International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Vector Databases and Retrieval-Augmented Generation for Accurate AI-Driven Project Estimation

Sangeeta Manna (*nee* Datta)

BSc. Hons. (Cal.), Master of Computer Applications (UTech. WB. India)
Senior Software Programmer in Banking, Finance & Service-related IT Industry

Abstract: ***Large Language Models (LLMs) can analyse unstructured enterprise documents but lack organizational memory, resulting in inconsistent and unreliable software project estimations. Retrieval-Augmented Generation (RAG) addresses this limitation by integrating vector databases that store semantic embeddings of historical project artifacts. Using similarity-based retrieval rather than keyword matching, RAG enables LLMs to ground estimations in relevant past projects despite variations in terminology or technology. This evidence-driven approach improves estimation accuracy, consistency, and explainability, providing a scalable foundation for AI-assisted software project estimation in enterprise environments.***

Keywords: Vector Database, Retrieval-Augmented Generation (RAG), Large Language Models, Software Project Estimation, Semantic Search, AI Decision Support Systems

1. Introduction

The project effort estimation remains one of the most challenging activities in software engineering. Accurate estimation depends on expert judgment, historical analogies, and domain knowledge accumulated across previous projects. Traditional AI-based estimation approaches, when built solely on Large Language Models (LLMs), lack access to such historical context and therefore operate in isolation.

Although LLMs [1] can parse the documents such as presales documents or Business Requirement Documents (BRDs) and generate structured outputs, they fail to incorporate prior organizational experience, resulting in inconsistent and often overly optimistic estimates. This paper addresses this limitation by introducing a vector database–backed RAG architecture that enables AI systems to reason over semantically similar historical estimations at inference time.

2. Problem Statement

In enterprise environments, similar projects are repeatedly executed with minor variations in scope, technology, or methodology. Human estimators naturally rely on past experiences to calibrate new estimates. LLM-only systems lack this capability because they do not retain project-specific historical data.

Key challenges include:
- Lack of contextual grounding
- High variance in repeated estimations
- Absence of learning across iterations
- Over-generalized risk and timeline outputs

3. Aims and Objectives

The primary aim of this research is to design, implement, and evaluate a Retrieval-Augmented Generation (RAG)–based AI architecture that leverages vector databases and semantic embeddings to improve the accuracy, consistency, and reliability of AI-driven software project effort estimation by incorporating historical organizational knowledge at inference time. The objective of this research is to investigate how vector databases and Retrieval-Augmented Generation (RAG) can be systematically integrated to overcome the limitations of LLM-only approaches in software project estimation. The study aims to examine the internal architecture and mathematical foundations of vector databases, including embedding representations, similarity measures, and approximate nearest neighbour search, in order to establish their suitability for semantic retrieval in enterprise decision-making systems. The objective of this research is to investigate how vector databases and Retrieval-Augmented Generation (RAG) can be systematically integrated to overcome the limitations of LLM-only approaches in software project estimation. The study aims to examine the internal architecture and mathematical foundations of vector databases, including embedding representations, similarity measures, and approximate nearest neighbour search, in order to establish their suitability for semantic retrieval in enterprise decision-making systems.

4. Methodology & Design

4.1 Vector Databases: Internal Architecture and Evolution

Vector databases [5] are designed to store and retrieve high-dimensional numerical representations known as embeddings. Unlike keyword-based systems, vector databases enable semantic similarity search independent of lexical overlap.

1) Mathematical Representation
Let a document d be mapped to an embedding vector:

$$f(d) = v\_d \text{ in } R\text{^ } k$$

where k is 1024 in modern transformer-based embeding models.

Similarity between two documents is computed using cosine similarity:

$$sim(v_a, v_b) = (v_a \cdot v_b) / (||v_a|| \, ||v_b||)$$

documents with similar meaning cluster together in the vector space.

2) Embeddings and Semantic Neighbourhoods

Embeddings transform unstructured text into continuous representations where semantic proximity corresponds to geometric proximity. This property enables analogy-based reasoning, which is an essential mechanism for accurate project effort estimation.

Example

Three BRDs describing:

- Payment gateway integration
- Money transfer process
- E-Check generation process

may share little vocabulary but produce embeddings with high cosine similarity, enabling semantic retrieval.

3) Keyword Search vs Vector Search (Mathematical Comparison)

Keyword search represents documents as sparse vectors over vocabulary v while vector search uses dense representations.

Property	Keyword Search	Vector Search
Vocabulary dependence	High	None
Semantic generalization	No	Yes
Robust to paraphrasing	No	Yes
Scalability	Limited	High (ANN)

Vector search strictly generalizes keyword search and forms the foundation for RAG systems.

4.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation is an architectural pattern that enhances LLM inference by dynamically injecting retrieved external knowledge.

1) RAG Pipeline

Input -> Embedding -> Vector Retrieval -> Context Augmentation -> LLM -> Output

Instead of relying solely on pretrained parameters, the LLM reasons over retrieved historical data.

2) Design Principle: Two-Phase Execution Model

A key design principle of the proposed system is the **separation of inference and learning**.

Phase 1: Inference (Estimation Generation)

- Uses historical data
- Does not modify the vector database

Phase 2: Knowledge Accumulation

- Stores finalized estimations
- Enables future retrieval

This separation ensures controlled growth of organizational knowledge.

4.3 System Architecture

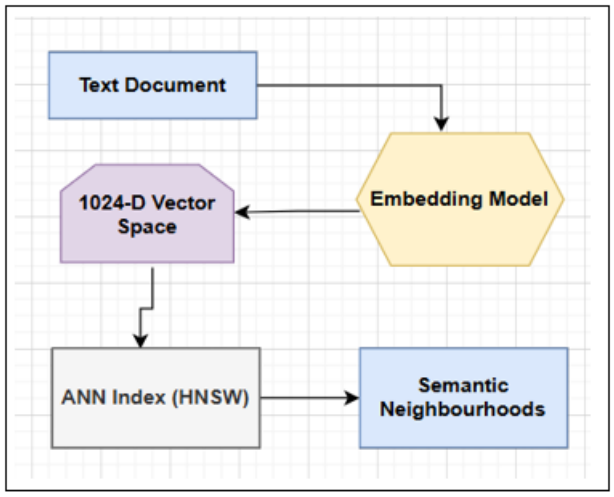

Figure 1: Vector Database Internal Architecture

Internal architecture of a vector database showing embedding generation, ANN indexing, and semantic clustering.

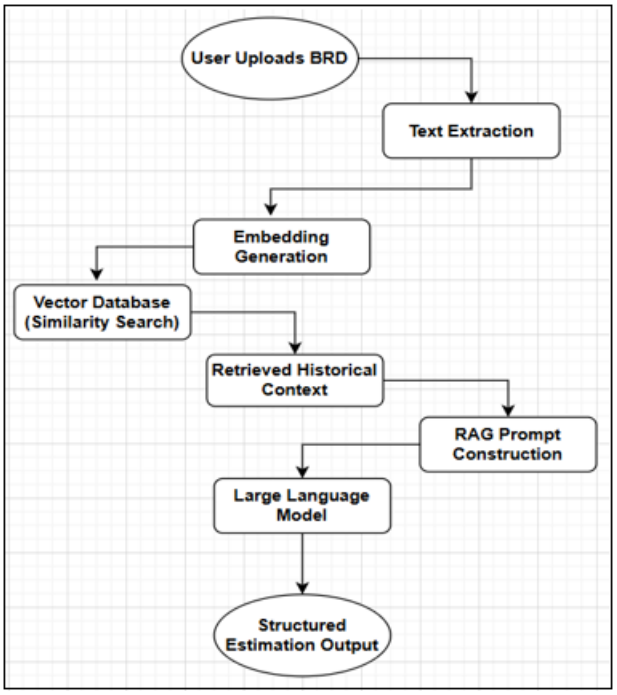

Figure 2: Retrieval-Augmented Generation (RAG) Pipeline

End-to-end RAG-based project estimation pipeline.

1) Comparison- LLM vs RAG-based Estimation

LLM Only

Input -> LLM -> Output

(no memory)

RAG-based System

Input -> Retrieval -> LLM -> Output

(context-aware)

2) Implementation

The system was implemented using:

- Spring AI for orchestration [3]
- Transformer-based embedding models
- A vector database for semantic retrieval
- A large language model for structured reasoning

Only embeddings and metadata were stored, ensuring data confidentiality.

5. Related Works & Implementation

5.1 Case Study: Industry-Level Project Estimation

The system was evaluated using a collection of enterprise Business Requirement Documents (BRDs) related to financial systems and payment platforms. These documents covered a range of real-world use cases, including payment processing, bank integrations, and transaction management workflows. For each BRD, the system generated structured estimation outputs, including module-wise Work Breakdown Structures (WBS), role-based effort allocation across development and support functions, methodology-specific delivery timelines aligned with Agile or Waterfall models, and a comprehensive analysis of project assumptions and risks.

Outputs included:

- Module-wise Work Breakdown Structures
- Role-based effort allocation
- Methodology-specific timelines
- Risk and assumption analysis

a) Experimental Results & Analysis

Comparative evaluation showed:

Approach	Estimation Variance	Consistency
LLM only	High	Low
LLM + RAG	Low	High

The RAG-based system consistently aligned new estimates with historical organizational patterns.

Code Snippet:

```
// new: replace methodology placeholder in template (if present)
private String buildPrompt(String brd) {  1 usage   sangeeta *
    try {
        String template = new String(
                promptResource.getInputStream().readAllBytes(),
                StandardCharsets.UTF_8
        );
        return template.replace( target: "{{BRD}}", brd);
    } catch (Exception e) {
        throw new RuntimeException("Failed to load prompt template", e);
    }
}

Edit | Explain | Test | Document | Fix
private String buildPrompt(String brd, String methodology, String historicalContext) {  2 usages   new *

    String template = buildPrompt(brd);

    template = template.replace( target: "{{METHODOLOGY}}", methodology);

    if (template.contains("{{HISTORICAL_CONTEXT}}")) {
        template = template.replace( target: "{{HISTORICAL_CONTEXT}}", historicalContext);
    } else {
        template += "\n\n### Historical Estimation Context\n"
                + historicalContext + "\n";
    }
    return template;
}
```

Prompt Engineering:

```
JSON structure (MUST match exactly):
{
  "methodology": "",
  "wbs": [
    {
      "module": "",
      "tasks": [
        { "task": "", "estimate_hours": 0, "owner_role": "" }
      ]
    }
  ],
  "effort":[
    { "module":"", "fe":0, "be":0, "qa":0, "uiux":0, "devops":0, "total":0 }
  ],
  "resources":[
    { "role":"", "count":0, "duration_weeks":0 }
  ],
  "timeline":[
    { "phase":"", "duration_weeks":0, "depends_on":"" }
  ],
  "assumptions":[ "" ],
  "risks":[
    { "risk":"", "mitigation":"" }
  ]
}
```

5.2 Data Collection & Preprocessing

Historical project estimation summaries were collected from previously completed enterprise projects and prepared for semantic retrieval through a structured preprocessing pipeline. The collected data was first normalized to remove formatting inconsistencies, redundant information, and noise commonly present in unstructured project documents. The normalized text was then segmented into smaller, semantically coherent chunks to ensure effective embedding generation and to improve the precision of similarity-based retrieval.

Each chunk was transformed into a high-dimensional embedding using a transformer-based embedding model and stored in the vector database along with relevant metadata, such as project methodology (Agile or Waterfall) and application domain. These metadata attributes enabled filtered retrieval during inference, ensuring that only contextually relevant historical estimations were used for grounding the estimation process. To preserve data confidentiality and reduce storage overhead, raw project documents were not persisted in the vector database; instead, only embeddings and associated metadata were retained.
Historical estimation summaries were:

- Normalized
- Chunked
- Embedded
- Stored with metadata filters (methodology, domain)

Raw documents were not persisted in the vector database.

6. Result Analysis

The proposed RAG-based estimation system was evaluated by comparing estimation outputs generated using a baseline LLM-only approach against those generated using the vector database–augmented RAG architecture. The evaluation focused on qualitative and quantitative aspects of estimation quality, including consistency, realism, and alignment with historical project data.

6.1 Estimation Consistency

When the same or closely related BRDs were provided as input, the LLM-only approach produced noticeably different effort estimates [4] across multiple runs, reflecting high variance and sensitivity to prompt phrasing. In contrast, the RAG-based system consistently retrieved semantically similar historical projects and produced estimates that remained within a narrow range of variation. This indicates that historical grounding via vector similarity significantly improves estimation stability.

6.2 Effort and Timeline Realism

Estimates generated by the LLM-only approach tended to be optimistic, particularly in development timelines and quality assurance effort. The RAG-based system, by incorporating prior project outcomes, produced more conservative and realistic timelines that closely aligned with historical delivery patterns. This effect was especially pronounced in Agile-based projects, where sprint-level estimates benefited from retrieval of comparable sprint structures from previous projects.

6.3 Risk and Assumption Identification

The RAG-based system demonstrated improved contextual awareness in identifying project risks and assumptions. Retrieved historical estimations contributed domain-specific risk factors- such as third-party API dependencies and compliance constraints- that were frequently omitted by the LLM-only approach. This grounding resulted in more comprehensive and relevant risk analysis.

6.4 Overall Impact

Overall, the experimental results indicate that integrating vector database–driven retrieval into the estimation workflow reduces estimation variance, enhances contextual relevance, and improves trustworthiness of AI-generated outputs. These findings validate the effectiveness of Retrieval-Augmented Generation as a practical and scalable solution for enterprise AI-driven project estimation.

7. Future Work

Future enhancements include:

- **Multi-modal embeddings** to incorporate non-textual artifacts such as architectural diagrams, workflows, and tabular specifications into the estimation process.
- **Confidence scoring mechanisms** to quantify the reliability of AI-generated estimations based on retrieval similarity distributions and historical variance.
- **Cross-organization estimation transfer**, enabling anonymized and domain-aligned historical estimation knowledge to be shared across organizations while preserving data privacy.

8. Conclusion

By integrating vector databases with Retrieval-Augmented Generation, AI-driven project estimation systems can move beyond static inference toward context-aware, continuously improving decision support tools. The proposed architecture demonstrates how organizational memory can be leveraged at inference time to improve estimation accuracy without retraining language models.

References

[1] Lewis et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
[2] Malkov & Yashunin, Efficient Approximate Nearest Neighbour Search
[3] Spring AI Documentation
[4] IEEE Software Engineering Estimation Standards
[5] Pinecone Systems Inc. Vector Search and Similarity Indexing Architecture. Technical Documentation, 2023.
[6] Boehm, B. W. Software Engineering Economics. Prentice Hall, 1981.
[7] IEEE Computer Society. Guide for Software Cost Estimation. IEEE Std 14143, 2016.