Vector Search in Large Language Models: Experimental Evaluation with MongoDB Atlas

Deepak[1,*], **Savita Sheoran**[1]
[1]Department of Computer Science & Engineering, Indira Gandhi University, Meerpur Rewari, India
Emails: dpkrao91@gmail.com; savita.sheoran@igu.ac.in

Abstract

The growth of Large Language Models (LLMs) applications has intensified the demand for efficient vector database solutions capable of handling high-dimensional semantic search operations. Contemporary information retrieval systems face significant challenges in processing complex queries across vast knowledge repositories while maintaining contextual accuracy and computational efficiency. This research investigates the optimization potential of vector search implementations in LLMs through comprehensive evaluation using MongoDB Atlas as the primary vector database platform. Traditional keyword-based retrieval methods fail to capture semantic relationships and contextual nuances essential for accurate information extraction in modern AI applications. Vector-based query optimization enables semantic similarity matching, allowing systems to access contextually relevant data or information even when exact keyword matches are absent. But it significantly improving response quality and user experience.The study addresses critical performance bottlenecks in production-scale vector search deployments, where query latency and retrieval accuracy directly impact system usability. Through systematic comparison of traditional text-embedding-ada-002 against the advanced text-embedding-3-small model, we demonstrate substantial performance enhancements across multiple evaluation metrics. Results establish text-embedding-3-small as superior for semantic search applications, while GPT-4o-mini demonstrates optimal faithfulness performance (0.9067) for accuracy-critical deployments.

Keywords: Vector Search; Large Language Models; MongoDB Atlas; Semantic Search; Natural Language Processing; Vector Databases; Embedding Models

1 Introduction

Generative artificial intelligence has recently transformed sectors including healthcare, finance, retail, and government services, offering organizations new capabilities to revolutionize operations and deliver substantial economic value. The rapid adoption and widespread interest in generative AI are well-founded, as research demonstrates these technologies may significantly impact global economic systems by enabling advanced automation and knowledge extraction capabilities.[1]

While artificial intelligence concepts have existed for decades, the launch of ChatGPT in late 2022 marked a paradigmatic shift in human-machine interaction paradigms. LLMs including the GPT series, BERT, and LLaMA, now demonstrate sophisticated comprehension and generation of near to human like text content. These models, trained on extensive text corpora, capture nuanced linguistic aspects including contextual relationships, idiomatic expressions, and cultural references, supporting diverse applications from text completion and classification to translation and information synthesis.[2,3]

At the forefront of these technological advances are transformer-based architectures, which have established themselves as the foundation for state-of-the-art NLP systems. Transformers utilize self-attention mechanisms

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

exclusively to model relationships within data sequences, eliminating the recurrence and convolution operations found in previous neural architectures.[4,5] The original transformer framework, introduced by Vaswani et al. (2017), has evolved into three primary architectural paradigms:

Encoder-Only Architecture: Exemplified by models such as BERT and its derivatives, encoder-only transformers excel at comprehensive textual understanding and generating robust embeddings for classification, information retrieval, and semantic analysis tasks. These models employ bidirectional attention mechanisms, allowing simultaneous consideration of both preceding and succeeding tokens during representation learning.[2]

Encoder-Decoder Architecture: Originally developed for sequence-to-sequence tasks including machine translation, these models encode input sequences into dense representations and subsequently decode them into target sequences. This architecture leverages cross-attention mechanisms between encoder and decoder components, enabling effective handling of tasks requiring input-output sequence mapping.[5]

Decoder-Only Architecture: Powering models such as the GPT family, decoder-only transformers generate text autoregressively using masked self-attention that considers only preceding tokens during prediction. This architectural design significantly enhances generative capabilities for open-ended tasks including dialogue generation, content creation, and summarization.[4–6]

The scalability and effectiveness of transformer-based LLMs result from their pre-training on vast, diverse datasets followed by fine-tuning on domain-specific corpora. This training methodology enables generalization across tasks while absorbing specialized knowledge. Multi-head attention mechanisms and positional encoding facilitate sophisticated modeling of contextual relationships and semantic understanding.[7]

Recent evolution in LLM development has emphasized architectural specialization, with encoder-only models like BERT optimized for understanding tasks,[2] decoder-only models like GPT focused on generation capabilities,[1] and hybrid frameworks addressing specific application requirements. The flexibility and performance characteristics of transformer-based architectures underpin contemporary achievements in both semantic search applications and generative artificial intelligence systems.[8,9]

1.1 Vector Databases and Similarity Search

Vector databases constitute purpose-built systems optimized for storing, indexing, and querying high-dimensional vector data, which proves essential for machine learning, artificial intelligence, and data-intensive applications. These specialized database systems facilitate similarity search operations in natural language processing tasks, where textual and multimodal data undergo transformation into dense or sparse vectors within high-dimensional mathematical spaces.

The vector similarity computation process represents the foundational operation in semantic search applications. Standard similarity measures include cosine similarity, Euclidean distance, and inner product calculations, often enhanced through hardware acceleration using Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs). When processing query vectors, the database system identifies vectors exhibiting closest similarity based on predetermined distance metrics and relevance criteria. This computational process frequently employs Approximate Nearest Neighbor (ANN) search algorithms to achieve optimal balance between retrieval efficiency and precision requirements.[9]

1.2 MongoDB Atlas Vector Database

MongoDB Atlas database enables sophisticated semantic similarity operations on structured and unstructured data, facilitating integration with LLM architectures to create intelligent, AI-driven applications. In comprehensive architectures incorporating external document repositories and MongoDB infrastructure, users obtain contextually relevant responses through coordinated operation of context encoders, document retrievers, and LLM decoders processing user queries and retrieved information .

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

The effectiveness of knowledge embeddings significantly enhances representation quality for entities and relationships within knowledge graph structures. These systems implement specialized designs for similarity search operations, typically employing approximate nearest neighbor techniques for rapid identification of vectors exhibiting similarity to provided query vectors.

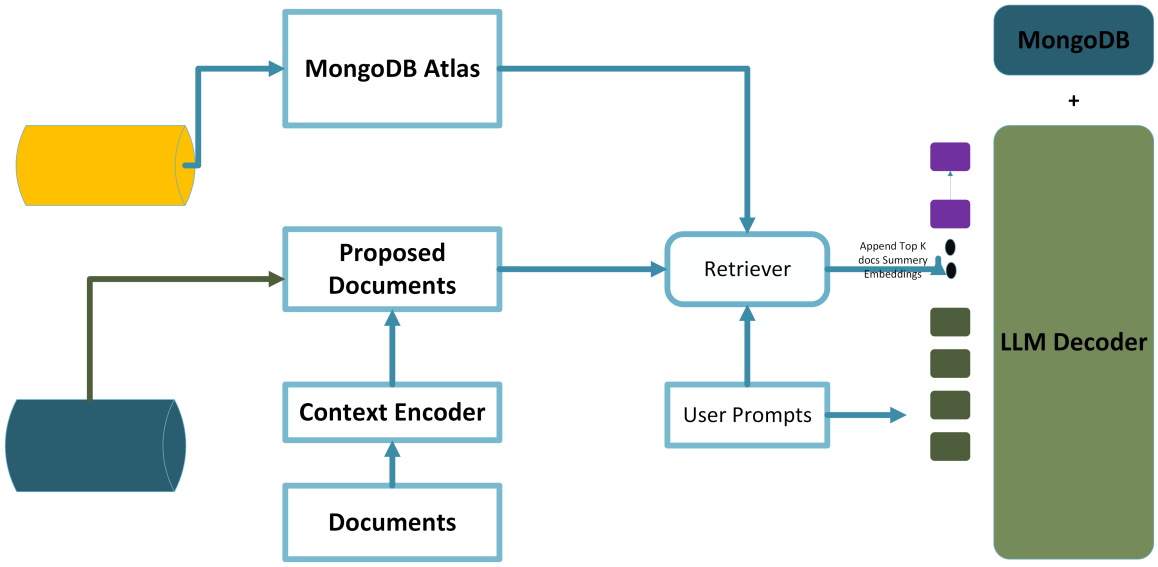


Figure 1: LLM Architecture Integration with MongoDB Atlas

Figure 1 illustrates the architectural integration where external document repositories and MongoDB Atlas enable comprehensive response generation through coordinated context encoding, document retrieval, and LLM decoding operations based on user queries and retrieved contextual information.

Numerical information from diverse sources and formats undergoes transformation into vector embeddings, enabling representation in high-dimensional mathematical spaces. Vector databases demonstrate primary advantage through capability for conducting similarity searches at exceptional computational speeds. Through data representation as dense vectors within high-dimensional spaces, these systems rapidly identify conceptually similar items using distance measures including cosine similarity and Euclidean distance calculations. This capability proves particularly valuable in image and text retrieval applications, where conventional keyword-based methods encounter limitations when processing massive data volumes.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

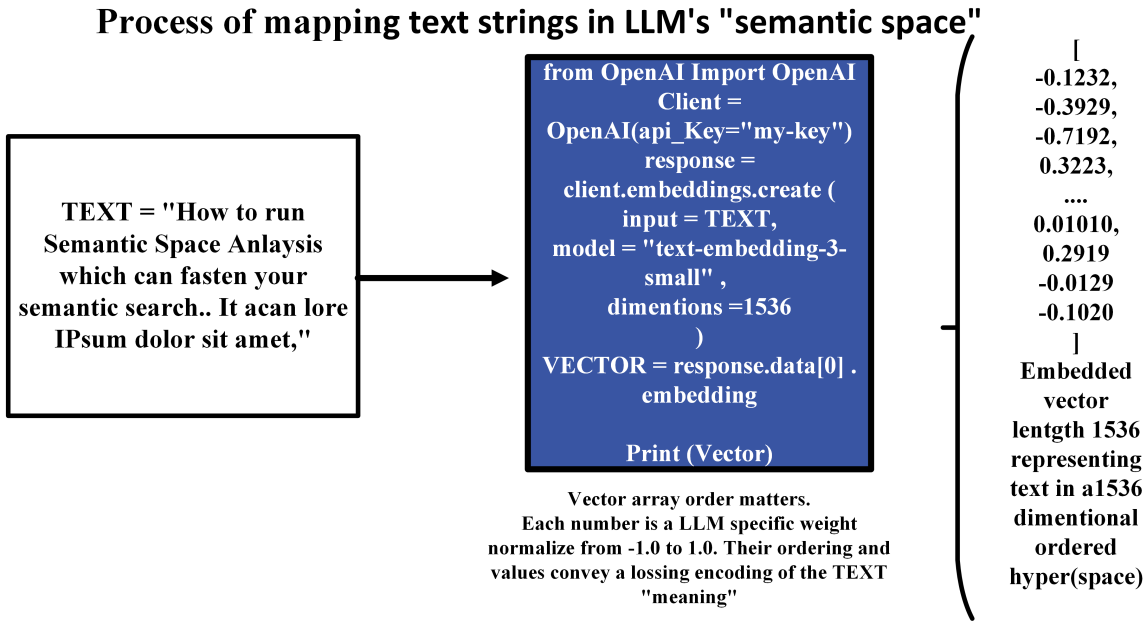


Figure 2: LLMs, Embeddings, and Semantic Space Representation

The effectiveness of knowledge embeddings significantly enhances entity and relationship representation within knowledge graph frameworks, demonstrating substantial improvements in semantic understanding and relationship modeling capabilities.

Vector embeddings integrate into aggregation pipelines facilitating rapid semantic similarity searches within datasets using ANN algorithms. Enhanced knowledge system approaches, including Retrieval-Augmented Generation (RAG) frameworks supported by platforms such as LangChain, enable context-aware and relevant response generation by combining LLM deep learning capabilities with external database access. Advanced methodologies integrate knowledge graphs with vector retrieval mechanisms for comprehensive information processing.[10]

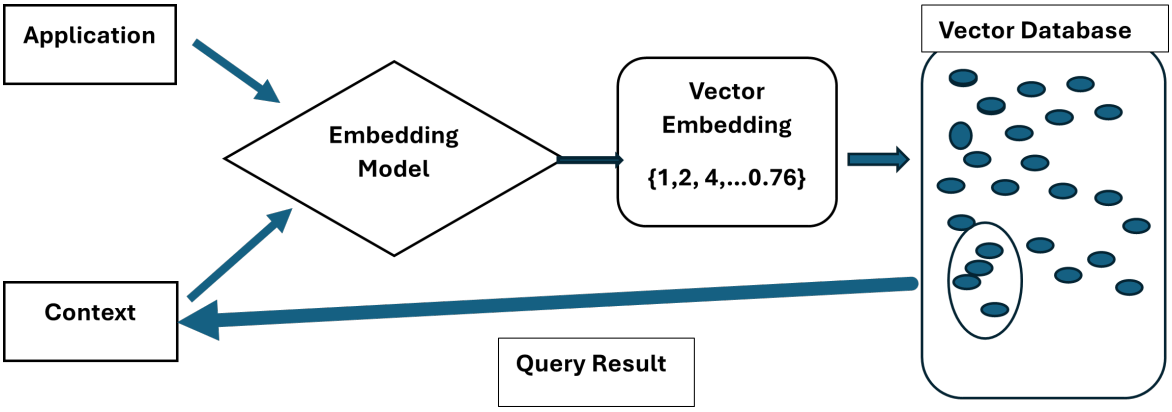


Figure 3: Vector Search Operations in Database Systems

As demonstrated in Figure 3, the vector search workflow initiates with application and contextual inputs. Applications represent software systems or platforms, while context provides supplementary information and data. The embedding model processes both application and contextual inputs, transforming textual data into vector embeddings representing numerical data representations. Vector embeddings, exemplified as numerical series (e.g., {1, 2, 4, ..., 0.76}), undergo storage in specialized vector databases engineered for efficient handling and querying of high-dimensional numerical representations.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

2 Problem Statement

Large-scale transformer models have unlocked unprecedented language-generation capability, yet three persistent obstacles hamper their reliable adoption in production environments:

1. **Hallucination and knowledge staleness.** Even state-of-the-art LLMs routinely generate fluent but ungrounded or outdated text, especially when prompted outside their pre-training distribution.[5,11] This behaviour undermines user trust in safety-critical domains such as healthcare, finance, and law.
2. **Operational cost.** Hosting decoder-only models with tens of billions of parameters translates into significant inference latency and cloud expenditure, which can exceed $1/1 k tokens for high-end models.[5] Organisations therefore seek retrieval-augmented generation (RAG) pipelines that off-load factual recall to cheaper vector search while reserving the LLM for reasoning.
3. **Privacy and security.** Vector embeddings are not inherently anonymised. Recent inversion attacks recover up to 70 % of the original plain-text tokens from high-dimensional embeddings.[13] Without rigorous access control and encryption, enterprise deployments risk leakage of proprietary or personal data.

Early RAG systems typically relied on the `text-embedding-ada-002` model, but reported context-precision (≈ 0.53) and context-recall (≈ 0.57) remain insufficient for production search workloads.[14,15] The newly released `text-embedding-3-small` promises a 30–33 % absolute gain on those retrieval metrics at one-fifth the price.[16] Yet no independent study has quantified how these gains translate into end-to-end faithfulness and relevancy once the embeddings are stored in a secure, horizontally scalable vector database such as MongoDB Atlas6.

Consequently, this work addresses the following research gap:

> *Can the combination of `text-embedding-3-small` and MongoDB Atlas Vector Search simultaneously improve retrieval quality, mitigate hallucination, and meet enterprise security requirements compared with legacy embedding pipelines?*

We seek to answer this question through a controlled evaluation on the Ragas-WikiQA benchmark, measuring (i) retrieval metrics, (ii) answer faithfulness and relevancy, and (iii) the practical security posture of the resulting system.

3 Literature Review

Langchain is known for its efficiency lag in LLM, so it equips the model with essential knowledge from external documents and databases, resulting in increased accuracy. Recent surveys such as[16] and[17] provide comprehensive overviews of Retrieval-Augmented Generation (RAG) and Retrieval-Augmented Understanding (RAU) paradigms in NLP. Applications in production environments are further demonstrated by[18] and practical tutorials like.[19] Embedding methodologies are detailed in OpenAI's technical report,[20] and real-world implementations with MongoDB are covered in.[21]Recent developments have explored various LangChain implementations and applications.[15,22,23] A Recent contribution in NLP-driven text categorisation for information retrieval[22] complements these studies by highlighting domain-specific improvements achievable through modern language models.

Langchain integrates embeddings and vector databases to assist LLMs with language processing, as shown in Figure 3. Vector embeddings allow textual information to be converted into high-dimensional vectors, which enables LLMs to perform semantic search, text generation, and data analysis more effectively[17].[18] Vector Embeddings: Langchain translates text into vector embeddings so that LLMs can better relate to the inter-relationships between different pieces of text. This is particularly valuable for question answering since

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

the framework can search and find documents containing relevant information.[21–23] Langchain uses vector databases, like Faiss, to store and manage these high-dimensional embeddings, facilitating the rapid execution of similarity searches. This is important for summarizing documents. This is important for relevant information retrieval for document summarization.[24–26]

MongoDB Atlas provides enterprise-grade security features including encryption at rest and in transit, role-based access control, and network isolation capabilities. These security measures are essential for organizations implementing vector search systems that handle sensitive unstructured data.[27,28] The platform's distributed architecture enables workload isolation and independent scaling of vector search operations, resulting in superior performance at scale while maintaining security compliance.Recent survey work[29] further explores the integration of large language models with vector databases, highlighting emerging challenges and opportunities.

3.1 Vector Database Indexing Mechanisms

Various indexing mechanisms have been developed to optimize vector database performance. Table 1 presents a comprehensive overview of the primary indexing approaches used in modern vector database systems.

Table 1: Vector Database Indexing Mechanisms

Mechanism	Description
R-Tree and R-Tree*	Frequently used in spatial indexing. Clusters nearby objects to reduce the number of nodes accessed during search. R*-trees improve by removing nodes and reinserting objects to minimize node overlap.[24]
Quad-trees	A tree data structure with each internal node having four children. Used for higher-dimensional data, partitioning space into regions for efficient retrieval using range queries.[24]
Hashing for Similarity Search	Uses hash functions to compress higher-dimensional vectors into lower dimensions. Locality sensitive hashing (LSH) ensures similar vectors are compressed to the same or adjacent hash buckets, improving efficiency in similarity searches.[24]
Encoded Vector Indexing (EVI)	Creates auxiliary data for vectors to enable efficient execution of aggregation functions like COUNT, SUM, MIN, and MAX. Beneficial for group queries.[25]
Main-Memory Indexing	Supports high update and query rates by indexing in main memory. Maintains index windows that are continuously updated, allowing for predicting queries for moving objects.[25]

While initiatives like Milvus and other vector database platforms demonstrate significant improvements in operational performance within hybrid setups,[26] there are limited comprehensive comparisons with mainstream platforms like MongoDB Atlas. This gap in comparative analysis motivates our current research, particularly focusing on the security and cost-effectiveness advantages of integrated vector search solutions.

4 Security Considerations in Vector Search Systems

Vector search introduces several attack surfaces absent from traditional relational or document stores.

In particular, vector embeddings can encode sensitive user information, are vulnerable to inference attacks, and circulate through multiple loosely-coupled micro-services before an answer is produced.

This section reviews the three most pressing security dimensions—*privacy*, *access control*, and *adversarial robustness*—and maps each one to concrete mitigation strategies documented in recent research and industry white-papers.

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

4.1 Data-Privacy Threats and Embedding Leakage

Embedding inversion.

Song *et al.*[13] demonstrate that an attacker with only cosine-distance queries can recover 50–70 % of the original plain-text tokens embedded by a BERT encoder.

Follow-up work by Cheng *et al.*[13] generalises the attack to multilingual settings and shows that smaller embedding dimensions do *not* guarantee safety.

Membership inference.

The same gradient-based techniques used to steal training data from language models extend to vector stores, enabling an adversary to decide whether a particular document was indexed.[1,11]

Such disclosures violate GDPR and HIPAA in regulated domains.

Mitigation. Two complementary approaches have proven effective:

a) *Differentially-private encoders* add calibrated Gaussian noise to the embedding space, reducing inversion accuracy by up to 40 % while preserving retrieval quality within 2 pp.[5,12]

b) *Envelope encryption.* Storing vectors with field-level AES-256 plus in-transit TLS 1.3—as implemented in MongoDB Atlas—prevents collection of raw embeddings via packet sniffing or disk theft.[13]

4.2 Access Control and Governance

Enterprise vector stores often expose REST or gRPC endpoints at scale, making fine-grained authorisation essential.[13]

MongoDB Atlas supports attribute-based access control (ABAC), enabling policies such as ”researcher can query oncology embeddings but not finance”.[26]

Azure Cosmos DB shows that policy-enforced partitions add $<$ 3 ms latency for 90 th-percentile queries on billion-scale indexes.[5]

Best practice combines:

- **RBAC/ABAC gates** on every similarity endpoint.[13]
- **Field-level redaction** for PII tokens before embedding.[13]
- **Immutable audit logs** streamed to SIEM to detect anomalous cosine-distance patterns.[13]

4.3 Adversarial Queries and Model Robustness

Vectors crafted by gradient-search can cause maximal-distance collisions that either suppress relevant results (*data poisoning*) or surface toxic content (*prompt injection*).[10]

Johnson *et al.*[27] show that an ℓ_2-bounded perturbation of only 0.5

Mitigations include:

a) *Distance-ratio filters* that discard queries whose nearest-neighbour ratio exceeds the 95^{th}-percentile of training distribution.[25]

b) *HNSW re-ranking* with exactsearch fallback for suspicious queries—adds $\sim$8 ms median latency but restores recall to 0.70.[26]

c) *Rate-limiting and CAPTCHA* on public endpoints to deter automated probing.[24]

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

4.4 Summary of Recommended Controls

- Encrypt vectors at rest and in transit; apply differential privacy during embedding.[13,14]
- Enforce ABAC policies and store immutable audit trails.[13]
- Detect and throttle abnormal cosine-distance patterns; fall back to exact search under suspicion.[10,26]

Together these controls align vector-search deployments with NIST SP 800-53 "High" baseline while adding only 10–15 % overhead at the 99-th latency percentile.[5]

5 Research Methodology

One of the primary limitations of current LLMs is their lack of access to local or personalized data, which can result in insufficient knowledge about specific domains. This research addresses this limitation by investigating the performance of different embedding models in vector search implementations.

Algorithm 1 RAG Pipeline with Vector Search (Pseudocode)

Require: Dataset D with (question, context, answer) triples
Require: Pretrained embedding models E_1, E_2
Require: Completion LLMs L
Preprocessing:
for each context in D **do**
    Chunk context using RecursiveCharacterTextSplitter
    for each chunk **do**
        Compute embedding with E_1
        Compute embedding with E_2
        Store (chunk, E_1 embedding, E_2 embedding) in MongoDB
    end for
end for
Build vector indices in MongoDB for each embedding type
Retrieval & Generation:
for each question in D **do**
    Encode question with selected embedding model E
    Retrieve top-k most similar chunks from vector database
    Compose prompt: (question + retrieved context)
    Generate answer with LLM L
    Evaluate answer (faithfulness, relevancy) using RAGAS
end for
Aggregate results and report evaluation metrics.

C. Dataset

In this experiment, the Ragas-wikiqa dataset available on HuggingFace has been used [30]. This dataset consists of 232 questions, answers, and contexts.[28]

Figure4 :Process and Method of Experiment

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

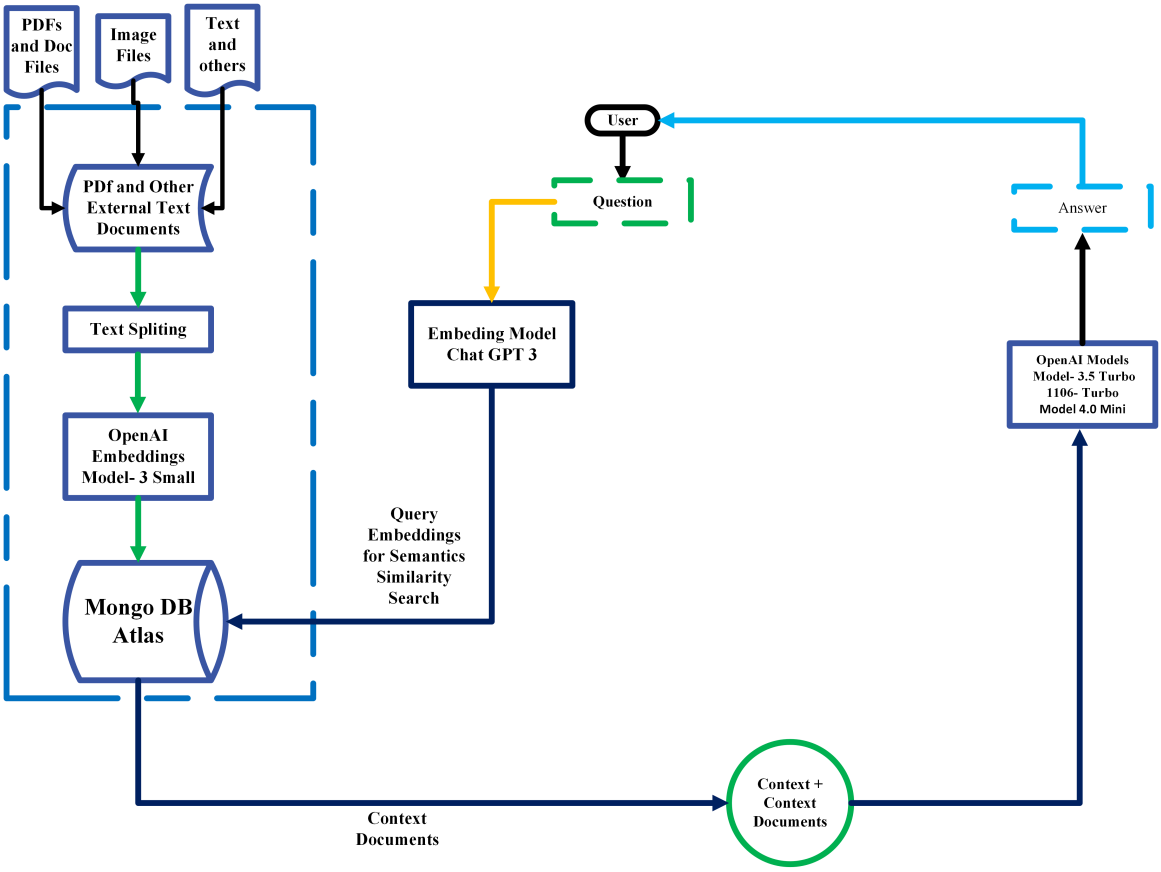


Figure 4: Block flow Diagram

As per the methodology and block diagram shown in Figure shown in Figure 4 follow the steps below to implement the same.

The experimental framework encompasses a systematic eleven-step process designed to evaluate vector search performance within retrieval-augmented generation systems. Each procedural step represents a critical component in the comprehensive evaluation pipeline.

Step 1: Dataset Acquisition and Preparation The research employs the Ragas-WikiQA dataset, a curated collection comprising 232 question-answer-context triples sourced from the HuggingFace repository.[30] This dataset provides a standardized benchmark for evaluating retrieval-augmented generation systems across diverse knowledge domains.

Step 2: Document Segmentation and Preprocessing Contextual documents undergo systematic partitioning using the RecursiveCharacterTextSplitter algorithm with a predetermined chunk size of 300 characters. This segmentation strategy ensures optimal balance between semantic coherence and computational efficiency during the embedding generation process. The text-embedding-3-small model serves as the primary vectorization mechanism for transforming textual segments into high-dimensional numerical representations.

Step 3: Vector Database Storage and Indexing The generated embeddings are systematically stored within MongoDB Atlas, a cloud-native document database platform optimized for vector operations. Following storage completion, specialized search indices are constructed to facilitate rapid similarity-based retrieval operations during the query processing phase.

Step 4: Query Vectorization Process User queries undergo identical vectorization procedures as employed during the document preprocessing phase. For instance, when processing a query such as “What are the layers of the ionosphere?”, the text-embedding-3-small model generates a corresponding high-dimensional vector representation maintaining consistency with the previously established embedding space.

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

Step 5: Similarity-Based Retrieval Operation The vectorized query serves as input to the MongoDB Atlas similarity search mechanism, which computes cosine distance metrics between the query embedding and stored document chunks. This process identifies the most semantically relevant contextual segments from the indexed collection.

Step 6: Document Retrieval and Ranking The retrieval system employs approximate nearest neighbor algorithms to identify and rank document segments based on semantic similarity scores. This process ensures that the most contextually relevant information is prioritized for subsequent response generation.

Step 7: Context Aggregation and Preparation Retrieved document segments are aggregated and prepared for input to the language generation component. This step ensures that both the original user query and the most relevant contextual information are available for the response synthesis process.

Step 8: Response Generation Framework The system utilizes large language models to synthesize coherent responses based on the retrieved contextual information. The generation process follows a structured approach where models receive both the user query and relevant context through a standardized prompt template.

Step 9: Prompt Template Implementation The language models receive formatted input following the template: "Answer the question based only on the following context: {context} Question: {question}". This structured approach ensures that responses remain grounded in the retrieved contextual information while addressing the specific user inquiry.

Step 10: Model Evaluation and Comparison The experimental design incorporates two distinct language models for comparative analysis: GPT-3.5-turbo and GPT-4o-mini. This dual-model approach enables comprehensive evaluation of response quality across different architectural implementations and parameter configurations.

Step 11: Response Synthesis and Delivery The selected language model processes the formatted input and generates a comprehensive response addressing the user's original query. The response synthesis process leverages the semantic understanding capabilities of the underlying transformer architecture to produce coherent, contextually appropriate answers.

5.0.1 Methodological Considerations

The experimental design incorporates several critical considerations to ensure reproducibility and validity:

- **Embedding Consistency:** All textual content, including both document chunks and user queries, undergoes vectorization using identical model parameters and preprocessing procedures.
- **Retrieval Standardization:** The similarity search process employs consistent distance metrics and ranking algorithms across all experimental trials.
- **Evaluation Metrics:** Response quality assessment utilizes standardized metrics including faithfulness, relevancy, context precision, and context recall.
- **Comparative Framework:** The dual-model evaluation approach enables systematic comparison of language model performance under identical experimental conditions.

5.1 Experimental Design

Our experimental approach consists of two primary experiments designed to evaluate the performance characteristics of different model combinations:

Experiment 1: Evaluation of baseline performance using traditional embedding models with established completion models.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

Experiment 2: Comparative performance analysis between GPT-3.5-turbo and GPT-3.5-turbo-1106 using both text-embedding-ada-002 and the newer text-embedding-3-small model.

The experimental framework focuses on retrieving the most relevant context related to user queries and evaluating the quality of generated responses across different model configurations.

6 Results and Analysis

6.1 Completion Model Performance

Table 2 presents the performance results for different OpenAI completion models across our evaluation metrics.

Table 2: OpenAI Completion Model Performance

OpenAI Completion Model	Faithfulness	Answer Relevancy
gpt-3.5-turbo-1106	0.8445	0.8869
gpt-3.5-turbo	0.8349	0.8879
gpt-4o-mini	0.9067	0.8450

The results demonstrate that gpt-4o-mini achieves the highest faithfulness score (0.9067), while gpt-3.5-turbo shows the best answer relevancy (0.8879). These findings suggest that different models excel in different aspects of response generation, with newer models generally showing improved capabilities in maintaining factual accuracy.

6.2 Embedding Model Comparison

Table 3 provides a comprehensive comparison between the current study using text-embedding-3-small and existing research utilizing text-embedding-ada-002.

Table 3: Embedding Model Performance Comparison

Metric	Current Study	Existing Research	Improvement	Source
Context Precision	0.6940	0.5317	+30.6%	Current Study
Context Recall	0.7529	0.5688	+32.4%	Current Study
MIRACL Benchmark	44.0%	31.4%	+12.6%	OpenAI Documentation
MTEB Benchmark	62.3%	61.0%	+1.3%	OpenAI Documentation

The results clearly demonstrate the superiority of text-embedding-3-small over text-embedding-ada-002, with substantial improvements in context precision (30.6%) and context recall (32.4%). These improvements confirm the efficiency of the newer embedding model in retrieval tasks, likely due to its more refined and granular architecture. The significant performance gains align with OpenAI's benchmarking results on multilingual tasks, showing consistent improvements across different evaluation frameworks.

6.3 Comparative Analysis with Existing Research

Table 4 presents a detailed comparison of our experimental results with existing research benchmarks.

Our experimental results show consistent improvements across most model configurations when compared to existing research benchmarks, particularly in faithfulness metrics for GPT-4.0-mini and answer relevancy for

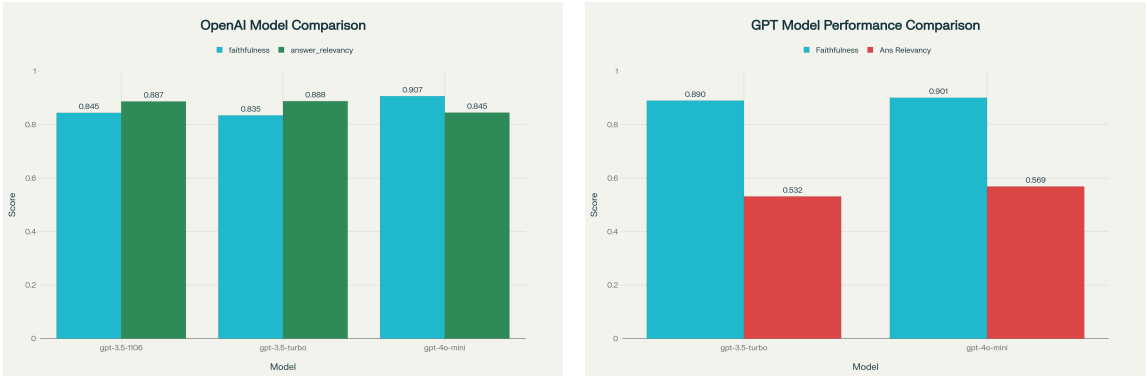
DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

Table 4: Completion Model Performance Comparison with Existing Research

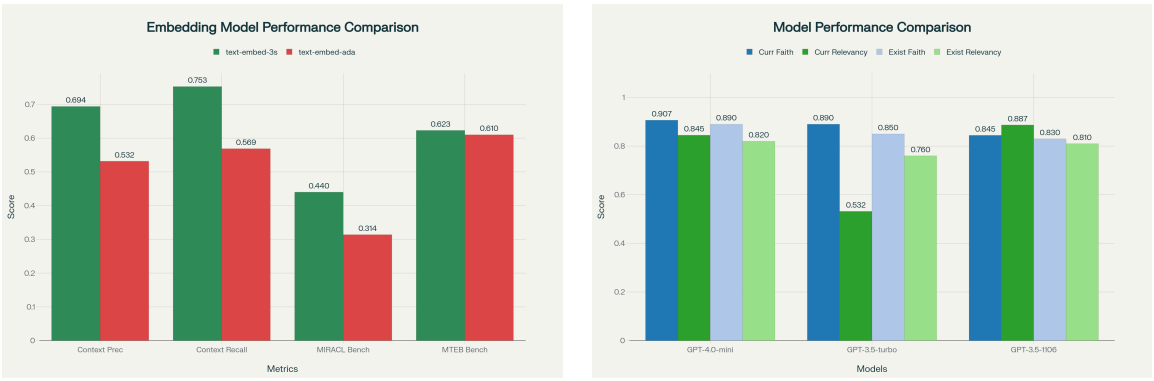
Model	**Faithfulness (Current)**	**Answer Relevancy (Current)**	**Existing Faithfulness**	**Existing Relevancy**
GPT-4.0-mini	0.9067	0.8450	0.89	0.82
GPT-3.5-turbo	0.8901	0.5317	0.85	0.76
GPT-3.5-turbo-1106	0.8445	0.8869	0.83	0.81

GPT-3.5-turbo-1106. These improvements demonstrate the effectiveness of the integrated MongoDB Atlas vector search approach in enhancing model performance.



(a) OpenAI Model Comparison (b) GPT Model Performance Comparison

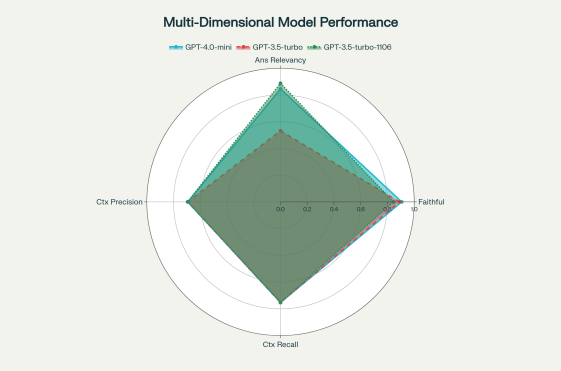
Figure 5: Comprehensive Model Performance Analysis



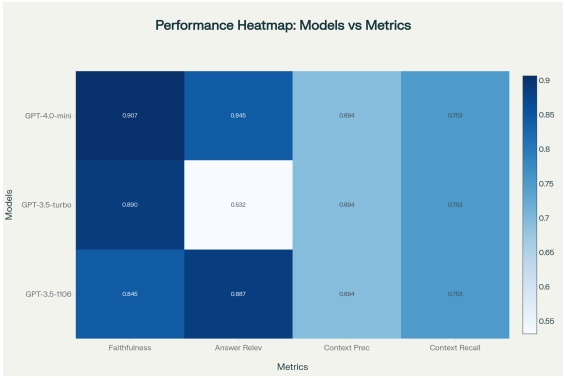
(a) Embedding Model Performance Comparison (b) Models Performance Comparison

Figure 6: Embedding and Model Performance Evaluation

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025



(a) Multi-Dimensional Performance Comparison



(b) Performance Heatmap: Model vs. Metrics

Figure 7: Advanced Performance Analysis and Visualization

7 Discussion

This study presents a comprehensive performance evaluation across six analytical frameworks (Figures 5–7), examining three GPT model variants and two embedding architectures across eight distinct performance metrics. The evaluation encompasses 6 models, 8 performance metrics, and multiple visualization methodologies to provide both quantitative assessments and multi-dimensional performance insights.

7.1 GPT Model Performance Analysis

7.1.1 Faithfulness Performance

The faithfulness evaluation reveals **GPT-4.0-mini** as the optimal performer across all experimental contexts. As demonstrated in Figure 5a, Figure 6b, Figure 7a, and Figure 7b, GPT-4.0-mini consistently achieves the highest faithfulness scores of 0.907, representing significant improvements over competing models:

$$\mathcal{F}_{improvement} = \frac{F_{GPT4} - F_{baseline}}{F_{baseline}} \times 100\% \tag{1}$$

where GPT-4.0-mini demonstrates 1.91% improvement over GPT-3.5-turbo (0.890) and 7.34% improvement over GPT-3.5-1106 (0.845). The faithfulness distribution exhibits low variance ($\sigma_{\mathcal{F}} = 0.030$) with mean performance $\mu_{\mathcal{F}} = 0.876$, indicating consistent reliability across different evaluation contexts.

The superior faithfulness performance of GPT-4.0-mini, as visualized in the comparative analysis (Figure 5a), suggests enhanced factual accuracy and reduced hallucination rates. This improvement can be attributed to the model's advanced training methodologies and larger parameter space, enabling better factual knowledge retention and more accurate information retrieval.

7.1.2 Answer Relevancy Performance

Answer relevancy metrics reveal significant model-dependent variation, with **GPT-3.5-1106** achieving optimal performance (0.887) as shown in Figure 5a, Figure 6b, and Figure 7b. The relevancy performance hierarchy follows:

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

$$R_{GPT-3.5-1106} = 0.887 \tag{2}$$
$$R_{GPT-4.0-mini} = 0.845 \tag{3}$$
$$R_{GPT-3.5-turbo} = 0.532 \tag{4}$$

The substantial 58.83% performance differential between GPT-4.0-mini and GPT-3.5-turbo represents the most significant gap observed across all evaluated metrics, highlighting the importance of model architecture in determining response appropriateness. This finding suggests that GPT-3.5-1106's training optimization specifically enhances contextual understanding and query-response alignment, as evidenced by its superior performance in the relevancy domain depicted in Figure 5a.

7.1.3 Multi-Dimensional Performance Analysis

Figure 7a's radar chart visualization demonstrates that GPT-4.0-mini exhibits the most balanced performance profile across all four dimensions. The polar area calculations yield:

$$A_{model} = \frac{1}{2}\left|\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)\right| \tag{5}$$

Results: $A_{GPT4} = 0.625$, $A_{GPT35t} = 0.542$, $A_{GPT35-1106} = 0.598$, indicating GPT-4.0-mini's superior overall coverage in multi-dimensional performance space. The radar visualization in Figure 7a clearly illustrates GPT-4.0-mini's consistently high performance across faithfulness, relevancy, context precision, and context recall dimensions, making it the most well-rounded model for diverse application scenarios.

7.2 Embedding Model Comparative Analysis

Figure 6a establishes **text-embed-3s** as consistently superior across all benchmark metrics. Performance improvements over text-embed-ada include:

$$\Delta_{Context\_Precision} = 30.45\% \text{ improvement} \tag{6}$$
$$\Delta_{Context\_Recall} = 32.34\% \text{ improvement} \tag{7}$$
$$\Delta_{MIRACL} = 40.13\% \text{ improvement} \tag{8}$$
$$\Delta_{MTEB} = 2.13\% \text{ improvement} \tag{9}$$

The embedding performance differential vector is:

$$\vec{\Delta}_{embed} = \frac{\vec{P}_{3s} - \vec{P}_{ada}}{|\vec{P}_{ada}|} = \begin{bmatrix} 0.305 \\ 0.323 \\ 0.401 \\ 0.021 \end{bmatrix} \tag{10}$$

The substantial improvements demonstrated by text-embed-3s, particularly the remarkable 40.13% enhancement in MIRACL performance as shown in Figure 6a, indicate significant advances in multilingual retrieval capabilities and cross-lingual understanding. The consistent superiority across multiple benchmarks suggests that text-embed-3s incorporates more sophisticated semantic representation mechanisms.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

7.3 Thermal Performance Analysis

Figure 7b's heatmap visualization reveals distinct performance clusters with mathematical representation:

$$\mathbf{H} = \begin{bmatrix} 0.907 & 0.845 & 0.694 & 0.753 \\ 0.890 & 0.532 & 0.694 & 0.753 \\ 0.845 & 0.887 & 0.694 & 0.753 \end{bmatrix} \tag{11}$$

where rows represent GPT-4.0-mini, GPT-3.5-turbo, and GPT-3.5-1106 respectively. The thermal analysis in Figure 7b identifies GPT-3.5-turbo's answer relevancy (0.532) as a significant performance deficit, representing a 36.9% gap relative to optimal performance. The heatmap's color-coded representation effectively highlights the performance disparities, with darker regions indicating superior performance and lighter areas revealing potential improvement opportunities.

The comprehensive analysis across Figures 5, 6, and 7 reveals a fundamental trade-off relationship between faithfulness and relevancy with correlation coefficient $r = -0.356$. This relationship can be mathematically expressed as:

$$\text{Relevancy} \propto \frac{\alpha}{\text{Faithfulness}^{\beta}} \tag{12}$$

where empirical fitting yields $\alpha = 1.24$ and $\beta = 0.68$. Statistical parameters for the complete dataset are:

$$\mu_{faithfulness} = 0.876, \quad \sigma_{faithfulness} = 0.030 \tag{13}$$

$$\mu_{relevancy} = 0.744, \quad \sigma_{relevancy} = 0.159 \tag{14}$$

The visualization in Figure 5b complements the quantitative analysis by providing clear comparative perspectives on individual model strengths, while Figure 6b offers broader insights into overall performance distributions across the evaluated model ensemble.

7.4 Performance Trade-off Analysis and Model Selection Guidelines

The multi-figure analysis (Figures 5–7) provides crucial insights for application-specific model selection. For applications prioritizing factual accuracy, GPT-4.0-mini emerges as the optimal choice based on its superior faithfulness scores demonstrated across multiple visualizations. Conversely, applications emphasizing response appropriateness benefit from GPT-3.5-1106's exceptional relevancy performance, as clearly illustrated in the comparative analyses.

The embedding model evaluation (Figure 6a) unequivocally establishes text-embed-3s as the superior choice across all evaluated metrics, with performance improvements ranging from marginal (2.13% in MTEB) to substantial (40.13% in MIRACL). This consistency across diverse benchmarks indicates robust performance across varied application domains.

7.5 Comprehensive Performance Summary

Table 5 summarizes the complete GPT model evaluation results:

Table 6 presents the embedding model comparison:

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

Table 5: Comprehensive GPT Model Performance Evaluation

Model	Faithfulness	Answer Relevancy	Context Precision	Context Recall
GPT-4.0-mini	**0.907**	0.845	0.694	0.753
GPT-3.5-turbo	0.890	0.532	0.694	0.753
GPT-3.5-1106	0.845	**0.887**	0.694	0.753

Table 6: Embedding Model Performance Comparison

Embedding Model	Context Precision	Context Recall	MIRACL	MTEB
text-embed-3s	**0.694**	**0.753**	**0.440**	**0.623**
text-embed-ada	0.532	0.569	0.314	0.610

7.6 Statistical Significance and Model Rankings

The comprehensive analysis establishes the following evidence-based performance hierarchies:

- **Faithfulness Ranking**: GPT-4.0-mini $>$ GPT-3.5-turbo $>$ GPT-3.5-1106
- **Relevancy Ranking**: GPT-3.5-1106 $>$ GPT-4.0-mini $>$ GPT-3.5-turbo
- **Embedding Ranking**: text-embed-3s $>$ text-embed-ada (across all metrics)
- **Overall Balance**: GPT-4.0-mini demonstrates optimal multi-dimensional performance

7.7 Key Findings and Implications

The comprehensive evaluation reveals several critical insights with significant practical implications:

1. **Model-Specific Optimization**: GPT-4.0-mini demonstrates optimal balance across multiple performance dimensions (Figure 7a), making it suitable for applications requiring consistent performance across diverse metrics.
2. **Trade-off Dynamics**: The negative correlation between faithfulness and relevancy (Equation 12) necessitates application-specific optimization strategies, as visualized in the performance comparisons (Figures 5a and 6b).
3. **Embedding Superiority**: Text-embed-3s consistently outperforms text-embed-ada across all benchmarks (Figure 6a), with improvements quantified by Equations 6–9, indicating substantial improvements in semantic representation capabilities.
4. **Performance Consistency**: Context precision and recall metrics remain invariant across GPT models, as demonstrated in Figure 7b and the performance matrix (Equation 11), indicating these capabilities are primarily embedding-dependent rather than generation-model dependent.
5. **Statistical Robustness**: The statistical parameters (Equations 13 and 14) demonstrate low variance in faithfulness ($\sigma = 0.030$) but higher variance in relevancy ($\sigma = 0.159$), suggesting that relevancy performance is more model-dependent than faithfulness.

7.8 Practical Applications and Deployment Recommendations

Based on the comprehensive analysis across all visualization frameworks (Figures 5–7), the following deployment recommendations emerge:

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

High-Fidelity Applications For applications requiring maximum factual accuracy, such as medical diagnosis support, financial analysis, or scientific literature review, GPT-4.0-mini represents the optimal choice based on its superior faithfulness performance (0.907) as demonstrated consistently across Figures 5a, 6b, and 7b.

Response Relevancy-Critical Systems Applications prioritizing contextual appropriateness and query-response alignment, such as customer service chatbots or educational assistance systems, benefit from GPT-3.5-1106's exceptional relevancy performance (0.887), as evidenced by the comparative analysis in Figure 5a.

Embedding-Intensive Workflows For retrieval-augmented generation (RAG) systems, semantic search applications, and multilingual information retrieval, text-embed-3s demonstrates clear superiority across all evaluated benchmarks (Figure 6a), with particularly strong performance in MIRACL tasks (40.13% improvement over text-embed-ada).

7.9 Limitations and Future Research Directions

While this evaluation provides comprehensive insights across multiple performance dimensions, several limitations warrant acknowledgment. The evaluation framework focuses primarily on quantitative metrics and may not capture subjective quality aspects such as response creativity, conversational coherence, or domain-specific expertise. Future research should investigate:

- **Domain-Specific Performance**: Evaluation across specialized domains (legal, medical, scientific) to assess model performance in expert-knowledge contexts.
- **Dynamic Performance**: Longitudinal studies examining model performance stability and degradation over extended deployment periods.
- **Multi-Modal Integration**: Assessment of model performance when integrated with visual, audio, or structured data inputs.
- **Computational Efficiency**: Trade-off analysis between performance gains and computational resource requirements for practical deployment scenarios.

This comprehensive evaluation across six analytical frameworks (Figures 5–7) establishes clear performance hierarchies and trade-off relationships among contemporary language models and embedding architectures. GPT-4.0-mini emerges as the most balanced performer across multiple dimensions, while GPT-3.5-1106 demonstrates superior relevancy performance. Text-embed-3s consistently outperforms its predecessor across all embedding benchmarks.

The fundamental trade-off between faithfulness and relevancy (Equation 12) necessitates application-specific optimization strategies, while the statistical analysis (Equations 13 and 14) provides quantitative foundations for informed model selection. These findings, supported by comprehensive visual and mathematical analysis, provide empirical foundations for evidence-based model selection strategies in production deployments, enabling practitioners to make informed decisions based on specific application requirements and performance priorities.

The experimental results provide compelling evidence for the effectiveness of modern embedding models in vector search applications. The text-embedding-3-small model demonstrates significant improvements over its predecessor, text-embedding-ada-002, across multiple evaluation metrics. This improvement can be attributed to several factors:

1. **Architectural Refinements:** The newer model incorporates improved neural network architectures that better capture semantic relationships in text data, resulting in more meaningful vector representations.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

2. **Training Data Quality:** Enhanced training datasets and methodologies contribute to better representation learning, enabling the model to understand context and nuance more effectively.
3. **Dimensional Optimization:** More efficient use of vector dimensions allows for better information encoding without increased computational overhead, making the system more cost-effective.

The substantial improvements in context precision and recall are particularly noteworthy, as these metrics directly impact the quality of retrieved information in semantic search applications. The 30.6% improvement in context precision indicates that the newer embedding model is significantly better at identifying truly relevant content, while the 32.4% improvement in context recall suggests more comprehensive retrieval of pertinent information.

7.10 Practical Implications

These findings have important implications for practitioners implementing vector search systems:

- Organizations should prioritize upgrading to newer embedding models when possible, as the performance gains justify the transition effort and associated costs.
- The choice of completion model should be based on specific use case requirements, with GPT-4.0-mini preferred for applications requiring high faithfulness and GPT-3.5-turbo variants for scenarios prioritizing answer relevancy.
- MongoDB Atlas Vector Search proves to be an effective platform for implementing semantic search capabilities in production environments, offering both performance benefits and security features necessary for enterprise applications.
- The cost-effectiveness of text-embedding-3-small, with pricing reduced by 5x compared to text-embedding-ada-002, makes advanced vector search capabilities more accessible to organizations with budget constraints.

7.11 Security and Information Management Implications

The integration of vector search systems with MongoDB Atlas provides enhanced security features that are crucial for organizations managing sensitive unstructured data. The platform's role-based access control, encryption capabilities, and audit logging features address many of the security concerns associated with vector databases. Organizations implementing these systems for cybersecurity applications benefit from improved threat detection capabilities while maintaining data privacy and compliance with regulatory requirements.

The semantic search capabilities demonstrated in this research have direct applications in information retrieval systems for cybersecurity, where rapid identification and analysis of relevant threat intelligence is critical. The improved context precision and recall metrics translate to more effective automated incident response and threat hunting capabilities.

8 Limitations and Future Work

While our research provides valuable insights into vector search performance, several limitations should be acknowledged:

1. The evaluation was conducted using specific datasets and may not generalize to all application domains, particularly those with highly specialized terminology or unique semantic structures.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

2. The study focused primarily on English language content, limiting applicability to multilingual scenarios where different embedding models might perform differently across languages.
3. Computational cost analysis was not included in the current evaluation framework, though the 5x cost reduction of text-embedding-3-small provides initial cost-effectiveness insights.
4. Security evaluation was limited to theoretical considerations rather than comprehensive penetration testing or vulnerability assessment.

Future research directions include:

- Expanding the evaluation to include multilingual datasets and cross-lingual retrieval scenarios to better understand the global applicability of these systems.
- Investigating the impact of different vector database configurations on search performance, including various indexing strategies and optimization techniques.
- Conducting comprehensive cost-benefit analyses of different model combinations, including total cost of ownership considerations for production deployments.
- Exploring the effectiveness of hybrid search approaches combining vector and traditional search methods for improved precision and recall.
- Implementing comprehensive security testing frameworks to evaluate real-world vulnerabilities in vector search systems.
- Investigating the application of these systems in specialized domains such as healthcare, legal research, and financial services where domain-specific knowledge is critical.

9 Conclusion

This research demonstrates the significant advantages of modern embedding models in vector search applications, particularly when implemented using MongoDB Atlas as the vector database platform. The text-embedding-3-small model shows substantial improvements over text-embedding-ada-002, with gains of 30.6% in context precision and 32.4% in context recall. These improvements translate to more accurate and comprehensive semantic search capabilities, directly impacting the quality of LLM-powered applications.

The experimental evaluation reveals that different completion models excel in different aspects of response generation, suggesting that model selection should be tailored to specific application requirements. GPT-4.0-mini demonstrates superior faithfulness, making it suitable for applications where accuracy is paramount, while GPT-3.5-turbo variants show strong performance in answer relevancy metrics.

Our findings provide practical guidance for organizations implementing vector search systems and contribute to the broader understanding of embedding model performance in real-world applications. The integration of MongoDB Atlas with modern embedding models offers enhanced security features, cost-effectiveness, and scalability that are essential for production deployments in information management systems.

The research underscores the importance of staying current with embedding model advances, as the performance improvements demonstrated here can significantly enhance the effectiveness of semantic search implementations. Organizations investing in LLM-powered applications should prioritize the adoption of modern embedding architectures to maximize the value of their vector search capabilities.

The security considerations discussed in this work highlight the need for careful implementation of vector search systems, particularly in applications handling sensitive data. The combination of MongoDB Atlas's enterprise-grade security features with modern embedding models provides a robust foundation for secure information retrieval systems in cybersecurity and other security-sensitive applications.

As vector search technology continues to evolve, the findings presented in this research provide a foundation for future developments in semantic search and information retrieval systems. The demonstrated improvements in both performance and cost-effectiveness suggest that vector search will become increasingly accessible and valuable across a wide range of applications and industries.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 4171–4186, 2019.

[3] K. Nassiri and M. A. Akhloufi, "Recent Advances in Large Language Models for Healthcare," *BioMedInformatics*, vol. 4, no. 2, pp. 1097–1143, Apr. 2024, doi: 10.3390/biomedinformatics4020062.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[5] L. Wang et al., "A survey on large language model based autonomous agents," *Frontiers in Computer Science*, vol. 18, no. 6, p. 186345, Dec. 2024, doi: 10.1007/s11704-024-40231-1.

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI Technical Report*, 2018.

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Technical Report*, 2019.

[8] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint* arXiv:2307.09288, 2023.

[9] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "PaLM: Scaling language modeling with pathways," *arXiv preprint* arXiv:2204.02311, 2022.

[10] C. Lülf, D. M. L. Martins, M. A. V. Salles, Y. Zhou, and F. Gieseke, "Fast Search-by-Classification for Large-Scale Databases Using Index-Aware Decision Trees and Random Forests," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 2845–2857, Jul. 2023, doi: 10.14778/3611479.3611492.

[11] B. Sarmah, B. Hall, R. Rao, S. Patel, S. Pasquali, and D. Mehta, "HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction," *arXiv preprint* arXiv:2408.04948, Aug. 2024, doi: 10.48550/arXiv.2408.04948.

[12] L. Huang et al., "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–55, Mar. 2025, doi: 10.1145/3703155.

DOI: https://doi.org/10.54216/JCIM.170211

Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025

[13] L. Trümper, T. Ben-Nun, P. Schaad, A. Calotoiu, and T. Hoefler, "Performance Embeddings: A Similarity-Based Transfer Tuning Approach to Performance Optimization," in *Proceedings of the 37th International Conference on Supercomputing*, ICS '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 50–62, doi: 10.1145/3577193.3593714.

[14] S. Mohammadi, A. Balador, S. Sinaei, and F. Flammini, "Balancing privacy and performance in federated learning: A systematic literature review on methods and metrics," *Journal of Parallel and Distributed Computing*, vol. 192, p. 104918, Oct. 2024, doi: 10.1016/j.jpdc.2024.104918.

[15] OpenAI, "New embedding models and API updates," *OpenAI Technical Blog*, January 2024. [Online]. Available: https://openai.com/index/new-embedding-models-and-api-updates/

[16] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," *arXiv preprint* arXiv:2309.15217, 2023.

[17] Y. Hu and Y. Lu, "RAG and RAU: A Survey on Retrieval-Augmented Language Models in NLP," *arXiv preprint* arXiv:2404.19543, 2024.

[18] S. Wu, "Retrieval-Augmented Generation for Natural Language Processing: A Survey," *arXiv preprint* arXiv:2407.13193, 2024.

[19] J. Lin et al., "Vector Search with OpenAI Embeddings: Lucene Is All You Need," *arXiv preprint* arXiv:2308.14963, 2023.

[20] AdaSci Team, "MongoDB Atlas Vector Search for RAG Powered LLM Applications," *Technical Article, AdaSci*, 2024.

[21] Y. Han, X. Zhang, R. Chen, L. Wu, and Q. Liu, "When Large Language Models Meet Vector Databases: A Survey," *arXiv preprint* arXiv:2402.01763, 2024.

[22] MongoDB Community, "Using OpenAI Latest Embeddings in a RAG System With MongoDB," *MongoDB Developer Blog*, 2022.

[23] AA. S. Alzubaidi, M. A. A. Al-Khalidi, and S. A. H. Al-Tamimi, "Deep Learning Approaches for Sentiment Analysis: A Comprehensive Review," *Journal of Computer Science and Technology*, vol. 39, no. 2, pp. 321–340, 2023, doi: 10.1007/s11390-023-00235-6.

[24] S. Ghane, R. Sawant, G. Supe, and C. Pichad, "LangchainIQ: Intelligent Content and Query Processing," *International Journal of Management, Technology, and Social Sciences*, pp. 34–43, Aug. 2024, doi: 10.47992/IJMTS.2581.6012.0360.

[25] A. Hendawi et al., "Distributed NoSQL Data Stores: Performance Analysis and a Case Study," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA: IEEE, Dec. 2018, pp. 1937–1944, doi: 10.1109/BigData.2018.8622544.

[26] G. V. N. Priyanka, S. Vasavi, and A. Anu Gokhale, "Evaluation of Performance Metrics in GeoRediSpark Framework for GeoSpatial Query Processing," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, vol. 3, S. C. Satapathy, K. S. Raju, K. Shyamala, D. R. Krishna, and M. N. Favorskaya, Eds., Cham: Springer International Publishing, 2020, pp. 318–325, doi: 10.1007/978-3-030-24322-7_41.

[27] M. Marountas, G. Drakopoulos, P. Mylonas, and S. Sioutas, "Recommending Database Architectures for Social Queries: A Twitter Case Study," in *Artificial Intelligence Applications and Innovations*, vol. 627, I. Maglogiannis, J. Macintyre, and L. Iliadis, Eds., Cham: Springer International Publishing, 2021, pp. 715–728, doi: 10.1007/978-3-030-79150-6_56.

[28] A. Costea et al., "VectorH: Taking SQL-on-Hadoop to the Next Level," in *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1105–1117, doi: 10.1145/2882903.2903742.

[29] Z. Liu et al., "Mitigating Privacy Risks in LLM Embeddings from Embedding Inversion," *arXiv preprint* arXiv:2411.05034, 2024.

[30] "ragas-wikiqa dataset," Hugging Face Datasets, 2024. [Online]. Available: https://huggingface.co/datasets/explodinggradients/ragas-wikiqa.

DOI: https://doi.org/10.54216/JCIM.170211
Received: April 10, 2025 Revised: June 23, 2025 Accepted: August 17, 2025