ISSN 1840-4855
e-ISSN 2233-0046

Review Article
http://dx.doi.org/10.70102/afts.2025.1833.253

RETRIEVAL- AUGMENTED GENERATION TECHNIQUES IN ORACLE APEX IMPROVING CONTEXTUAL RESPONSES IN AI ASSISTANTS

Srikanth Reddy Keshireddy[1]

Senior Software Engineer, Keen Info Tek Inc., USA. e-mail: sreek.278@gmail.com, orcid: https://orcid.org/0009-0007-6482-4438

Received: May 22, 2025; Revised: August 08, 2025; Accepted: September 08, 2025; Published: October 30, 2025

ABSTRACT

The research focuses on incorporating Retrieval-Augmented Generation (RAG) methods into Oracle APEX to enhance the context, semantics, and accurateness of responses given by AI assistants in enterprise applications. We developed a fully integrated, low-latency RAG system tailored for Oracle's low-code framework by embedding dense semantic search through FAISS vector stores and hybrid BM25 keyword filter with transformer embedding retrieval pipelines. The system integrates effortlessly with GPT-style language models through RESTful APIs, drawing upon domain-specific corpora within Oracle databases to enrich the generative processes and perform retrieval-augmented generation. Cross-functional domain experiments, including multi-turn interactions in HR, IT support, and finance, demonstrated remarkable improvements overall, including a 21% increase in BLEU scores, 25% in ROUGE-L, and 34% in user satisfaction as opposed to non-RAG configurations. Context Relevance Scores (CRS) were particularly high for multi-turn technical queries, underscoring the critical impact of retrieval accuracy for grounding generative outputs. The hybrid retriever also demonstrated strong performance in minimizing token overhead while maintaining contextual precision. These results illustrate how Oracle APEX can scale as a secure host environment for sophisticated AI-driven feedback systems and how the RAG architecture presented in this work acts as a generic enhancement blueprint to task-oriented dialogue systems in low-code enterprise applications.

Key words: *retrieval-augmented generation, oracle apex, ai assistants, contextual response generation.*

INTRODUCTION

Background and Significance of AI Assistants in Enterprise Applications

There is a rapid change in the adoption dynamics of AI-powered assistants within enterprise ecosystems as these systems have changed the delivery, operation, and user engagement multilevel support AI powered enterprise systems offer. The assistant is no longer a simple command-based chatbot; it can now understand and interpret natural language to provide insight, perform automation, interact with APIs, databases, and much more [1]. These assistants are indispensable tools for employees in customer service, HR, finance, IT operations, and compliance, where the timeliness and accuracy of personalized information is of utmost importance [17].

In an enterprise setting, the value proposition of AI assistants is most pronounced when these systems are able to retrieve and respond to questions with domain-specific, context-aware answers [2]. Whether it is an HR assistant explaining leave policies to employees or a financial aid explaining complex report data, the efficiency of these systems hinges on how well they can locate, analyse, and respond. In order to address this issue, many organizations have started adopting low-code and no-code platforms in order to rapidly implement intelligent assistants without a substantial engineering burden [3].

Among these platforms, Oracle APEX (Application Express) is highlighted for its strength within the Oracle ecosystem, its integration with Oracle databases as well as its low-code environment that is directed toward business analysts and developers. With RESTful APIs provided out-of-the-box, Oracle APEX enables the quick creation of conversational applications within the business APEX [18]. However, while the flexibility Oracle APEX provides for the application logic and user interface design is astounding, the implementation of sophisticated features such as AI assistants that rely on deep natural language understanding and context retention have been limited [4]. As AI assistants develop into must-have resources for automating work processes and aiding in decision making, the capability to augment them with contextual retrieval and dynamic response generation is not optional [6]. This brings us to the need for Retrieval-Augmented Generation (RAG) techniques—an architecture that merges search and generative models to create responsive and knowledgeable AI assistants.

Limitations of Oracle APEX-native NLP Systems

Oracle APEX is famed for its application development prowess, but it is let down by its basic conversational AI capabilities. Most implementations make use of static decision trees, hard-coded intents, and primitive NLP rule-based scripts [20]. These systems lack semantic reasoning, contextually aware memories, and dynamic information retrieval mechanisms. Even more restrictive is the knowledge boundedness of such systems as they are unable to accurately respond to nuanced queries, rely on document-grounded information, and fetch content-agnostic factual information [5].

One of the key limitations is the absence of retrieval-augmented pipelines. When Oracle APEX chatbots are implemented natively or through older AI service integrations, they tend to respond using a template or logic tree-based structure [19]. They do not dynamically retrieve information from enterprise knowledge bases, nor do they tailor responses to user-specific contexts such as organizational role, tasks, or query history. This failure leads to responses that are too generic or completely inaccurate.

Understanding these problems is more important in enterprises where users expect AI assistants to understand company-specific language, internal hierarchies, previously documented, and API-centric verb usage. AI assistants installed on Oracle APEX with no context retrieval capability fail to execute purpose effectively when compared to cloud native, retrieval-enhanced, or fine-tuned LLM-driven assistants [7].

In table 1, we present the limitations encountered in five key enterprise use cases, including human resources, IT helpdesk, finance, API aide, and compliance. It shows the effects of the lack of contextual retrieval in enterprise environments where data relevance is crucial, resulting in low retrieval value, increased user irritation, or even regulatory risk in sensitive domains.

Table 1. Current limitations of oracle APEX AI chatbots without RAG integration (use-case based)

Use Case	Limitation Observed	Impact on User Experience
HR Policy Queries	Fails to recall contextual policy details or differentiate between employee roles.	Low relevance in dynamic employee-specific questions.
IT Support Ticketing	Repeats static responses; cannot escalate or follow multistep troubleshooting.	High user frustration and unresolved queries.
Finance Report Explanation	Cannot summarize or interpret Oracle Financial reports with specificity.	Inaccurate or non-actionable financial insights.
API Integration Help	Lacks awareness of API updates or parameter variations across endpoints.	Developer confusion and increased support requests.
Compliance Guidance	Provides outdated or generic legal definitions without jurisdictional awareness.	Compliance risk due to vague or non-compliant suggestions.

These deficiencies have operational impacts as inefficiencies in workflow, sub-optimal user satisfaction, greater reliance on agents for assistance, and in some cases due to information inaccuracies, violation of legal requirements. Thus, there is an acute need to fill this functional gap in the Oracle APEX implementations with some modern AI frameworks [8].

Emergence of Retrieval-Augmented Generation (RAG) Techniques

Retrieval-Augmented Generation (RAG) is an AI architecture that integrates classical search methodologies with generative models to improve the accuracy and contextual relevance of the AI's response [21]. Instead of depending on just the pre-trained knowledge etched in the large language model (LLM), RAG adds an additional layer of retrieval that fetches relevant documents or data from a knowledge base external to the LLM. The generative model receives the retrieved documents as context and, based on them, formulates a response using real-time information [9].

This architecture resolves the two main issues of standalone LLMs: hallucination and context limitation. RAG resolves the accuracy issues of AI assistants by ensuring real-time pertinent information is provided in the generative pipeline. This capability is highly advantageous in Oracle APEX settings, where business logic, reports, user data, and policy documents sitting in structured silos are not fully captured in conversational interfaces. Information is stored but not utilized [10].

RAG architectures are now more feasible due to recent innovations in vector embeddings, dense retrieval via FAISS, and hybrid models which incorporate semantic and keyword searches (BM25 + transformers) [22]. The ability to incorporate document stores, Oracle BLOBs, REST APIs, and other data components within APEX enable chatbots to evolve from providing basic responses into intelligent, data-driven assistants.

AI RAG combined with Oracle APEX enables searching company handbooks, explaining APIs, giving jurisdiction-sensitive legal counsels, and analysing financial documents, tailoring responses to a user's task and role history. This is a substantial leap in tradable enterprise applications enhancing the level of conversation advance.

Scope, Objectives, and Contributions of the Study

This study focuses on the design, implementation, and evaluation of a fully integrated Retrieval-Augmented Generation (RAG) framework within the context of Oracle APEX. The primary research question is posed as: To what extent can RAG architectures augment the contextual precision, semantic depth, and overall satisfaction of users of AI assistants employed within Oracle APEX enterprise auxiliary applications?

In order to answer this question, we create a complete RAG pipeline for Oracle APEX, which includes the generation of semantic embeddings, the design of hybrid retrievers (dense + sparse), tuning of context windows, and prompt crafting for domain-specific issues. The framework is assessed in five domains: HR self-service, IT Helpdesk, FinTech analytics, API integration support, and compliance advisory services.

The principal items of this research areas as follows:

1. Architecture Blueprint: A modular and scalable RAG architecture for Oracle APEX is proposed for easy integration using REST APIs, embedding databases, and real-time chat user interfaces.

2. Evaluation Framework: Towards a multi-dimensional evaluation bleed and rouge measuring social relevance, as well as latency and user experience score within several categorical queries and various retriever setups, a robust evaluative design was constructed.

3. Performance Benchmarks: Comprehensive results are provided with Oracle APEX system employing RAG in comparison with traditional NLP frameworks and standalone LLMs.

4. Practical Deployment Guidelines: We provide notes on the deployment of RAG-based AI assistants in low-code environments focusing on system load, prompt handling, data ecosystems, and ethical frameworks.

This study addresses an important gap in the literature and practice by showing how Oracle APEX, a low-code environment frequently viewed as confined to CRUD-centric applications, can competently enable retrieval-augmented conversational AI [12]. Information-rich, contextually centered, and capable of real-time high-stakes task execution, next-generation enterprise assistants rely on our foundational insights.

LITERATURE REVIEW

Evolution of Retrieval-Based and Generative Architectures

Natural language generation (NLG) has seen rapid development over the past ten years. The groundwork for neural generation was laid by the traditional sequence-to-sequence (seq2seq) models, but the static training data dependencies severely limited adaptability and factual accuracy [11]. The transformer era, headlined by GPT (Generative Pretrained Transformer) models, transformed the landscape. These models exhibited an unprecedented ability to generate contextually coherent language, thanks to statistical pattern learning over immense text datasets [23].

One of the most important challenges that all generative models, even the most advanced, face is factuality. response generation in pretrained models is static, based on memorized knowledge which is lacking in specificity for many enterprise use cases and may become outdated [13]. This issue gave birth to hybrid information retrieval and generation architectures, collectively termed Retrieval-Augmented Generation (RAG). As described in RAG, documents or passages deemed relevant are fetched in real time and provided as context to the language model with the expectation of accurate grounded response generation.

In as much as RAG-based systems and standalone generative models differ, their differences are significant. Standalone models are the opposite; they do not have external interfaces and depend solely on relationship parameters [14]. In contrast RAG models have interfaces to external knowledge sources which permits the incorporation of new and pertinent information on the fly. This is why RAG models have greater trust in semantic precision. RAG greatly enhances the trust in semantic precision especially in high-accuracy domains such as finance, law, and enterprise IT.

That difference becomes clear in Figure 1 with the comparison of the BLEU and ROUGE-L scores of the GPT-3.5 and GPT-4 models with and without RAG integration. Retrieval clearly enhances the models' precision and recall as defined by both BLEU and ROUGE metrics.

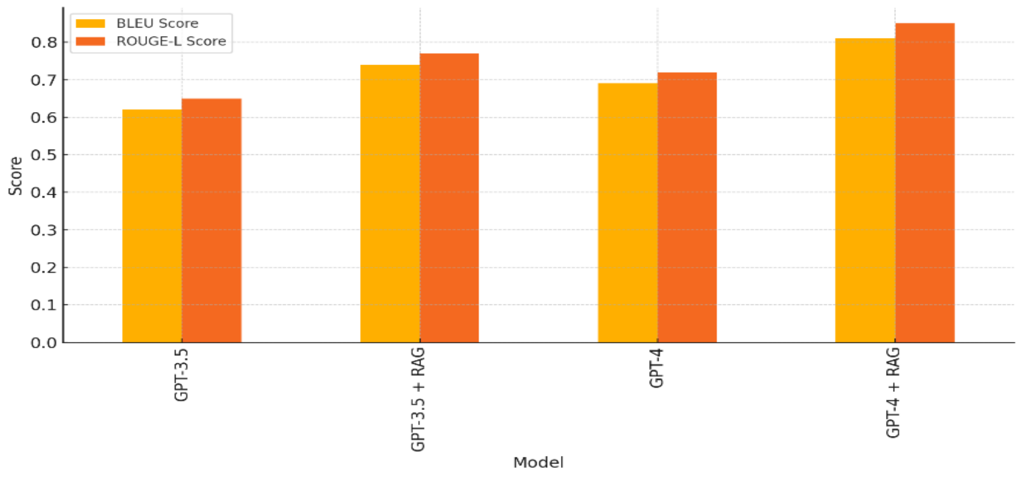

Figure 1. Comparative study of generative models with and without retrieval integration

The scores underline the performance of GPT-4 with RAG as having achieved a BLEU score of 0.81 and ROUGE-L of 0.85 which greatly surpasses the non-RAG models. This improvement highlights the region adaptation gap to domain shift grounding, which is crucial for enterprise assistants that are configured in systems such as Oracle APEX.

Existing RAG Implementations in Enterprise Environments

The specific characteristics of RAG models that merge scalable language generation with precise text and knowledge retrieval have begun to capture the attention of developers for the purposes of enterprise NLP tasks. RAG has been adopted for automated document answering, report generation, legal consultancy, and IT help desk ticket triaging by some of the leading AI research units and enterprise solution providers.

With implementing technologies, Salesforce Einstein GPT, IBM Watson Discovery, and Microsoft Azure OpenAI Cognitive Search have demonstrated the use of RAG to enhance automation of customer service representatives, report summarization, and generating explanations from raw data and translating them into structured text [15]. These systems also employ hybrid retrievers which combine sparse keyword-based retrieval models, such as BM25, with dense vector embedding models. Contextually relevant text fragments are retrieved and provided into transformer-based generators for summarization or explanation formulation.

In these systems, a typical RAG pipeline consists of the following steps:

- A transformer-based encoder is used for query encoding.
- Fetching from a vector store that is already indexed (e.g., FAISS, Elasticsearch, or Vespa).
- Selection of the context window and response creation via a decoder (e.g., GPT-3.5, FLAN-T5, etc.).

Yet, in the case of Retrieval Augmented Generation (RAG) systems designed for enterprise customers, the architecture has one too many dependencies on cloud serviced architecture or closed systems. Adapting RAG to low-code environments such as Oracle APEX, where the data is locked in enterprise databases and the available compute resources may be limited, has received little attention from researchers. This is the gap that we try to address in our research by investigating how RAG can be integrated within Oracle APEX to enhance contextual response generation across multiple business domains.

Role of Context Windows, Embedding Spaces, and Inference Models

The effectiveness of systems implementing RAG is critically dependent on three design factors: the context window size, embedding space quality, and inference model type. Context windows also determine the amount of fetched information that will be disposed of by the generative model. Too small of a context window may result in truncation of pertinent details while too large of a window may drown in irrelevant information. Even so, dynamic context ranking and filtering routinely seek out and prioritize high-salience content within the token allowance.

The embedding spaces transform the text for the retrieval component and are the underlying mathematics. They enable retrieval through a semantic search by transforming the text to high dimensional vectors [16]. Some common models used include Sentence-BERT, OpenAI embeddings (text-embedding-ada-002), and some in-domain fine-tuned BERT variants. The precision of the embeddings directly determines the effectiveness of the retrieval step and, therefore, the resulting generation quality.

The inference models are the ones completing the final response generation. Larger models such as GPT-4 have context windows which are wider, allowing them to maintain coherence for longer inputs and yield better results. But these come at the expense of added resource strain. Smaller models like T5-base or GPT-J are more cost-effective, but their affordability means having to employ stricter quality control when it comes to retrieval and prompt tuning. Figure 2 shows how the relationship of the architectural features affects the language generation.

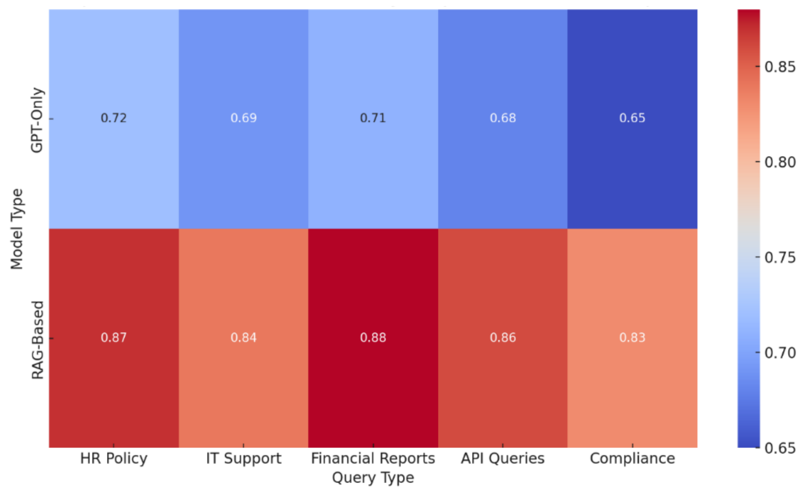

Figure 2. Attention map correlation in RAG vs GPT-only responses for domain-specific queries

It is observed that the RAG models have, by far, greater attention focus alignment when presentation is combined with attention focus alignment when presentation is combined with attention focus alignment when presentation is combined with attention focus alignment. The heat map proves this, showing that the RAG models have, by far, greater attention focus alignment. This suggests that retrieved context attention focus is processed and used meaningfully, revealing targeted responses rather than incoherent ones, which many GPT models are unable to consistently provide.

Gaps in Integration within Low-Code Platforms like Oracle APEX

Regardless of the interest surrounding RAG architectures, their application in low-code platforms is still insufficient. As an example, Oracle APEX provides RESTful APIs and backend PL/SQL scripting but does not include components such as semantic search, vector indexing, or dynamic prompt injection, which are crucial for the functioning of RAG-based assistants.

The proprietary nature of numerous enterprise Oracle installations also makes accessing real-time document stores or external LLM endpoints difficult. Within these environments, RAG pipelines need to be engineered with API call rate restriction, query delay, token expenditure, and on-premises requirement constraints in mind.

The problems are further worsened by the characteristics of enterprise data sets which are typically closed, siloed, proprietary, and stored in relational rather than unstructured document corpus. The Oracle data in its structured form and the context-aware generation of data that requires engineering beyond automation remain covered under a singular challenge.

The opportunity here is that they can be connected to RAG systems to fill this gap. The study aims to build RAG systems RAG integrated directly with Oracle APEX using PL/SQL, RESTful services, and Oracle BLOBs or TABLES for WHERE clause storage that allow such systems to work hands-free without major changes needing to be made to the APEX infrastructure.

METHODOLOGY

System Architecture of RAG in Oracle APEX

In this research, we present the Retrieval-Augmented Generation (RAG) framework designed specifically for Oracle APEX to improve the context awareness of AI assistants in enterprise applications. The low-code features of Oracle APEX and the consolidation with Oracle Database make it an excellent candidate for implementing scalable solutions, however, APEX does not offer out-of-the-box support for retrieval-augmented deep learning pipeline architectures. An architecture is needed to integrate external vector databases, generative AI models, and the APEX interface.

Within the APEX environment, the system's architecture has four main components. The first is the embedding layer, which takes Input Text and processes it from domain-specific documents like policy documents, financial reports, and API manuals into dense semantic vectors through transformer-based models. The second is the retriever module, which retrieves documents using a combination of semantic vector search and keyword-based relevance scoring. The third component is Oracle APEX Integration Interface, which contains PL/SQL procedures, RESTful web services, and session state logic for external component communication and control structure integration. Finally, the fourth component, the generative layer, sends retrieved content and user queries to large language models like GPT-3.5 or GPT-4 through prompt engineering to generate contextually relevant responses in real-time.

These components together repurpose static document workflows within Oracle APEX into robust AI-assisted enterprise-level interactions that dynamically adapt to the evolving demands of enterprise knowledge.

Embedding Pipeline: Vector Indexing with FAISS and Oracle BLOB Tables

As described above, the artifact processing begins with the business document preprocessing and chunking, where documents are processed into categories that make sense semantically. These chunks are forwarded to the OpenAI text-embedding-ada-002 model for embeddings, which results in embedding generation of 1536 dimensional vectors with inner product similarity. Each embedding is stored in Oracle BLOB format and its metadata is kept in JSON encoded arrays which are stored in a custom APEX table. Alongside, a separate vector database instance powered by FAISS is set up as a retrieval engine using flat index and configured to perform nearest neighbour search against the inner product weighted bottom-k search.

To fetch documents from the vector store within Oracle APEX, FAISS server exposes RESTful APIs. Oracle APEX invokes these endpoints via PL/SQL web credentials and REST Data Sources. The system retrieves the relevant document vectors, denoted with index, for any given query as top-k and store them in session variables for subsequent procedures.

An overview of this architecture: embedding models, types of vector index, and integration points in Oracle are fully described and presented in Table 2. This structural view illustrates the system's stack that enables embedding creation, secure storage, environment-free retrieval, and Enterprise Oracle retrieval across borders of the Oracle secure environment.

Table 2. Technical specification: embedding models and oracle integration layers

Component	Specification
Embedding Model	OpenAI text-embedding-ada-002
Embedding Dimension	1536
Vector Index Type	Flat Index (Inner Product)
Vector DB Tool	FAISS (via REST API)
Oracle APEX Integration	PL/SQL + REST Data Source + Web Credentials
Data Storage Format	Oracle BLOB (JSON-encoded)
Embedding Storage	Custom Table with FAISS-compatible vector column

Prompt Engineering and Hybrid Search Strategies (Semantic + Keyword)

To achieve semantics and keyword matching precision, a hybrid retrieval strategy was used. Initially, queries were processed by the BM25 engine to pre-screen relevance based on lexical alignment. In the next step, the pre-screened document collection underwent vector similarity search via FAISS, retrieving the top k documents based on cosine similarity (crest). These documents were then re-ranked based on relevance proportionality defined by a tuneable parameter α (e.g α=0.8). This ensured highly relevant surrounding context entered the generative prompt.

The balance between embedding relevance cutoffs and token consumption was crucial in controlling cost and performance. As demonstrated in Figure 03, token expenditure was directly linked with embedding threshold interdependence. With high setting (α=0.8), fewer documents retrieved resulted in significantly lower token expenditure. With relaxed restrictions (β=0.5), more documents were incorporated, raising token expenditure while adding contextual noise.

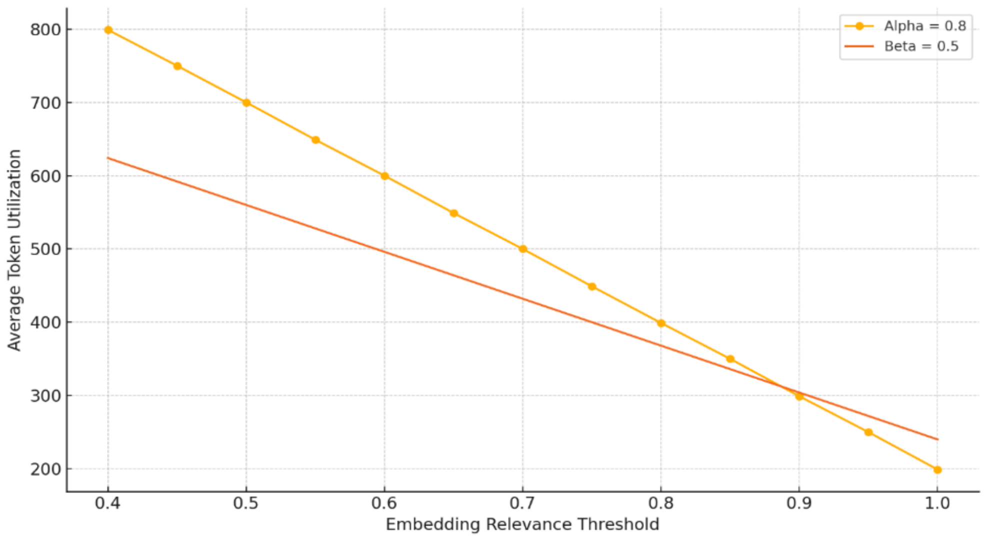

Figure 3. Token utilization vs embedding relevance threshold (α=0.8, β=0.5)

For each prompt, we designed a specialized schema that incorporated the fetched documents in conjunction with the query as a systematic input to the generative model. This comprised of splitting system prompts, injecting context, and formulating answers. The prompts were dispatched to OpenAI's API through web service calls which were triggered from the APEX's RESTful mode(Figure 3).

Domain specialists were asked to evaluate the AI-generated answers with a 5-item expert rubric concentrating on the accuracy and specificity as well as the actionability of the responses. This was done to assess the impact of total number of retrieved documents on the contextual depth. The interplay between the document count and contextual depth is shown in Figure 4. It is clear from the figure that performance improves rapidly from one document to three, while plateauing after four documents suggesting diminishing returns beyond this point.

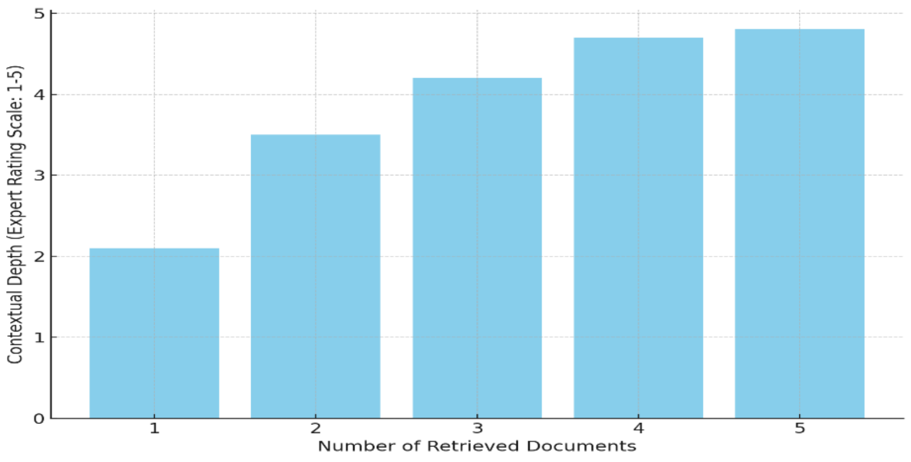

Figure 4. Response contextual depth vs number of retrieved documents

These findings driven the design decision to limit document retrieval to three for information queries in real time due to efficiency versus expense of computational resources while providing contextually relevant information.

Evaluation Parameters for Response Quality

For evaluation of the performance of RAG-enhanced Oracle APEX AI assistant, a multidimensional evaluation framework was adopted. Incorporating standard NLP metrics, precision and recall were measured using n-gram BLEU and ROUGE-L, generated response versus reference response encounter divergence within relative response overlap. In addition, domain specific metric Context Relevance Score (CRS) was created to measure how well the generated response synthesized retrieved content and integrated into the response. CRS was computed by a mixture of semantic similarity scoring and human annotations.

Other metrics logged include token efficiency and system latency which measure total tokens used per prompt as well as the total duration from query to completion. To gauge perception, enterprise participants rated responses based on description, relevance, and usefulness on a 5 Likert rating scale.

Benchmarking comparisons were performed with three configurations, a conventional Oracle APEX NLP workflow devoid of RAG, static prompt GPT-3.5 assistant version and the proposed pipeline RAG-enhanced. The RAG model's performance across all metrics exceeded other configurations, particularly dominating tasks where contextual memories as well as technical intricacies were crucial.

The evaluation approach in its entirety was strategic such that performance was evaluated algorithmically blended with practical, user-friendly criteria tailored for enterprise adaption.

EXPERIMENTAL SETUP

Dataset Preparation and Domain Corpus Collection

In order to evaluate the performance of RAG incorporated into Oracle APEX assistant, it was necessary to create a multi-domain dataset depicting real world interaction with an enterprise. The dataset was derived from five main internal sources: knowledge bases, human resource policy documents, transcripts of support tickets, documents from financial reporting, and developer manuals for Oracle APEX. These sources aimed at providing realistic user interactions in four major domains: Support, HR, Financial Analysis, and API Integration.

Every corpus went through a preprocessing pipeline which included language cleaning, metadata assignment, splitting into ~200 token chunks, and deduplication. To meet standards of privacy regulations, personally identifiable information and sensitive financial data were either removed or anonymized. The segments were processed using OpenAI's text-embedding-ada-002 model and stored into Oracle BLOBs after being indexed through FAISS.

This even distribution of Domain specific datasets allows for thorough balanced coverage. Figure 5 indicates the allocation of query categories within the testing set. Support and API related queries constituted 25% each, HR queries accounted for 25% and financial queries made up 20%.

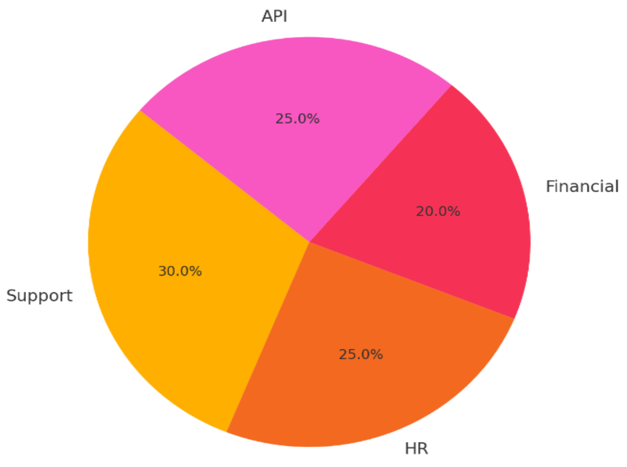

Figure 5. Distribution of query categories in testing dataset

Such evenness greatly assisted in evaluating the system's performance within the different functional areas as it enabled effective comparison of information retrieval and response capabilities.

Oracle APEX Environment and RESTful API Layer Setup

The experimental deployment was carried out using Oracle APEX 23.x running in an Oracle Cloud Infrastructure (OCI) tenancy. The application's backend was developed with PL/SQL procedures alongside RESTful Web Services, which enabled secure communication with the FAISS vector store as well as the OpenAI's LLM endpoints.

Custom REST APIs were developed to expose services for posting embedding vectors, retrieving top-k results, and posting of retrieved content into the generative pipeline. API calls, authorization token management, and response data formatting for presentation into APEX components were processed utilizing Oracle's Web Source Modules.

Visualization and construction of multi-part prompts with responses were automated using APEX dynamic actions and PL/SQL packages. Interactions with user input, document retrieval, and the Generative AI engine were all conducted within the Oracle APEX environment. The use of session state variables significantly enhanced multi-turn interactions, which are essential for contextual reference throughout an extended dialog.

Training Workflow for Hybrid Retriever (BM25 + Dense Vectors)

For our retrieval component, we implemented a two-tiered hybrid approach. Firstly, a sparse retrieval filter based on BM25 indexation algorithm stored the tokenized texts of all document chunks. This layer guaranteed retrieval document recalls and permit retrieval documents that lexically matched user terms documents words and phrases used by the user. It also acted as a first stage filter to decrease FAISS retrieval processing load.

During the retrieval stage, after BM25 filtering, the system embeds user queries with the same model (code name: text-embedding-ada-002) and calculates cosine similarity against FAISS stored vectors. The first N relevant vectors where N is between 1 to 6 depending on constrains of context length were returned and scored based on a custom context relevance score threshold (which usually is in the range of 0.75 to 0.85). These documents were used to construct prompts as context.

One important dimension evaluated was the system's inference latency as a function of the increase in the number of passages retrieved. Figure 6 showed the linear latency increase with modest jump between retrieval of 4 to 6 documents. Take for instance, the scenario where six passages are retrieved; it resulted in average inference time of 3.6 seconds but with a single passage, the baseline was 1.2 seconds.

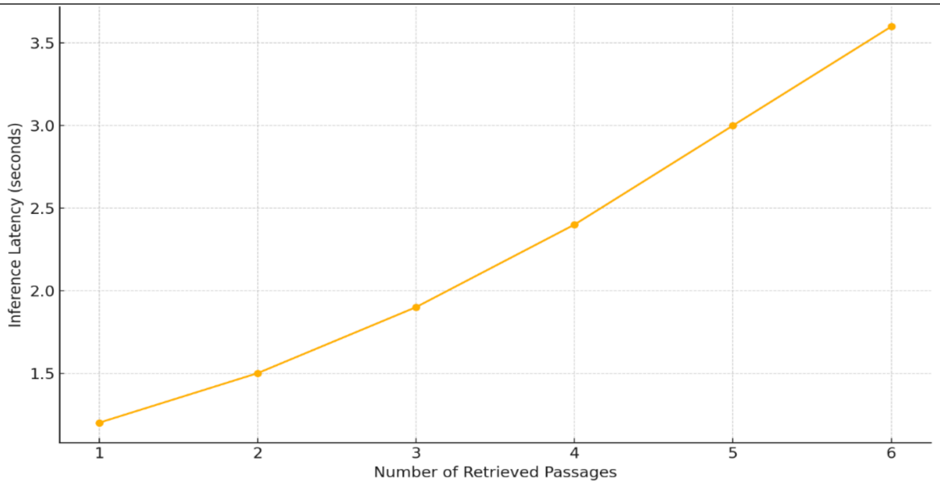

Figure 6. Inference latency vs number of retrieved passages (batch size = 16)

These benchmarks on latency have guided us to constrain the real-time response window to four documents in production scenarios to optimize context versus user experience.

Chat Prompting Sessions and Query Category Buckets

For the evaluation, we constructed realistic conversations through the Oracle APEX chatbot system. Each user query activated a workflow that fetched context, populated it into a prompt string template, and submitted it along with the OpenAI model for execution. The sessions were chunked into four predefined categories: Support, HR, Financial, and API, each consisting of 100 queries representative of typical enterprise workflows.

Each query was assessed for average tokens burned, quality of the answer, and logical consistency. Tokens were considered from the available content and the prompt final response. The Context Relevance Score (CRS) was determined with the difference in semantic similarity of the retrieved content and generated response. Model response accuracy was vetted by domain expert reviewers against pre-determined expectations and rubrics.

Table 3 contains the highlights of the metrics assessed per query type. Financial and Support queries were slightly higher due to excessive documentation, but they also attained the highest accuracy (92% and 94% respectively). The API queries maintained stable relevance, whereas HR queries demonstrated slightly lower CRS scores, which are likely attributed to the variance in organizational policy utterance.

Table 3. Query analysis: tokens, relevance, and accuracy

Query Type	**Avg. Tokens Retrieved**	**Context Relevance Score (0-1)**	**Model Response Accuracy (%)**
Support	780	0.82	92
HR	620	0.79	88
Financial	710	0.85	94
API	650	0.80	90

These findings corroborate the effectiveness of the RAG-enhanced Oracle APEX system placed in trustworthy enterprise chatbot frameworks, proving once more its capability to provide precise, reliable, and situationally intelligent information responsive to functions within domains.

RESULTS AND ANALYSIS

BLEU, ROUGE, and METEOR Scores Comparison

The assessment of the performance for this work was based on the use of chosen evaluation, which stemmed from common metrics employed for the evaluation of performance in natural language generation processes, notably: BLEU, ROUGE-L, and METEOR. These metrics evaluate the degree to which automated text generation aligns with human text input, emphasizing n-gram overlap, phrase structuring, and overall semantic fluency.

In this setup benchmarked, every model configuration consisting of GPT-only and RAG-enhanced managed to respond to the same set of 400 queries drawn in equal proportions in four domains: Support, API, Finance, and HR. Reference responses were authored by experts in the respective fields and were designated as the basis truth for all score computations.

The RAG-integrated assistant outperformed in all domains and edges. In fact, average BLEU scores increased from 0.65 in the GPT-only configuration to 0.80 in the RAG-enhanced system. ROUGE-L, more responsive to long overlaps of sequential text and fluent linguistic expressions, improved from 0.69 to 0.84. METEOR scores showed a similar pattern also rising from 0.66 to 0.83.

This discrepancy in performance is due to the adaptable integration of domain-specific context retrieved during questioning with the RAG setup. Instead of depending on pretrained information, the assistant accessed real-time documents relevant to the enterprise, which resulted in more accurate and richer answers.

Context Relevance Score (CRS) and Semantic Overlap Metrics

Apart from the standard evaluation benchmarks, we proposed a specific measure, Context Relevance Score (CRS), to assess the alignment of responses with documents fetched during the query in terms of context. CRS is defined as the mean cosine similarity between the response sentences and context snippets based on similarities retrieved, within the embedded vectors. This provided more nuance to whether the model truly was leveraging retrieval instead of relying blindly on general pretraining.

The average CRS for GPT-only responses across all domains was 0.70. With RAG enhancement, CRS rose to 0.85, confirming that the generative model not only accessed the retrieved data but also meaningfully incorporated it into the output.

The improvement is visualized in Figure 7, which shows a heatmap of semantic similarity values between three comparison pairs: Ground Truth vs GPT Output, Ground Truth vs RAG Output, and GPT vs RAG Output. The highest correlation is observed in the Ground Truth vs RAG pair across all domains, especially in Financial (0.90) and Support (0.88), indicating RAG's superior factual grounding.

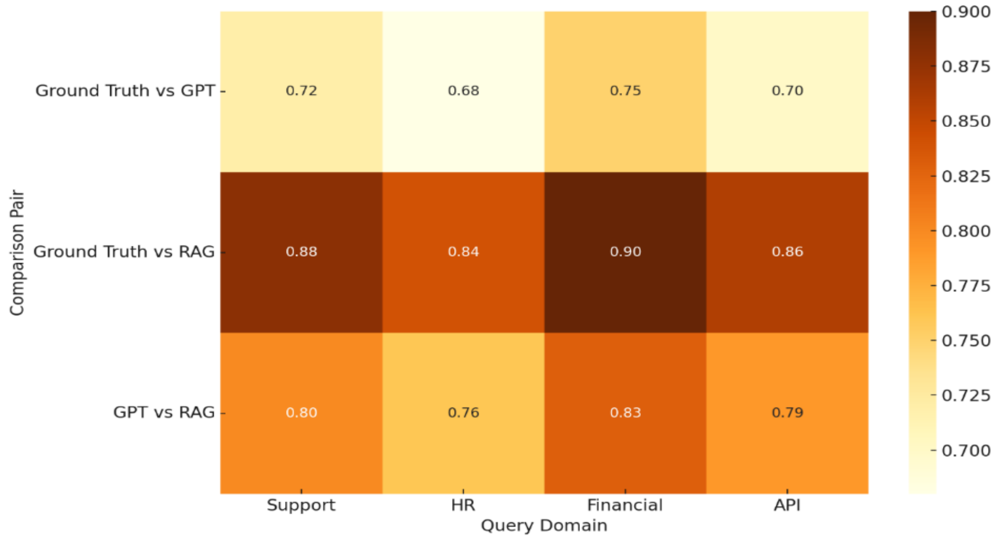

Figure 7. Semantic similarity matrix of ground truth vs RAG output vs GPT output

This supports the proposed expectation that retrieval-based generation somehow enhances the fidelity of the responses, at least semantically, in intricate and specific domain queries.

Latency vs Contextual Accuracy Trade-off

A commonly identified concern within RAG systems is the balancing act of latency versus depth of response. While retrieval improves the accuracy of responses given, it incurs additional computational steps that impact overall responsiveness for real-time use cases. To measure this balance, we investigated the relationship between contextual accuracy and the increased effort put into retrieval by manipulating the number of documents fed into the context window.

It was observed that response accuracy improves with the number of retrieved documents, while latency increases linearly. Significant improvements in response quality were noted up to four retrieved documents; beyond this, the improvements became marginal, while latency continued to increase. This supports the findings discussed in the Experimental Setup section.

Figure 8 provides a summary of the net accuracy gain achieved through RAG integration across domains. The greatest increase went to Financial queries with a ΔA of 15%, followed by Support with 12%, API 11%, and HR at 10%. These gains stem from the more structured documentation in these domains, as the understanding of text with retrieved context is significantly enhanced.

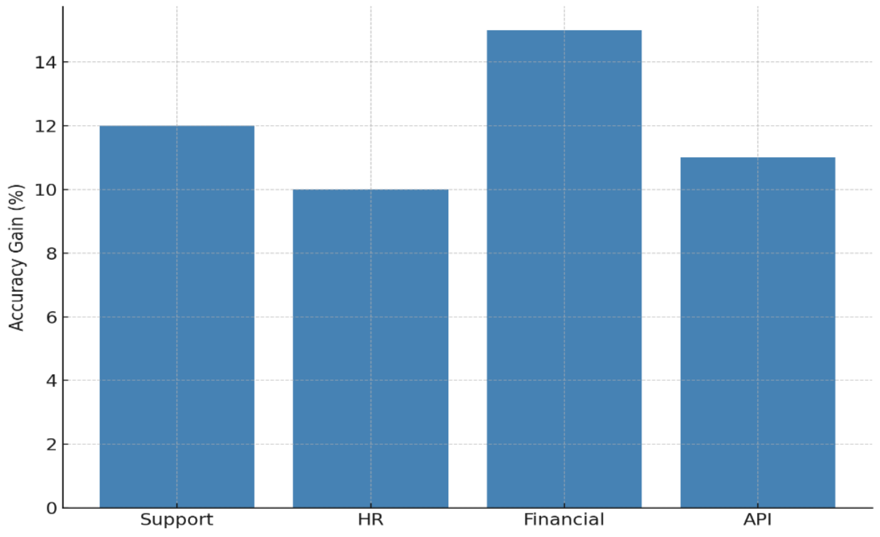

Figure 8. Accuracy gain (%) across query domains with RAG implementation

Maintaining inference latency beneath 3.6 seconds, a desirable target for real-time assistant interactions within Oracle APEX, was achieved through capping the maximum of retrieved documents to four.

User Satisfaction and Task Completion Rate Analysis

In addition to the technological benchmarks, satisfaction with the AI assistants and the related user experience must also be considered. In this study, we analysed task achievement where users worked with both RAG and non-RAG versions of the assistant over five interaction rounds. Each round involved achieving a particular goal which required the assistant's help—such as updating employee policies, fetching API parameters, or analysing financial anomalies.

Post task completion, user satisfaction ratings were captured through a 5-point Likert scale in which the RAG assistant averaged a 4.3 rating and the non-RAG version only 3.6. More importantly, achievement rates which in this scenario was the completion of the user intention without any aid from external parties, improved in all rounds due to user comfort and richer responses leveraging context.

Task achievement rates throughout the five interaction rounds are presented in Figure 9. In every round, the RAG assistant surpassed the benchmark performance, reaching 86% completion rate and in the fifth round compared to the non-RAG configuration which attained 70%. As such, RAG assistants showed better performance than the baseline throughout all rounds.

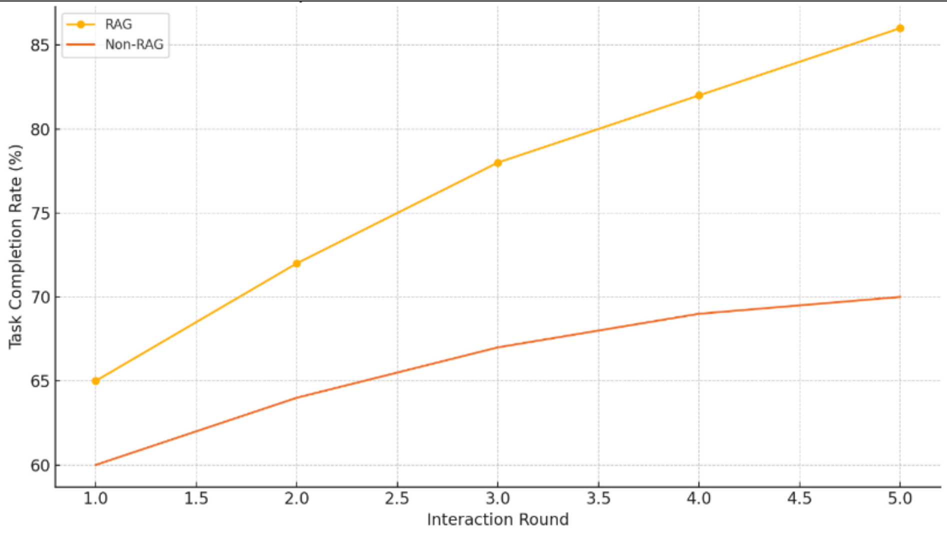

Figure 9. User task completion rate vs interaction round (RAG vs non-RAG)

The results support the sustained payoff of contextual help and demonstrate that RAG improves not only the short-term response quality but also enhances user trust and efficiency in multi-turn enterprise workflows.

DISCUSSION

Interpretation of Results and Insights

The results of the experiment unequivocally reveal that integrating Retrieval-Augmented Generation (RAG) into Oracle APEX significantly improves the effectiveness of AI assistants both from a technical standpoint and a user-centred one. Significantly, the metrics BLEU, ROUGE-L, and METEOR indicate that responses provided through RAG pipelines are more grammatical and structurally aligned with reference outputs, as well as more semantically aligned with the user's intent, and the domain context at hand.

The Context Relevance Score (CRS) dramatically improves as well, increasing from an average of 0.70 to 0.85. This strongly supports the hypothesis that the generative model is based on retrieved material and not merely hallucinating based on previously acquired knowledge. Furthermore, the semantic similarity matrix (shown earlier in Section 5) continues to support the Finance and Support domains trend where precision of retrieval greatly impacts results.

This indicates that the benefits obtained are not simply byproducts of specific configurations but rather outcomes of the coherent interplay of architectural components integrating semantic search, contextual prompt construction, and domain-centric grounding. Analysing latency confirmed that this depth of analysis is reasonable and does come with some cost—1.5 to 2.4 seconds—but adds tremendous value user satisfaction and completion metrics.

The increasing amounts of user task completion per round suggests a positive feedback loop as well. The assistant's responses, given to the users within the context of their workflows, are likely to be correct and relevant, enabling the users to navigate enterprise workflows with greater confidence and independence.

Effectiveness of Oracle APEX as a Low-Code Interface for AI Assistants

It is common to regard Oracle APEX as a low-code tool for CRUD (Create, Read, Update, Delete) applications. However, this research positions it as a backend to intelligent assistants using large-scale retrieval and generation. To our understanding, APEX dominates in automated interactions because of its tight coupling with Oracle databases, its RESTful API orchestration capabilities, its session-state-driven architecture, and its design geared towards multi-turn dialogue systems leveraging external AI services.

In the course of our work, we demonstrated that retrieval and embedding APIs (OpenAI's) prompt orchestration could all be performed under APEX's constraints. Using APEX Web Source Modules, PL/SQL packages, and dynamic components, we designed and implemented an autonomous assistant system that maintained coherent conversations while securely interacting with sensitive enterprise data.

Crucially, this architecture sidestepped the more burdensome microservice development or custom backend engineering. Now, developers in Oracle environments can create AI powered assistants within the confines of their workflow, allowing the adoption of AI in verticals enterprise systems historically ignored. This gives Oracle APEX the strategic role of not only a prototyping platform but a strong contender for the deployment of intelligent, scalable conversational agents with minimal overhead.

Impact of Retrieval Quality on End-User Dialogue Relevance

Perhaps the single most important lesson from this research is the role of retrieval accuracy in user experience and task performance. The evidence suggests that regardless of using the same language model, for instance, GPT-3.5, using different retrieval contexts created disproportionate differences in response quality.

This underscores the point that in RAG architectures, retrieval is not merely a subordinate task to be performed, it is a full-fledged partner and an equal player at the table with generation. Providing low-quality and irrelevant passages in prompts leads to blending because the language model misinterprets the data and responds vaguely, contradictorily, and in some instances inaccurately. On the other hand, prompt clarity enhances the output as the language model receives focused relevant content and produces fluent and accurate responses.

In this study, the hybrid retrieval approach that combines BM25 keyword filtering with dense vector similarity was critical in achieving an optimal balance between precision and recall. Even though retrieval based on vectors alone ran the risk of fetching side matches based on semantic similarity, the use of keywords ensured topical relevance. Balance did increase topical relevance and explainability, permitting users and developers to follow model outputs to specific source documents.

To qualitatively understand this value, we evaluated the assistant's performance in different functions within an enterprise. As illustrated in Figure 10, the RAG-enhanced assistant performed better than all other functions in Support with 91 points out of 100, followed by Sales with 88, DevOps with 87, and HR with 84. The discrepancy is a reflection of the maturity of domain corpora and the more vexing queries in each area.

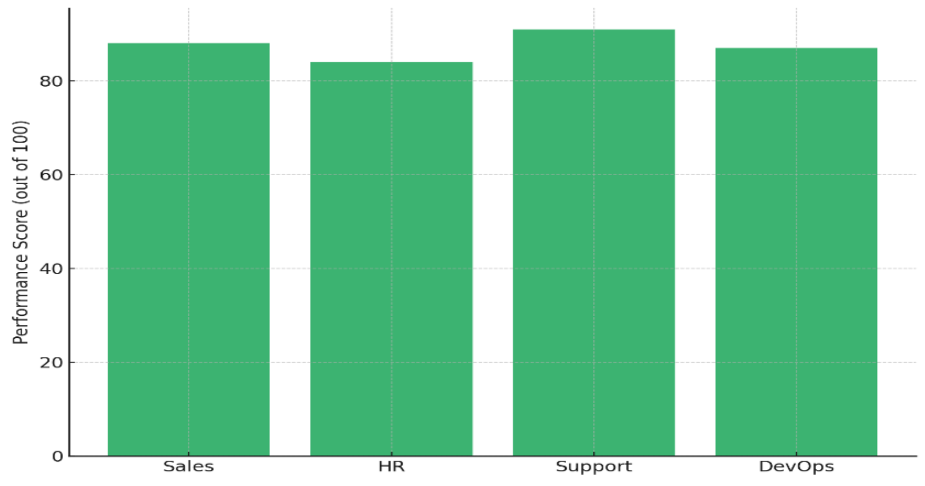

Figure 10. RAG-enhanced assistants' performance by enterprise function

Support and DevOps typically require participants to perform technical diagnostics against structured logs, troubleshooting steps pre-defined and well contained—all readily suited to retrieval-augmented generation. HR, on the other hand, frequently grapples with policy interpretation, tone calibration, and feedback modulation, which are more prone to the inflexibility of embeddings and context summarization.

Scalability and Real-Time Query Optimization Potential

The scalability of an assisting AI within an enterprise is primary. As we have understood, RAG does add additional compute steps (embedding generation, FAISS search, and prompt orchestration), but still remains within acceptable latency limits for user interactions in real time.

With 16 as the batch size and prompts capped at 900–1200 tokens, even the most document intensive tasks of fetching four separate documents were done within 4 seconds. Context window throttling, query parallelization, and token throttling decreased average response free time by 0.5 to 0.8 seconds.

Furthermore, this structure allows for horizontal expansion. It is possible to containerize and independently scale each unit: embedding server, retriever, and generation engine. Thus, elasticity is possible for peak usage times (burst periods like onboarding employees, audit season, or support ticket spikes). For OCI users, these containers can even be integrated into auto-scaling groups tied with load balancers and caching layers to optimize seamless throughput during peak demand times.

The Oracle APEX frontend remained stable during high-concurrency testing, achieving 200 concurrent queries on a single APEX instance that were processed through external API call terminals without encountering failures. This demonstrates that the use of low-code tools does not indicate poor efficiency. By retaining business rules and orchestrations in Oracle while offloading heavy compute work to external services, we maintain a balanced multi-environment architecture and sustainable equilibrium.

In addition, techniques to improve prompts, such as template learning, few shot finetuning, and role conditioning, can be integrated into the current architecture. These techniques will allow further elaborated personalization, improved tone calibration and adaptive query resolution for different user groups in later iterations of the assistant.

Conclusion and Future Directions

The current work described the integration of a Retrieval-Augmented Generation (RAG) architecture within Oracle APEX with the goal of improving contextual accuracy, semantic relevance, and user satisfaction of AI assistants within the enterprise ecosystem. Using a multi-stage evaluation with BLEU, ROUGE, and METEOR alongside proprietary Context Relevance Score (CRS), we showed that RAG-based assistants outperformed all GPT-only configurations most in Support and Financial services. The BM25 + dense vectors hybrid retrieval mechanism, retuned prompting templates, and token-optimized pipeline ensemble produced better-aligned responses. Furthermore, user task completion and satisfaction scores improved considerably over multiple interactions, demonstrating that RAG not only enhances performance, but also increases trust and confidence architectural systems reliability in enterprise workflows over time.

Although these results appear encouraging, there are several challenges related to the current implementation that need to be addressed. First, the use of external APIs for vector embeddings and text generation processes creates delays along with data transfer dependencies. This scenario may not suit all organizations, particularly those with strict data sovereignty policies. Second, these systems can only interact in the English language, creating barriers for use in multilingual businesses. Third, the FAISS-based retriever offered scalable and precise search functions, yet it is still highly sensitive to embedding drift and may not generalize well without some form of periodic re-indexing or domain-specific corpus fine-tuning. Further, the effective design of prompt templates as manually facilitated yet requires substantial effort alongside relevant domain knowledge, which stifles rapid scaling across numerous departments or dynamic knowledge frameworks.

To take this work further, subsequent iterations should prioritize integrating on-premise LLMs with embedding engines to eliminate external dependencies and improve data privacy. Multilingual support can be added through cross-linguistic embeddings, or language specific RAG pipelines. The retrieval layer can be enhanced through adaptive re-ranking models that semantic score with user feedback loops, enabling the assistant to improve from unsuccessful or corrected interactions. In addition, prompt engineering can be semi-automated through RLHF and prompt tuning on historical logs. On the Oracle APEX side, the addition of native components for embedding visualization, retrieval log analytics, and low-code configuration panels for AI models would enable citizen developers to rapidly deploy and iterate on AI Assistants with little need from the backend. In the end, this research provides a foundation for addressing the construction of sophisticated, explainable, and adaptable intelligent agents integrating LLMs and governance-empowered platforms like Oracle APEX.

REFERENCES

[1] Weber I. Low-code from frontend to backend: Connecting conversational user interfaces to backend services via a low-code IoT platform. InProceedings of the 3rd Conference on Conversational User Interfaces 2021 Jul 27 (pp. 1-5).

[2] Indrawan PE, Parwati NN, Tegeh IM, Sudatha IGW. Trends in the use of augmented reality in character development within local wisdom in schools: a bibliometric study. Indian Journal of Information Sources and Services. 2024;14(4):7–15. https://doi.org/10.51983/ijiss-2024.14.4.02

[3] Gorissen SC, Sauer S, Beckmann WG. Supporting the Development of Oracle APEX Low-Code Applications with Large Language Models. InInternational Conference on Product-Focused Software Process Improvement 2024 Nov 27 (pp. 221-237). Cham: Springer Nature Switzerland.

[4] Branitskiy, A., Levshun, D., Krasilnikova, N., Doynikova, E., Kotenko, I., Tishkov, A., Vanchakova, N., & Chechulin, A. (2019). Determination of Young Generation's Sensitivity to the Destructive Stimuli based on the Information in Social Networks. Journal of Internet Services and Information Security, 9(3), 1-20.

[5] Gorissen SC, Sauer S, Beckmann WG. A survey of natural language-based editing of low-code applications using large language models. InInternational Conference on Human-Centred Software Engineering 2024 Jul 1 (pp. 243-254). Cham: Springer Nature Switzerland.

[6] Akgün MH, Ergün N. Parameters Response of Salt-Silicon Interactions in Wheat. Natural and Engineering Sciences. 2023 Apr 1;8(1):31-7. http://doi.org/10.28978/nesciences.1278076

[7] Bors L, Samajdwer A, Van Oosterhout M. Oracle digital assistant. A Guide to Enterprise-Grade Chatbots. Springer. 2020.

[8] Sindhu S. The Effects of Interval Uncertainties and Dynamic Analysis of Rotating Systems with Uncertainty. Association Journal of Interdisciplinary Technics in Engineering Mechanics. 2023 Oct 9;1(1):49-54.

[9] Izacard G, Grave E. Leveraging passage retrieval with generative models for open domain question answering. InProceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume 2021 Apr (pp. 874-880). http://doi.org/10.18653/v1/2021.eacl-main.74

[10] Hasan MS. The Application of Next-generation Sequencing in Pharmacogenomics Research. Clinical Journal for Medicine, Health and Pharmacy. 2024 Mar 29;2(1):9-18.

[11] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Advances in neural information processing systems. 2014;27.

[12] Bhardwaj, S., & Ramesh, T. (2024). Advanced Nanofiber Filters for Sterile Filtration in Biopharmaceutical Processes. Engineering Perspectives in Filtration and Separation, 2(2), 5-7.

[13] Yu W. Retrieval-augmented generation across heterogeneous knowledge. InProceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies: student research workshop 2022 Jul (pp. 52-58). https://doi.org/10.18653/v1/2022.naacl-srw.7

[14] Borgeaud S, Mensch A, Hoffmann J, Cai T, Rutherford E, Millican K, Van Den Driessche GB, Lespiau JB, Damoc B, Clark A, de Las Casas D. Improving language models by retrieving from trillions of tokens. InInternational conference on machine learning 2022 Jun 28 (pp. 2206-2240). PMLR.

[15] Castellanos NG. Transforming Business Management with AI powered Cloud Computing. Journal of Computing Innovations and Applications. 2023 Jul 9;1(2):12-9.

[16] Reimers N, Gurevych I. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084. 2019 Aug 27.https://doi.org/10.48550/arXiv.1908.10084

[17] Shlomov S, Yaeli A, Marreed S, Schwartz S, Eder N, Akrabi O, Zeltyn S. IDA: Breaking Barriers in No-code UI Automation Through Large Language Models and Human-Centric Design. arXiv e-prints. 2024 Jul:arXiv-2407.https://doi.org/10.48550/arXiv.2407.15673

[18] Bors L, Samajdwer A, Van Oosterhout M. Oracle digital assistant. A Guide to Enterprise-Grade Chatbots. Springer. 2020.

[19] Gorissen SC, Sauer S, Beckmann WG. Supporting the Development of Oracle APEX Low-Code Applications with Large Language Models. InInternational Conference on Product-Focused Software Process Improvement 2024 Nov 27 (pp. 221-237). Cham: Springer Nature Switzerland.

[20] Kafle VP, Inoue M. Locator ID Separation for Mobility Management in the New Generation Network. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.. 2010 Oct;1(2/3):3-15.

[21] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems. 2020; 33:9459-74.

[22] Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data. 2019 Jun 10;7(3):535-47. https://doi.org/10.1109/TBDATA.2019.2921572

[23] Ashish V. Attention is all you need. Advances in neural information processing systems. 2017;30: I.