

On The Reproducibility Limitations of RAG Systems

Baiqiang Wang
 University of Washington
 Seattle, WA, USA
 wbq@uw.edu

Nathan R Tallent
 Pacific Northwest National Laboratory
 Richland, WA, USA
 nathan.tallent@pnnl.gov

Dongfang Zhao
 University of Washington
 Seattle, WA, USA
 dzhao@uw.edu

Luanzheng Guo
 Pacific Northwest National Laboratory
 Richland, WA, USA
 lenny.guo@pnnl.gov

Abstract

Retrieval-Augmented Generation (RAG) is increasingly employed in generative AI-driven scientific workflows to integrate rapidly evolving scientific knowledge bases, yet its reliability is frequently compromised by non-determinism in their retrieval components. This paper introduces **ReproRAG**, a comprehensive benchmarking framework designed to systematically measure and quantify the reproducibility of vector-based retrieval systems. ReproRAG investigates sources of uncertainty across the entire pipeline, including different embedding models, precision, retrieval algorithms, hardware configurations, and distributed execution environments. Utilizing a suite of metrics, such as Exact Match Rate, Jaccard Similarity, and Kendall's Tau, the proposed framework effectively characterizes the trade-offs between reproducibility and performance. Our large-scale empirical study reveals critical insights; for instance, we observe that different embedding models have remarkable impact on RAG reproducibility. The open-sourced ReproRAG framework provides researchers and engineers productive tools to validate deployments, benchmark reproducibility, and make informed design decisions, thereby fostering more trustworthy AI for science.

1 Introduction

Retrieval-Augmented Generation (RAG) [12] has become a cornerstone architecture for building knowledge-intensive AI systems, particularly in scientific research [20]. By grounding LLMs with evidence retrieved from vast corpora, RAG systems mitigate factual hallucinations [8] and provide access to domain-specific, timely information. However, as these systems move from prototypes to critical components in scientific workflows, their reliability and consistency become paramount, a central concern in the broader pursuit of Trustworthy AI [17].

A fundamental challenge that threatens this reliability is non-determinism, a well-documented issue in parallel and deep learning systems [16]. A RAG system, even with identical inputs, can produce different results across multiple runs. This inconsistency often originates in the vector-based retrieval stage. Despite its importance, there is a lack of standardized tools and methodologies to quantify this specific form of non-determinism. While performance benchmarking for vector search has been studied [1], it does not address reproducibility, forcing researchers to rely on anecdotal evidence or ad-hoc tests. Without rigorous measurement, it is impossible to understand the trade-offs between performance and

consistency, validate production deployments, or confidently build upon prior computational results.

To fill this measurement gap, we introduce **ReproRAG**, a comprehensive testing framework for analyzing and measuring the reproducibility of RAG retrieval systems. Our contributions are:

- We design and implement **ReproRAG**, an open-source framework for systematically benchmarking the reproducibility of RAG retrieval pipelines, defining a suite of metrics to quantify multiple dimensions of uncertainty.
- We present a large-scale empirical study that establishes a **hierarchy of uncertainty**, demonstrating that the choice of embedding model and dynamic data insertion are the most significant sources of result variation.
- We provide a detailed empirical analysis that challenges common assumptions, demonstrating that core ANN retrieval algorithms and distributed protocols can achieve perfect run-to-run reproducibility, and that common software-level determinism flags may have no observable effect on modern embedding models.
- We offer **actionable insights for practitioners**, including a quantification of precision-induced "embedding drift" and evidence that performance-oriented settings can provide a speedup without harming reproducibility for certain classes of models.

2 Background and Related Work

2.1 Vector Search and Non-Determinism

Modern RAG systems rely on Approximate Nearest Neighbor (ANN) search to efficiently find relevant documents in high-dimensional vector spaces. Libraries like FAISS (Facebook AI Similarity Search) [9] implement various ANN algorithms. These methods often trade perfect accuracy for significant speed improvements, but this trade-off can introduce sources of non-determinism.

For instance, **Hierarchical Navigable Small World (HNSW)** [13] builds a probabilistic graph for searching, where choices made during graph construction can be a source of randomness. Similarly, **Inverted File (IVF)** indexes first partition the vector space using k-means clustering, whose random initialization can lead to different index structures [7]. Another common technique, **Locality-Sensitive Hashing (LSH)** [5], uses random projections to group

similar items, where the choice of projections is inherently stochastic. Furthermore, beyond algorithmic randomness, parallel execution on multi-core CPUs or GPUs can introduce non-determinism through the reordering of floating-point operations.

2.2 Related Work

Our work is situated at the intersection of three key areas: the application of RAG in science, the broader challenge of computational reproducibility, and the practice of benchmarking AI systems.

RAG in Scientific Workflows. The application of Large Language Models to accelerate scientific discovery is a rapidly growing field [20]. RAG frameworks, in particular, are critical for grounding these models in specialized, up-to-date scientific literature, making them powerful tools for tasks like hypothesis generation and literature review. While frameworks like SciTrust [6] evaluate the trustworthiness of the LLM component itself, they assume a stable, reproducible input. Our work is complementary, focusing on ensuring the reliability of the retrieval stage that produces this input.

The Reproducibility Crisis in Computational Science. The challenge of reproducibility is a well-documented issue across computational science [4]. In response, the high-performance computing community has championed efforts to improve the repeatability of research, for instance, through the introduction of Artifact Description and Evaluation tracks at major conferences [3]. Our framework applies these established principles of rigorous measurement and validation to the specific and novel challenges posed by large-scale vector search systems.

Benchmarking AI and Vector Search Systems. Standardized benchmarking is essential for the progress of AI systems. Efforts like MLPerf [15] have become industry standards for measuring ML training and inference performance. In the domain of vector search, frameworks like ann-benchmarks [1] provide comprehensive performance comparisons of different ANN algorithms. ReproRAG extends this tradition of rigorous benchmarking to a new and critical dimension: reproducibility. While existing benchmarks focus on speed and accuracy, our work provides the first comprehensive framework for quantifying the consistency and deterministic behavior of these systems.

3 The Test Framework

ReproRAG is architected to be a flexible and scalable testing harness. It systematically creates experimental configurations, executes repeated retrieval runs, and analyzes the results to generate detailed reproducibility reports.

3.1 Sources of Non-Determinism Investigated

The framework is designed to isolate and measure uncertainty from four primary sources:

- (1) **Embedding Uncertainty:** Variations arising from the choice of embedding models and their numerical precision (e.g., FP32 vs. FP16).
- (2) **Retrieval Uncertainty:** Inconsistencies stemming from the choice of indexing techniques (e.g., Flat, IVF, HNSW) and the selection of parameters.

- (3) **Hardware Variations:** The impact of the underlying hardware, including differences between CPU and GPU execution and parallel execution effects.
- (4) **Distributed Computing:** Consistency challenges in multi-node environments, including data sharding strategies and inter-node communication.

3.2 Architecture and Methodology

Motivated by these challenges, ReproRAG employs a systematic methodology integrating multi-level testing and controls.

System Architecture. The framework is built with a modular architecture: an **ExperimentConfig** module for defining test parameters; a **FaissRetrieval** core engine; an MPI-based **DistributedFaissRetrieval** module; and a **ReproducibilityMetrics** library for analysis.

Determinism Control. To isolate sources of randomness, the framework implements comprehensive controls, including standardization of random seeds, enforcement of deterministic CUDA operations, and fixed thread counts.

Multi-Precision Analysis. To evaluate embedding uncertainty, the framework tests multiple precision configurations, including full-precision FP32, and half-precision FP16 for memory efficiency. It also supports specialized formats like TF32 and BF16 on compatible hardware.

Hardware and Environment Profiling. The framework automatically detects GPU architecture, profiles memory utilization, and logs driver and library versions to ensure environmental consistency is tracked throughout all experiments.

3.3 Core Reproducibility Metrics

ReproRAG utilizes two levels of metrics.

Embedding Stability Metrics. To measure the consistency of the generated vectors directly, before they are used in a search index, we compute:

- **L2 Distance** [11]: Also known as Euclidean distance, this measures the absolute, straight-line distance between two embedding vectors in the vector space. A score of 0.0 indicates that the vectors are numerically identical.
- **Cosine Similarity** [14]: This measures the cosine of the angle between two vectors, evaluating their semantic orientation independent of magnitude. A score of 1.0 indicates the vectors point in the exact same direction.

Retrieval Metrics. To measure the final outcome of the retrieval task, we use:

- **Jaccard Similarity** [11]: This metric evaluates the overlap between two sets of retrieved documents (A and B), ignoring their rank. It is calculated as the size of the intersection divided by the size of the union ($|A \cap B|/|A \cup B|$). A score of 1.0 indicates the two runs returned the exact same set of documents.
- **Kendall's Tau** [10]: This measures the rank correlation between two lists of retrieved results. It evaluates the similarity of the ordering of documents by counting the number

of concordant and discordant pairs. A score of 1.0 indicates that the documents common to both lists were ranked in the exact same order.

- **Score Stability:** This is the standard deviation of the similarity scores (e.g., L2 distances) of the top-k retrieved documents across multiple runs. A lower value indicates higher numerical consistency in the ranking function itself.

4 Experimental Evaluation

We now present our empirical evaluation using the ReproRAG framework to characterize reproducibility across several key dimensions discussed in Section 3.

4.1 Experimental Setup

Unless otherwise specified, all experiments share the following setup:

- **Platform:** The TAMU ACES HPC platform [19], utilizing both CPU-only and GPU-accelerated nodes.
- **Dataset:** We use a Medium Scale configuration (100,000 documents; 1,000 queries) from MSMARCO dataset [2].
- **Embedding Models:** To ensure our findings are generalizable, we evaluated three diverse, high-performing, and widely-used transformer-based models:
 - **BGE (BAAI General Embedding)** [23]: We used the bge-base-en-v1.5 model, a powerful sentence embedding model known for its strong performance on retrieval benchmarks.
 - **E5 (Embedding from ELI5)** [21]: We used the intfloat/e5-base-v2 model, which is notable for its effective training on a massive corpus of text pairs using contrastive pre-training.
 - **Qwen** [18]: We used the Qwen3-0.6B-Base model, a recent and powerful open-source text embedding model from a new generation of large language models.
- **Measure:** For each configuration, every query is executed over five runs. We report the average measurement.

4.2 Scenario 1: Temporal Reproducibility Under Data Insertion

4.2.1 Goal and Methodology. Our evaluation begins by addressing a practical challenge that extends beyond static, run-to-run consistency: **temporal reproducibility**. We aim to answer the question: how consistently can an index reproduce its initial search results after new data has been incrementally added to it? This scenario evaluates the stability of search outcomes over time in a common real-world situation where rebuilding an index from scratch after every update is computationally prohibitive.

Experimental Protocol. The methodology is designed to quantify the reproducibility of search results between a “before” (V1) and “after” (V2) state of the same index instance.

- (1) **Initial State (V1):** An index (e.g., HNSW, IVF) is created and trained (if applicable) using an initial dataset of ~7,920

documents. This establishes a fixed internal structure. These initial documents are then added to the index.

- (2) **Baseline Query:** A standard set of 100 queries is executed against this V1 index. The top-50 retrieved document IDs for each query are recorded as **Results_V1**, representing the original, reproducible state.
- (3) **Incremental Update:** A new set of ~1,980 documents is added directly to the same index instance using the `index.add()` function. No re-training or re-structuring of the index occurs.
- (4) **Follow-up Query:** The exact same 100 queries are executed again against the updated index. The top-50 results are recorded as **Results_V2**.

Stability Metrics. The degree to which **Results_V1** are reproduced in **Results_V2** is quantified using metrics specifically suited for comparing ranked lists:

- **Overlap Coefficient:** The percentage of documents from the V1 result list that are also present in the V2 list.
- **Kendall’s Tau:** Measures the rank-order correlation for the documents that appear in both result lists.
- **Rank-Biased Overlap (RBO)**[22]: A comprehensive, top-weighted score that evaluates the similarity of the two lists, penalizing disagreements at higher ranks more severely. An RBO of 1 indicates perfect temporal reproducibility.

4.2.2 Results and Discussion. Our experiment reveals a nuanced and consistent picture of how different indexing methods handle incremental updates. The primary finding is that while data insertion inevitably causes some “churn” in the retrieved results, the internal ranking of the original, persistent results remains perfectly stable across all tested approximate indexing methods.

Quantifying Result Churn. Table 1 summarizes the stability metrics, averaged across 100 queries. All three methods—HNSW, IVF, and LSH—exhibited very similar behavior. The mean Overlap Coefficient was approximately 0.80 for all methods, indicating that on average, 80% of the original top-50 documents were still present in the results after the data insertion. The Rank-Biased Overlap (RBO) score, which gives more weight to top-ranked items, was also remarkably consistent across the methods, averaging around 0.73.

Figure 1 visualizes the distribution of these stability metrics across the 100 test queries for the HNSW index. While the mean scores in Table 1 provide a good summary, the distributions show there is notable variance depending on the specific query. For instance, the Overlap Coefficient (Figure 1b) has a median of 0.80 but ranges from 0.64 to 0.92, indicating that some queries are much more sensitive to data insertion than others. The RBO distribution (Figure 1c) shows a similar spread.

Table 1: Retrieval Stability After Incremental Data Insertion.
Metrics are the mean values across 100 queries.

Index Type	Overlap Coeff.	RBO Score	Kendall’s Tau
HNSW	0.793	0.725	1.000
IVF	0.806	0.730	1.000
LSH	0.804	0.735	1.000

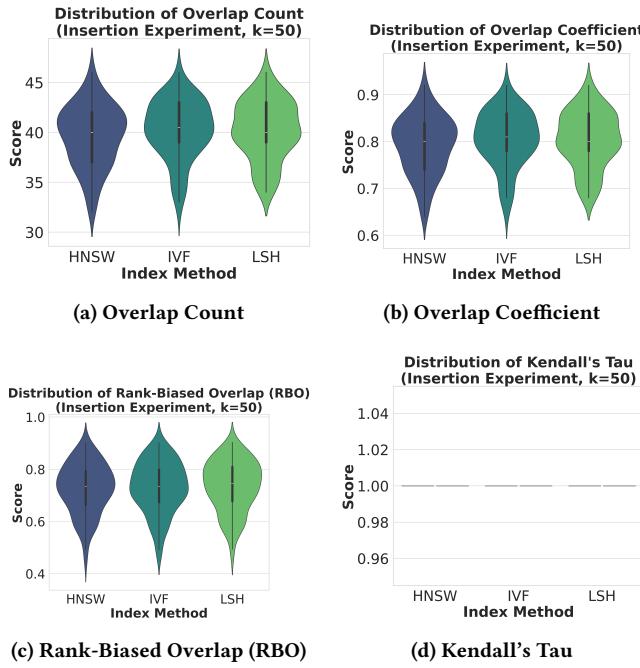


Figure 1: Distributions of temporal reproducibility metrics for the HNSW index across 100 queries after data insertion. The results show variance in overlap (a, b) and RBO (c), but perfect stability in ranking (d).

Perfect Ranking Stability of Persistent Results. The most striking result, visualized in Figure 1d, is the perfect Kendall's Tau score of 1.000 (with zero standard deviation) for all three index types. This reveals a critical insight into the nature of the result churn. It means that for the subset of documents that appeared in both the “before” and “after” result lists, their relative ranking did not change at all. The instability we measured is not due to a re-shuffling of the original results, but rather a process of “displacement”, where new or newly-promoted documents push some of the original results out of the top-50 list.

Discussion. The key takeaway from this scenario is that for these common indexing methods, the primary effect of “lazy” data insertion is displacement, not re-ranking. This makes their behavior more predictable than might be assumed. The perfect Kendall's Tau suggests that the underlying similarity landscape for the original documents is not fundamentally distorted by the addition of new data points. This is a crucial piece of information for practitioners building systems for dynamic environments. It implies that users are likely to see many of their original results preserved, with new, potentially more relevant, items added to the list, rather than seeing a chaotic re-ordering of familiar results.

4.3 Scenario 2: Cross-Embedding Model Retrieval Consistency

4.3.1 Goal and Methodology. Our first scenario revealed that the results from a single RAG pipeline can drift over time as new data is added. We now investigate a different dimension of uncertainty:

at a single point in time, do different state-of-the-art embedding models produce consistent results for the same query? This scenario quantifies the agreement between models on a downstream retrieval task, measuring how contingent the final search results are on the initial choice of embedding model.

Experimental Protocol. To isolate the embedding model as the sole variable, we enforced a strictly controlled retrieval environment:

- **Models Tested:** BGE, E5, and Qwen.
- **Index Type:** We used a FAISS Flat_L2 index for all tests, which performs an exact, brute-force search to eliminate any variation from approximate algorithms.
- **Precision:** All models were run using FP32 precision.
- **Dataset:** A 5,000-document subset of MSMARCO with 100 queries. Top-50 documents were retrieved.

We then performed pairwise comparisons of the top-50 retrieved document lists for each of the three model pairs (BGE vs. E5, BGE vs. Qwen, E5 vs. Qwen) across all 100 queries.

4.3.2 Results and Discussion. Our analysis reveals a significant divergence in the retrieval results produced by the different embedding models, underscoring that the choice of model is a dominant source of system-level uncertainty.

Low Agreement Across All Metrics. As summarized in Table 2, the agreement between the models was low. The Overlap Coefficient ranged from just 0.43 to 0.54, meaning that, on average, two state-of-the-art models only agreed on about half of the retrieved documents for the same query. The top-weighted RBO score was similarly moderate, indicating significant disagreement even at the top of the ranked lists.

The most striking result is the very low Kendall's Tau score (0.32–0.38). This indicates that even for the subset of documents that both models did retrieve, their relative ranking was very different. Furthermore, with p-values > 0.05, this weak correlation is not statistically significant.

Table 2: Mean Retrieval Agreement Between Model Pairs Across 100 Queries.

Model Pair	Overlap Coeff.	RBO Score	Kendall's Tau
BGE vs. E5	0.540	0.570	0.384
BGE vs. Qwen	0.454	0.486	0.338
E5 vs. Qwen	0.432	0.474	0.322

Visualizing the Distribution of Disagreement. Figure 2 visualizes the full distributions of these metrics across the 100 test queries. The violin plots show not only that the median agreement is low but also that there is high variance. For some queries, the models had almost no overlap (RBO scores near 0.2), while for others, they agreed more strongly. This indicates that the disagreement is not uniform but query-dependent.

Discussion. This scenario provides a critical insight for the “Hierarchy of Uncertainty.” While our other scenarios show that a single RAG pipeline can be made perfectly reproducible, this experiment demonstrates that the results are nonetheless highly contingent

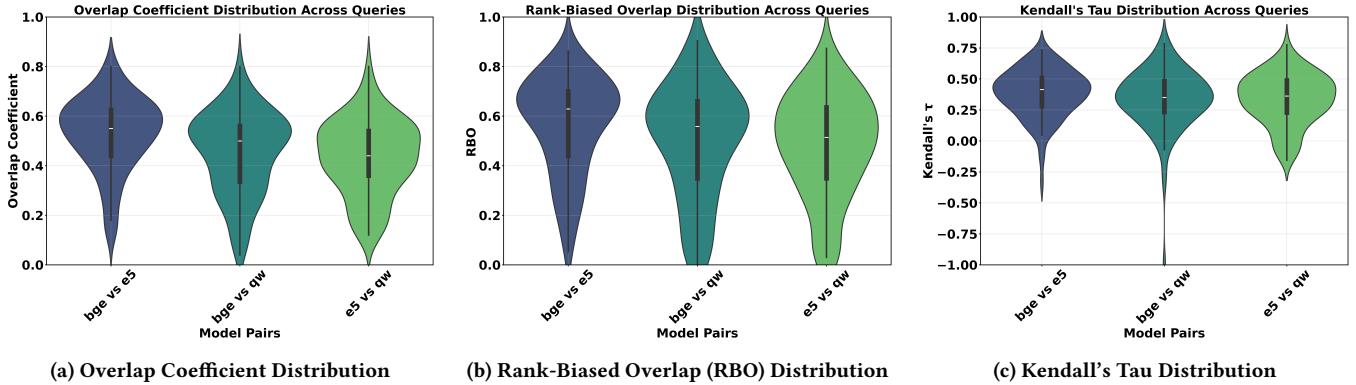


Figure 2: Distributions of retrieval agreement metrics for three model pairs across 100 queries. The low median values and wide distributions highlight a significant and variable lack of consensus between the models.

on the initial choice of embedding model. This represents a major challenge for the reproducibility of scientific findings that rely on RAG systems. A conclusion drawn using a BGE-based RAG may not be reproducible with an E5-based RAG, as they are fundamentally "seeing" different evidence. This underscores the need for researchers to be explicit about the embedding models used in their work and for the community to develop best practices for evaluating the robustness of findings across different models.

4.4 Scenario 3: Precision Uncertainty

4.4.1 Goal and Methodology. We investigate the impact of numerical precision and algorithmic determinism on the stability of the embedding generation process itself, which is a foundational source of uncertainty in any RAG system. The objective is to quantify how variations in floating-point formats and *non-deterministic* (non-det.) settings affect the consistency of vector embeddings before they are used for retrieval.

To achieve this, ReproRAG employs a comprehensive eight-configuration testing framework that examines embedding reproducibility across two key dimensions:

- **Precision Formats:** Four numerical precision types are tested: full-precision FP32, half-precision FP16, and specialized formats BF16 and TF32 on compatible hardware.
- **Deterministic Modes:** Two algorithmic settings are used: a *deterministic* (det.) mode and a non-det. mode. The **det. mode** uses fixed random seeds and deterministic cuDNN settings, while the **non-det. mode** uses time-based seeds and performance-optimized cuDNN settings.

This 8-configuration testing matrix (e.g., FP32-det., FP32-non-det., etc.) is applied to three widely-used embedding models—BGE, E5, and Qwen—on a diverse set of test texts.

Reproducibility Testing Protocol. Our methodology consists of two distinct testing protocols:

- (1) **Same-Configuration Reproducibility:** For each of the eight configurations, we perform five independent embedding generation runs on identical input texts. This protocol tests whether a single, specific setup (e.g., FP32-det.) produces the same embeddings every time it is executed.

- (2) **Cross-Configuration Precision Analysis:** We conduct pairwise comparisons of the generated embeddings *between* different precision formats (e.g., comparing the FP32-det. output to the FP16-det. output). This quantifies the vector drift caused purely by changes in numerical precision.

Technical Implementation. To ensure the integrity of the experiment, ReproRAG enforces strict environment controls:

- In **det. mode**, we fix random seeds across all relevant libraries (`torch.manual_seed(42)`, `np.random.seed(42)`) and enable deterministic cuDNN operations (`cudnn.deterministic=True`, `cudnn.benchmark=False`).
- In **non-det. mode**, we use time-based seeds and optimized, non-deterministic cuDNN settings (`cudnn.benchmark=True`) to reflect a production performance-oriented environment.
- To prevent state leakage, a fresh model instance is loaded for each experimental run.

4.4.2 Results and Discussion. Our evaluation of embedding uncertainty was conducted on three widely-used transformer-based models: BGE, E5, and Qwen. The analysis yielded two clear and impactful findings. First, and most surprisingly, all three models were perfectly reproducible across all tested configurations, and the software-level determinism flags had no observable effect on their output. Second, the choice of numerical precision, while a definite source of variation, results in a quantitatively small “embedding drift”.

The Ineffectiveness of Determinism Flags. Our primary goal was to measure the impact of common determinism controls. We tested eight configurations for each model—four precision formats, each in both a *deterministic* (det.) and a *non-deterministic* (non-det.) mode. The results were uniform.

While all three models produced identical outcomes, we present the detailed results for the BGE model as a representative example.

- (1) **Perfect Same-Configuration Reproducibility:** As shown in Table 3, Every single one of the eight configurations was perfectly reproducible over five independent runs, achieving a mean pairwise L2 distance of 0.0 and cosine similarity of 1.0.

- (2) **Identical det. vs. non-det. Outputs:** As shown in Table 5, for every precision format, the embeddings generated in det. mode were numerically identical to those generated in non-det. mode.

As summarized in Table 4, every single configuration for all three models was perfectly reproducible over five independent runs. This leads to a critical insight: **The deterministic behavior observed is not specific to one model but may be a general property of modern, optimized transformer embedding models on a standard HPC software stack.** The `cudnn.deterministic` and `cudnn.benchmark` flags did not influence the final embedding vectors. This result demonstrates that practitioners cannot assume that these flags will induce or control randomness; reproducibility must be empirically verified.

Table 3: Same-Configuration Embedding Reproducibility over 5 runs. An L2 distance of 0 indicates numerically identical vectors, and a cosine similarity of 1 indicates perfect angular agreement. All eight configurations achieved perfect reproducibility.

Configuration	L2 Distance	Cosine Similarity	Reproducible
FP32-det.	0.00	1.00	Yes
FP32-non-det.	0.00	1.00	Yes
FP16-det.	0.00	1.00	Yes
FP16-non-det.	0.00	1.00	Yes
BF16-det.	0.00	1.00	Yes
BF15-non-det.	0.00	1.00	Yes
TF32-det.	0.00	1.00	Yes
TF32-non-det.	0.00	1.00	Yes

Table 4: Summary of Intra-Configuration Reproducibility Across Three Models. Each model was tested over 8 configurations (4 precisions x 2 determinism modes), with 5 runs per configuration.

Embedding Model	Reproducible Configurations
BGE	8 out of 8
E5	8 out of 8
Qwen	8 out of 8

Quantifying Precision-Induced Embedding Drift. While software flags had no effect, the choice of numerical precision itself was a guaranteed source of variation. Our cross-configuration analysis confirmed that using different floating-point formats fundamentally alters the resulting embeddings. Figure 3 visualizes the pairwise L2 distances between the four precision formats.

The heatmap clearly shows that all off-diagonal comparisons result in non-zero distances. However, the magnitude of this drift is very small. For instance, the distance between the FP32 baseline and FP16 is only $5.74e-04$. This analysis allows us to rank the precision formats by their similarity. TF32 is most similar to the FP32 baseline (L2 distance of $4.09e-04$), making it a strong candidate for

optimization. Conversely, BF16 is the most distinct, exhibiting the largest drift from all other formats (e.g., an L2 distance of $6.31e-03$ from FP32).

This analysis provides two key takeaways. **First, changing precision will cause embedding drift. Second, this drift, while real, is quantitatively small for most pairs.** This allows practitioners to make informed trade-offs between computational efficiency and numerical fidelity, armed with the knowledge of how much the numerical embedding vector space will be perturbed.

Performance Analysis. To complete our analysis of the embedding stage, we measured the execution time for all 24 scenarios (3 models x 8 configurations). Figure 4 shows the embedding generation latency for each model. The results reveal a clear and actionable insight regarding the determinism settings.

Across all three models and all precision formats, we observed a consistent performance trend: the *non-deterministic* (non-det.) mode was measurably faster than the *deterministic* (det.) mode, typically by a margin of 10-15%.

This creates a powerful takeaway when combined with our earlier findings. Our reproducibility analysis showed that for these models, the det. and non-det. modes produced numerically identical embeddings. **This performance analysis now demonstrates that the non-det. mode is not only harmless to reproducibility but is also faster.** This means practitioners can enable the performance-oriented cuDNN settings to gain a “free” and consistent speedup without sacrificing any output reproducibility for this class of models.

Table 5: Comparison of Deterministic between Non-Deterministic Modes Within Each Precision Format.

Precision Type	L2 Distance	Result
FP32	0.00	Identical
FP16	0.00	Identical
BF16	0.00	Identical
TF32	0.00	Identical

4.5 Scenario 4: Stability of the Indexing and Search Stage

4.5.1 Goal and Methodology. The index uncertainty experiment aims to evaluate the reproducibility of vector indexing and search operations, isolating this critical stage from the rest of the RAG pipeline. This investigation addresses whether identical document embeddings consistently produce the same retrieval behaviors across multiple runs. The primary goals are to (1) Quantify the deterministic behavior over different FAISS indexing techniques, (2) Assess the impact of GPU-accelerated search on retrieval consistency, and (3) Measure how indexing variations propagate to final retrieval results.

To achieve this, ReproRAG employs a multi-layered testing approach using a controlled environment.

Controlled Environment and Configurations. Each test run is initialized with identical random seeds (42) when *deterministic* (det.)

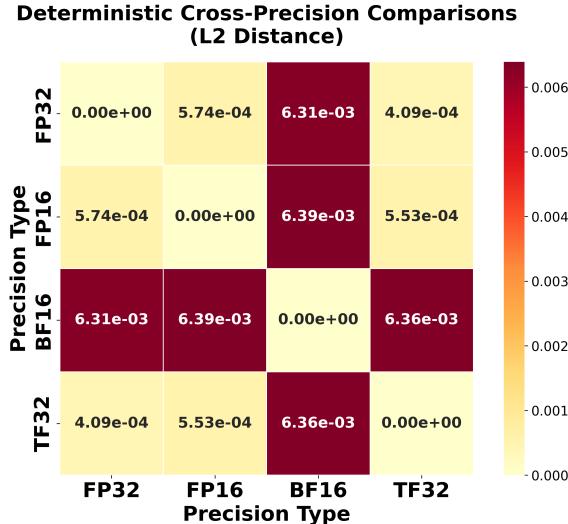


Figure 3: Heatmap of pairwise L2 distances between precision formats in Deterministic mode. Darker colors indicate greater dissimilarity.

mode is enabled. The framework also supports a *non-deterministic* (non-det.) mode using time-based seeds for sensitivity analysis. The evaluation covers multiple FAISS indexing techniques, including IndexFlatL2 (exact), IndexIVFFlat (inverted file), and IndexHNSWFlat (hierarchical navigable small world). For GPU testing, a CPU-built index is transferred to a GPU resource for accelerated search operations. Note that, to isolate the retrieval stage, all test runs for this scenario utilize identical, pre-computed, and cached document embeddings.

Reproducibility Measurement Protocol. For each index configuration, the system performs five independent runs. Each run consists of building a FAISS index from the cached embeddings and then executing a standard set of queries against it. For GPU-specific tests, this involves transferring the index to the GPU before querying. We measure the end-to-end retrieval consistency using our core metrics (EMR, Jaccard Similarity, Kendall's Tau). In addition, for clustering-based indices like IVF, we perform a **centroid stability analysis** by measuring the L2 distance between the cluster centroids generated in different runs.

4.5.2 Results and Discussion. The results of our retrieval uncertainty evaluation were definitive and surprising: we observed perfect reproducibility across every tested configuration.

Uniform Stability Across All Index Types. As shown in Table 6, all tested index types—from brute-force Flat_L2 to complex approximate methods like HNSW—achieved a perfect 1.000 score on all reproducibility metrics when run on the CPU. The Exact Match Rate, Jaccard Similarity, and Kendall's Tau were all perfect, indicating that identical document rankings were produced in every run. This finding demonstrates that with modern libraries like FAISS and under a strictly controlled environment (i.e., with fixed random

seeds), the indexing and CPU search algorithms themselves behave in a perfectly reproducible manner.

Table 6: Retrieval Reproducibility Across Various FAISS Index Types on CPU. All configurations demonstrated perfect stability.

Index Type	Exact Match	Jaccard	Kendall Tau
Flat_L2	1.00	1.00	1.00
Flat_IP	1.00	1.00	1.00
IVF	1.00	1.00	1.00
HNSW_accurate	1.00	1.00	1.00
HNSW_fast	1.00	1.00	1.00
LSH	1.00	1.00	1.00

GPU Search Operations Had No Impact on Reproducibility. Furthermore, our targeted tests of GPU-accelerated search yielded the same result. After transferring a deterministically built index to the GPU, the search results remained perfectly reproducible. Toggling determinism controls for GPU operations during the search phase had no impact on the final retrieved document list; all runs remained identical.

Discussion. The key insight from this scenario is that the FAISS retrieval stage—both indexing and search on CPU or GPU—when isolated and properly controlled, is not a significant source of non-determinism. This is a crucial finding for practitioners. It implies that when debugging reproducibility issues in a RAG system, the core FAISS operations are unlikely to be the culprit, provided that environmental factors like random seeds are managed. This shifts the focus of concern towards other parts of the pipeline, such as the embedding drift we quantified in the previous section, data versioning, or inconsistencies in the broader software environment. Our framework, by allowing us to isolate this component, has successfully falsified a common hypothesis about a primary source of system instability.

4.6 Scenario 5: Distributed Uncertainty

4.6.1 Goal and Methodology. This scenario evaluates reproducibility in a multi-node HPC environment, the setting where non-determinism is often most pronounced due to network timing, race conditions, and distributed coordination challenges. The goal is to measure the end-to-end consistency of our distributed retrieval protocol.

System Architecture and Protocol. Our distributed system implements a shared-nothing, scatter-gather architecture using MPI for inter-node communication. The key phases of the protocol are:

- (1) **Document Sharding:** The document collection is partitioned across all available nodes. We test three distinct distribution strategies: **hash-based** (using $\text{hash}(\text{doc_id}) \% \text{num_nodes}$ for deterministic balance), **range-based** (sequential blocks per node), and **random** (with a fixed seed for run-to-run consistency).
- (2) **Parallel Indexing:** Each node builds its local FAISS index independently using only its assigned document shard. This

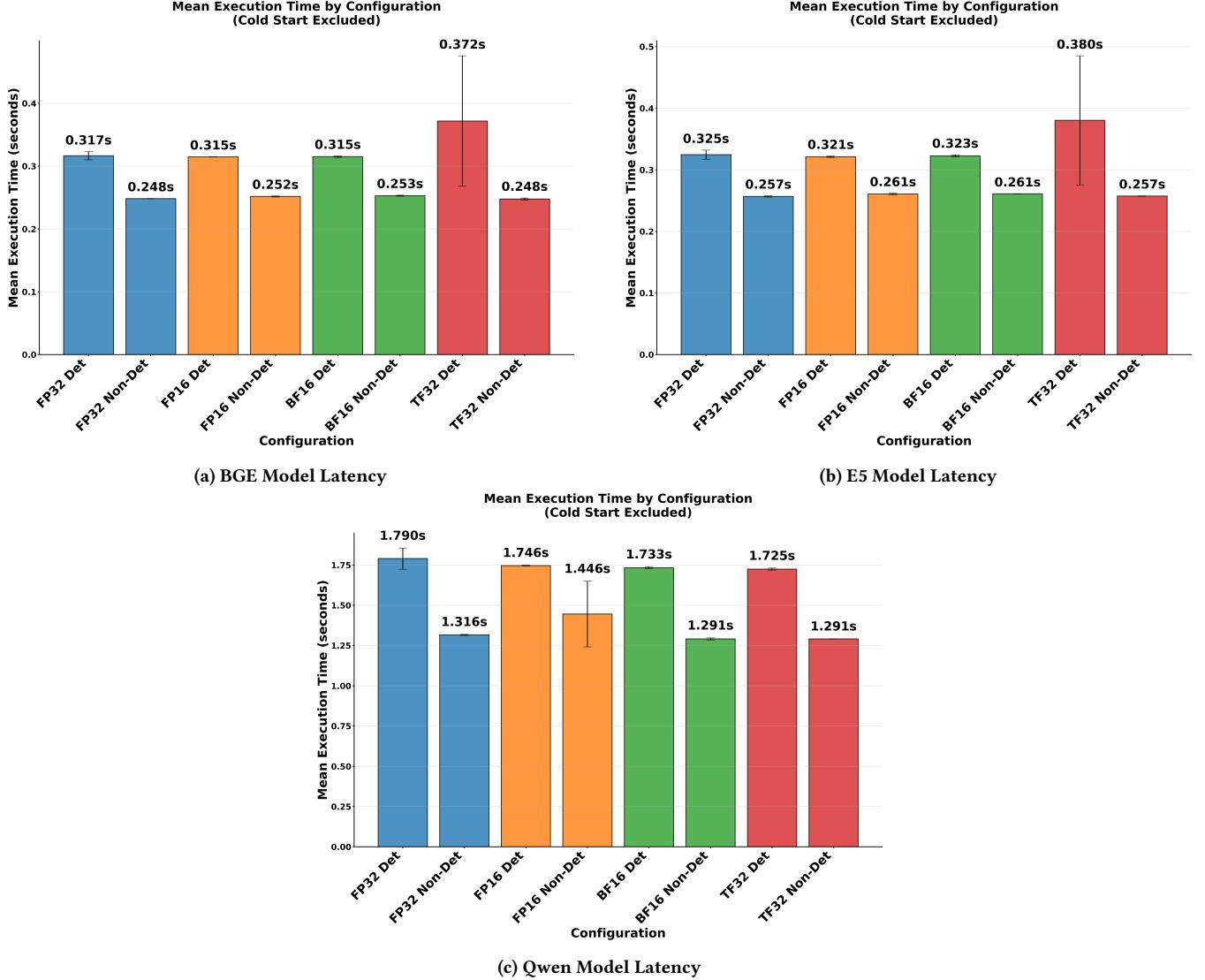


Figure 4: Embedding generation latency for each model across all 8 configurations, spanning the full page width for clarity. Lower is better. The non-deterministic mode is consistently faster across all models and precisions.

phase is “embarrassingly parallel” and requires no inter-node communication. An MPI barrier ensures all nodes complete indexing before proceeding.

- (3) **Distributed Search and Aggregation:** A query is broadcast to all nodes. Each node searches its local index in parallel. The top-k candidate results from all nodes are gathered to a root process (rank 0).
- (4) **Global Aggregation:** The root process performs a global merge of all candidate documents, re-ranks them based on their true similarity scores (e.g., L2 distance), and selects the final global top-k results.

Experimental Configurations. We conducted these tests on a 4-node cluster within TAMU ACES. We evaluated the reproducibility

of the distributed protocol across all the FAISS index types previously tested in the single-node scenario.

4.6.2 Results and Discussion. Across all tested configurations, the distributed retrieval protocol demonstrated perfect, 100% reproducibility.

Perfect Consistency Across All Factors. As summarized in Table 7, every combination of index type and sharding strategy yielded perfect scores on all our reproducibility metrics. The Exact Match Rate, Jaccard Similarity, and Kendall’s Tau were all 1.000, indicating that identical, identically-ranked results were returned in every run. This result held true for both simple indexes like Flat_L2 and complex approximate indexes like HNSW.

Table 7: Distributed Retrieval Reproducibility (4 Nodes). For all metrics, a score of 1.0 indicates perfect reproducibility (the best possible score), while 0.0 would indicate no overlap or correlation (the worst possible score).

Index Type	Sharding Type	Exact Match	Jaccard	Kendall Tau
Flat_L2	Hash	1.00	1.00	1.00
IVF	Hash	1.00	1.00	1.00
HNSW	Hash	1.00	1.00	1.00
LSH	Hash	1.00	1.00	1.00
Flat_L2	Range	1.00	1.00	1.00
IVF	Range	1.00	1.00	1.00
HNSW	Range	1.00	1.00	1.00
LSH	Range	1.00	1.00	1.00
Flat_L2	Random	1.00	1.00	1.00
IVF	Random	1.00	1.00	1.00
HNSW	Random	1.00	1.00	1.00
LSH	Random	1.00	1.00	1.00

Reproducibility by Design. This perfect stability is not accidental but a direct result of the protocol’s design. The key factors ensuring reproducibility are:

- **Deterministic Sharding:** Using hash-based or seeded-random sharding ensures that every node receives the exact same subset of documents on every run.
- **MPI Synchronization:** The use of MPI barriers guarantees that all nodes complete critical phases (like indexing) before any node can advance, eliminating temporal race conditions.
- **Centralized Aggregation:** The gather-and-aggregate step on a single root node ensures that the final global ranking logic is performed in a serial, deterministic manner.

Discussion. The key insight from this scenario is that the complexity of distributed operations does not inherently lead to non-determinism, provided the communication and data partitioning protocol is carefully designed for reproducibility. While the system has typical distributed computing trade-offs—such as network overhead during the gather phase and a temporary memory concentration on the root node—our results prove that it does not sacrifice consistency. This validation is critical for deploying RAG systems in production scientific environments where bit-for-bit identical results are often required. Our framework has successfully demonstrated that this distributed architecture is suitable for such high-stakes applications.

5 Conclusion

This paper introduced ReproRAG, a comprehensive framework for benchmarking the reproducibility of RAG retrieval systems. Our empirical evaluation, conducted across a series of controlled scenarios, not only quantifies sources of non-determinism but also challenges several common assumptions held by the community.

Our findings reveal a clear hierarchy of reproducibility challenges. While data insertion causes predictable result displacement (Scenario 1), and precision choice causes measurable drift (Scenario 2), we found that the core indexing/search algorithms (Scenario

3) and distributed protocols (Scenario 4) are, under proper control, perfectly stable on a run-to-run basis. This hierarchy suggests that efforts to improve RAG reproducibility should focus less on the algorithms themselves and more on data versioning, precision management, and strategies for handling dynamic data.

A key contribution of this work is the empirical falsification of the common hypothesis that approximate nearest neighbor algorithms are a primary source of run-to-run randomness. Similarly, we demonstrated that common software flags (`cudnn.deterministic`) may have no observable effect on the output of modern embedding models, underscoring the necessity of empirical validation over reliance on framework settings.

In conclusion, this work provides both the tools and the empirical evidence to move the discussion of RAG reproducibility from anecdote to quantitative measurement. The open-source ReproRAG framework empowers users to validate their own systems and make informed architectural decisions. Future work will involve extending the framework to other popular vector databases and integrating it into CI/CD pipelines for the continuous validation of trustworthy AI in science.

Acknowledgments

This work is based upon support of the Summer of Reproducibility, a program funded by the US National Science Foundation under Grant No. 2226407. This work used TAMU ACES at Texas A&M University through allocation CHE240191 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants # 2138259, # 2138286, # 2138307, # 2137603 and # 2138296. This research is supported in part by the U.S. Department of Energy (DOE) through the Office of Advanced Scientific Computing Research’s “Orchestration for Distributed & Data-Intensive Scientific Exploration” and the “Cloud, HPC, and Edge for Science and Security” LDRD at Pacific Northwest National Laboratory. PNNL is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

References

- [1] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2021. ann-benchmarks.com. <http://ann-benchmarks.com>.
- [2] Daniel Campos, Tri Nguyen, Maxim Rosenberg, Xiaofei Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and L. Deng. 2016. MS MARCO: A Human Generated MAchine READING Comprehension Dataset. In *Proceedings of the NIPS 2016 Workshop on End-to-end Learning for Speech and Language Processing*.
- [3] Christian Collberg and Todd A. Proebsting. 2016. Repeatability and Benefaction in Computer Systems Research. In *Proceedings of the 2016 Annual CGO Conference*. 92–103. <https://doi.org/10.1145/2854038.2854049>
- [4] David L. Donoho, Arian Maleki, Inam Ur Rahman, Morteza Shahram, and Victoria Stodden. 2017. 50 Years of Data Science. *Journal of Computational and Graphical Statistics* 26, 4 (2017), 745–766. <https://doi.org/10.1080/10618600.2017.1384734>
- [5] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*. 518–529.
- [6] Emily Herron, Junqi Yin, and Feiyi Wang. 2024. SciTrust: Evaluating the Trustworthiness of Large Language Models for Science. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. <https://doi.org/10.1109/SCW63240.2024.00017>
- [7] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. In *IEEE transactions on pattern analysis and machine intelligence*, Vol. 33. IEEE, 117–128.
- [8] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.
- [9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. In *IEEE Transactions on Big Data*, Vol. 7. IEEE, 535–547.

- [10] Maurice G. Kendall. 1970. *Rank Correlation Methods* (4th ed.). Griffin.
- [11] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets* (2nd ed.). Cambridge University Press.
- [12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Timo Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. 9459–9474.
- [13] Yu A. Malkov and Dmitry A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. In *IEEE transactions on pattern analysis and machine intelligence*, Vol. 42. IEEE, 824–836.
- [14] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- [15] Peter Mattson, Vijay Janapa Reddi, Christine Cheng, Cody Coleman, Greg Diamos, David Kanter, Paulius Micikevicius, David Patterson, Guenther Schmuelling, Hanlin Tang, Gu-Yeon Wei, and Carole-Jean Wu. 2020. MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance. *IEEE Micro* 40, 2 (2020), 8–16. <https://doi.org/10.1109/MM.2020.2974843>
- [16] Vaishnavh Nagarajan, Adarsh Baddepudi, Vivek Gopinath, Anant Kumar, Sakshi Singla, Priyansh Goyal, and Kurt Keutzer. 2018. The sources of irreproducibility in deep learning: An empirical study. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.
- [17] Linyi Sun, Yifei Huang, Haohan He, Xinyue Chen, Jialing Gao, Yang Liu, et al. 2024. TrustLLM: Trustworthiness in Large Language Models. *arXiv preprint arXiv:2401.05561* (2024).
- [18] Qwen Team. 2025. Qwen3 Technical Report. *arXiv:2505.09388 [cs.CL]* <https://arxiv.org/abs/2505.09388>
- [19] Texas A&M University. 2024. ACES (Accelerating Computing for Emerging Sciences). <https://hprc.tamu.edu/aces/> Accessed: 2024-07-31.
- [20] Bo Wang, Yu Zhang, Liu Liu, and et al. 2024. A Survey on Large Language Models for Scientific Discovery. (2024). *arXiv:2402.10175 [cs.CL]*
- [21] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv:2212.03533 [cs.CL]*
- [22] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)* 28, 4 (2010), 1–38.
- [23] Shitao Xiao, Zheng Liu, Peitian Wang, Fan Meng, Xing Yuan, Hong-Cheu Wang, Zhi-Jie Liu, Jia-Chen Wu, Zhi-Hao Wang, Jian Wu, et al. 2023. C-Pack: A General Framework for Unifying and Improving General-Purpose Text Embeddings. *arXiv:2309.07597 [cs.CL]*