplication, decontamination, and customized filtering (e.g., identifying subsets of large datasets relevant to specific queries).

Training a Search Agent. Search agents are in high demand for applications such as deep research; however, training them is challenging, as rollouts require high-QPS search calls (Jin et al. 2025), and commercial search engines are costly, slow, and rate-limited. DS SERVE addresses these issues by providing a fully controllable search backend, allowing developers to set their own latency-accuracy tradeoffs without incurring costs or rate limits.

Pushing the Frontier of Search. Commercial web search engines (e.g., Google) are powerful, but they have room for improvement: they perform well on short keyword queries but struggle with long or complex inputs. Vector-based retrieval is more effective in such cases and can complement or even outperform traditional search engines (as shown in Lyu et al. (2025)). Our voting features also enable collection of real-world labeled data, enabling creation of realistic benchmarks and training data for retrieval research.

References

Carbonell, J. G.; and Goldstein, J. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 335–336. Melbourne, Australia: ACM.

Hu, Z.; Murthy, V.; Pan, Z.; Li, W.; Fang, X.; Ding, Y.; and Wang, Y. 2025. HedraRAG: Coordinating LLM Generation and Database Retrieval in Heterogeneous RAG Serving. *ACM Symposium on Operating Systems Principles (SOSP 25)*.

Izacard, G.; Caron, M.; Hosseini, L.; Riedel, S.; Bojanowski, P.; Joulin, A.; and Grave, E. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1): 117–128.

Jin, B.; Yoon, J.; Han, J.; and Arik, S. O. 2024. Long-context llms meet rag: Overcoming challenges for long inputs in rag. *arXiv preprint arXiv:2410.05983*.

Jin, B.; Zeng, H.; Yue, Z.; Yoon, J.; Arik, S.; Wang, D.; Zamani, H.; and Han, J. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Liu, J.; Blanton, T.; Elazar, Y.; Min, S.; Chen, Y.; Chheda-Kothary, A.; Tran, H.; Bischoff, B.; Marsh, E.; Schmitz, M.; Trier, C.; Sarnat, A.; James, J.; Borchardt, J.; Kuehl, B.; Cheng, E.; Farley, K.; Sreeram, S.; Anderson, T.; Albright, D.; Schoenick, C.; Soldaini, L.; Groeneveld, D.; Pang, R. Y.; Koh, P. W.; Smith, N. A.; Lebrecht, S.; Choi, Y.; Hajishirzi, H.; Farhadi, A.; and Dodge, J. 2025. OLMoTrace: Tracing Language Model Outputs Back to Trillions of Training Tokens. arXiv preprint arXiv:2504.07096. V2, submitted to ACL 2025 demo track.

Lyu, X.; Duan, M.; Shao, R.; Koh, P. W.; and Min, S. 2025. Frustratingly Simple Retrieval Improves Challenging, Reasoning-Intensive Benchmarks. *arXiv preprint arXiv:2507.01297*.

Muennighoff, N.; Hongjin, S.; Wang, L.; Yang, N.; Wei, F.; Yu, T.; Singh, A.; and Kiela, D. 2024. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Shao, R.; He, J.; Asai, A.; Shi, W.; Dettmers, T.; Min, S.; Zettlemoyer, L.; and Koh, P. W. 2024. Scaling Retrieval-Based Language Models with a Trillion-Token Datastore. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Subramanya, S. J.; Devvrit; Kadekodi, R.; Krishnaswamy, R.; and Simhadri, H. V. 2019. DiskANN: Fast Accurate Billion-Point Nearest Neighbor Search on a Single Node. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 13701–13711. Curran Associates, Inc.

Turbopuffer. 2025. Turbopuffer Pricing. https://turbopuffer.com/. Accessed: 2025-09-13.