International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Intelligent Semantic Search Framework Using AI and ServiceNow Platform

Dolly Motwani
Department of Science and Technology, G.H.Raisoni Skill Tech University, Nagpur, India

ABSTRACT
Enterprise ServiceNow instances accumulate large volumes of documents, making precise, timely retrieval difficult through keyword search alone. A Retrieval-Augmented Generation (RAG) approach—combining embeddings, a vector store, semantic search, and a large language model (LLM)—enables users to upload a document and ask natural-language questions grounded in its content. This paper describes an end-to-end implementation using ServiceNow Virtual Agent (VA) for interaction, a PDF-to-text extraction step, Gemini-AI embeddings for vectorization, Qdrant as the vector database, and an Gemini- AI chat model for answer generation, following an eight-step pipeline from document capture to response delivery . We also propose an evaluation framework, borrowing the "confusion matrix as foundation" measurement mindset from the provided demo paper , and adapt it to RAG quality, faithfulness, and operational performance.

Keywords: ServiceNow, Virtual Agent, RAG, embeddings, semantic search, vector database, Qdrant, Gemini-AI.

This is an open-access article under the CC-BY 4.0 license

1. Introduction

ServiceNow is an enterprise platform used to streamline business operations, empower IT departments, and improve end-user experiences through standardized workflows and a common system of record . In practice, organizations use it not only for service management and case handling, but also to store and exchange large volumes of operational content—policies, runbooks, SOPs, implementation notes, and attachments tied to records. As this content grows, "finding the right information at the right time" becomes a recurring challenge; users often rely on keyword searches, manual navigation, or tribal knowledge, which can be slow and error-prone .

This paper addresses that challenge with a "Chat with Your Document" capability embedded directly into the ServiceNow experience via Virtual Agent (VA). The goal is to let a user upload a document and ask questions in natural language—similar to "chatting with a trusted colleague"—while ensuring answers remain grounded in the uploaded document's content rather than generic model priors . The approach combines OpenAI embeddings and semantic search to retrieve the most relevant passages and then uses an LLM to generate an answer using those retrieved passages as context .

Technically, the solution follows a Retrieval-Augmented Generation (RAG) pipeline implemented with ServiceNow-native orchestration and external AI services. The ingestion path starts when the VA topic captures a PDF through the VA **File Picker** component . The document is converted to text (ConvertAPI), split into smaller chunks to manage model input limits, embedded via OpenAI

embeddings, and stored in a vector database (Qdrant) . At question time, the user's query is embedded and used to perform semantic search in the vector store to retrieve the **top 3** most similar chunks as "context," which are then passed to an OpenAI chat model (example: gpt-3.5-turbo) to produce the final response . On the ServiceNow side, external calls are managed as **Outbound REST Messages** , core logic is centralized in a reusable **Script Include** (example: gptUtils) with an anti-reprocessing control via a sys_properties flag (example: gpt.attachment.embeded) , and the VA topic loops Q&A until the user types exit, calling chatWithDocument(...) each turn .

In summary, this work positions ServiceNow as the workflow and experience layer (secure entry point, orchestration, and conversational UI) while leveraging embeddings + semantic retrieval to make large documents "queryable by meaning," enabling faster, more intuitive access to information already residing within ServiceNow-managed processes .

1.1 Motivation

ServiceNow is widely used to streamline business operations and improve user experiences, but as organizations accumulate large volumes of data, knowledge articles, and record attachments, simply "finding the right information at the right time" becomes a major productivity and quality bottleneck . The motivation for "Chat with Your Document" is to replace brittle keyword-and-folder navigation with a natural-language experience where users can ask questions "like chatting with a trusted colleague," quickly retrieve the most relevant passages, glean insights, and make informed decisions without wrestling with complex search queries or sifting through documents . Embeddings and semantic search enable this by representing document chunks as vectors and retrieving meaningfully similar context even when wording differs, which can then be used to ground an LLM's response in the uploaded content and "revolutionize the ServiceNow experience" into intelligent document retrieval .

2.1 Contributions:

1. A reference architecture and implementation flow for "Chat with Your Document" in ServiceNow VA .
2. Concrete configuration elements (Outbound REST Messages, vector store setup, VA flow loop) aligned to the implementation guide .
3. An evaluation plan inspired by the demo paper's metric-oriented structure (accuracy/confusion-matrix-centric thinking) , adapted to RAG.

2 Related work

4. As in the demo paper's Section 2, which surveys prior approaches before presenting the proposed method , this section situates "Chat with Your Document" in ServiceNow within the broader literature and common industry patterns for conversational search, semantic retrieval, and retrieval-augmented generation (RAG).
5. **Conversational access to enterprise knowledge.** A recurring theme in enterprise platforms is reducing time spent searching across large, heterogeneous knowledge bases by providing a natural-language interface. In ServiceNow, Virtual Agent (VA) provides the interaction layer needed to capture user intent and orchestrate backend actions; the provided implementation uses VA's File Picker for document capture and a Q&A loop (exit-based) to repeatedly answer questions against the same uploaded content . This aligns with a broader shift from "search UI" to "task-oriented dialogue," where the conversation flow becomes the primary user experience and integration point.
6. **Semantic search with embeddings vs. lexical keyword search.** Keyword search is brittle when users paraphrase, use synonyms, or lack domain terminology. Embedding-based semantic search addresses this by mapping text into vectors where similarity reflects meaning; the

provided guide explicitly frames embeddings, vector stores, and semantic search and then uses that stack to retrieve the top-matching passages for a query . This is consistent with the general semantic-retrieval approach used across modern "chat with document" systems: chunk text, embed chunks, store embeddings, embed the query, then retrieve top-*k* chunks as context .

7. **Retrieval-Augmented Generation (RAG).** RAG combines retrieval (non-parametric memory) with generation (parametric LLM) to improve factuality, specificity, and updateability in knowledge-intensive tasks. The foundational RAG work by Lewis et al. (2020) formalized this retrieval+generation framework and showed performance gains on open-domain QA and related benchmarks by conditioning generation on retrieved documents rather than relying purely on model parameters [2] [5]. The ServiceNow implementation in the provided guide follows the same core pattern at an applied level: retrieve top-3 semantically similar chunks from a vector database and pass those chunks as "Context" into an OpenAI chat model to produce an answer .
8. **Vector databases for scalable retrieval.** Vector databases operationalize semantic search by storing embeddings alongside payload text and providing fast nearest-neighbor lookup. The provided implementation uses Qdrant, configured with cosine distance and a 1536-dimensional vector size (matching the referenced OpenAI embedding model), and supports insert/search operations via REST calls . This mirrors the broader ecosystem pattern where vector DBs (e.g., Qdrant/Pinecone/others) are treated as reusable retrieval infrastructure decoupled from the application channel.
9. **System integration patterns on the Now Platform.** A distinguishing aspect of this work (vs. generic RAG prototypes) is the ServiceNow-native integration approach: external AI services are called via Outbound REST Messages; orchestration and business logic live in Script Includes; and VA acts as the user-facing channel that repeatedly calls a server-side "chatWithDocument" function . This pattern is important in enterprise contexts because it supports centralized configuration (endpoints/headers), reuse across experiences, and controlled execution inside platform governance.
10. **Evaluation framing (measurement-first).** While the demo paper's domain differs, its "related work → method → experiments → metrics" structure highlights the importance of comparative evaluation and metric discipline . For ServiceNow RAG systems, that translates into comparing lexical search vs. LLM-only vs. RAG, and measuring both retrieval quality (e.g., whether the right passage is retrieved) and answer grounding (whether the answer is supported by retrieved context), since RAG's value proposition is precisely to reduce unsupported generation while improving relevance.

3. Research Methodology (modeled on the demo paper)

3.1 Problem statement

The demo paper frames methodology by first stating a concrete "problem statement," then describing a stepwise proposed system and algorithm . Analogously, the problem addressed here is: **ServiceNow users often have access to important documents (e.g., PDFs attached to records), but extracting the right answer quickly is hard with manual reading and keyword search; we need a conversational way to ask questions and get document-grounded answers inside ServiceNow.** This work therefore implements a "Chat with Your Document" capability where answers are generated **from semantically retrieved document passages** rather than from the LLM alone .

3.2 Research design and approach

This study follows an implementation-driven methodology: we design, configure, and validate a ServiceNow-based prototype that enables "chat with your document" using a Retrieval-Augmented

Generation (RAG) pipeline—semantic retrieval over embedded document chunks, followed by LLM-based answer generation grounded in retrieved context .

3.3 System architecture (components)

The prototype is composed of (1) a ServiceNow conversational front end (Virtual Agent File Picker + Q&A loop), (2) a document text extraction service (ConvertAPI), (3) an embedding service (OpenAI embeddings), (4) a vector database for semantic retrieval (Qdrant), and (5) an LLM for response generation (OpenAI chat model) .

3.4 End-to-end method (pipeline)

The methodology operationalizes the RAG workflow in eight steps (mirroring the demo paper's "high-level steps"): (1) collect a user document via ServiceNow VA File Picker, (2) extract text from the PDF using ConvertAPI, (3) split the extracted text into smaller chunks to fit model limits, (4) generate embeddings for each chunk using OpenAI embeddings, (5) store chunk vectors + text payloads in Qdrant, (6) embed the user's question and run semantic search to retrieve the top 3 most similar chunks, (7) pass the user question plus retrieved chunks as "context" into an OpenAI chat model (e.g., gpt-3.5-turbo), and (8) display the model's answer back in the Virtual Agent conversation .

3.5 Text chunking strategy

To support retrieval precision and model input constraints, the extracted text is segmented into fixed-size windows (e.g., 512 units) with overlap (e.g., 100) to preserve continuity across boundaries, then stored per-chunk as a payload alongside its embedding .

3.6 Retrieval configuration (semantic search)

At query time, the user prompt is embedded using the same embedding model and used as the search vector in Qdrant; the system retrieves a small set of highest-similarity chunks (limit = 3) to form the context provided to the LLM .

3.7 Generation step (grounded answering)

The retrieved context is injected into an OpenAI chat completion request (model example: gpt-3.5-turbo) so the response is conditioned on the document-derived passages rather than only the model's internal knowledge .

3.8 ServiceNow implementation method (orchestration and controls)

All external calls (ConvertAPI, OpenAI embeddings/chat, Qdrant search/upsert) are executed from ServiceNow via Outbound REST Messages and orchestrated through a reusable Script Include (utility layer). The VA topic invokes the utility method (e.g., "chatWithDocument") in a loop until the user exits, enabling multi-turn Q&A over the same uploaded file . To reduce redundant processing, the demo implementation includes a simple "already embedded" check using a ServiceNow property flag before re-indexing a document .

3.9 System overview (end-to-end approach)

We implement a Retrieval-Augmented Generation (RAG) pipeline inside a ServiceNow Virtual Agent experience: the user uploads a PDF via **VA File Picker**, the PDF is converted to text (ConvertAPI), text is chunked, embeddings are created with OpenAI, embeddings + text are stored in a vector database (Qdrant), then user questions are answered by retrieving the **top 3** most similar chunks via semantic search and sending those chunks as "Context" to an OpenAI chat model (e.g., `gpt-3.5-turbo).

3.10 Core components (and roles)

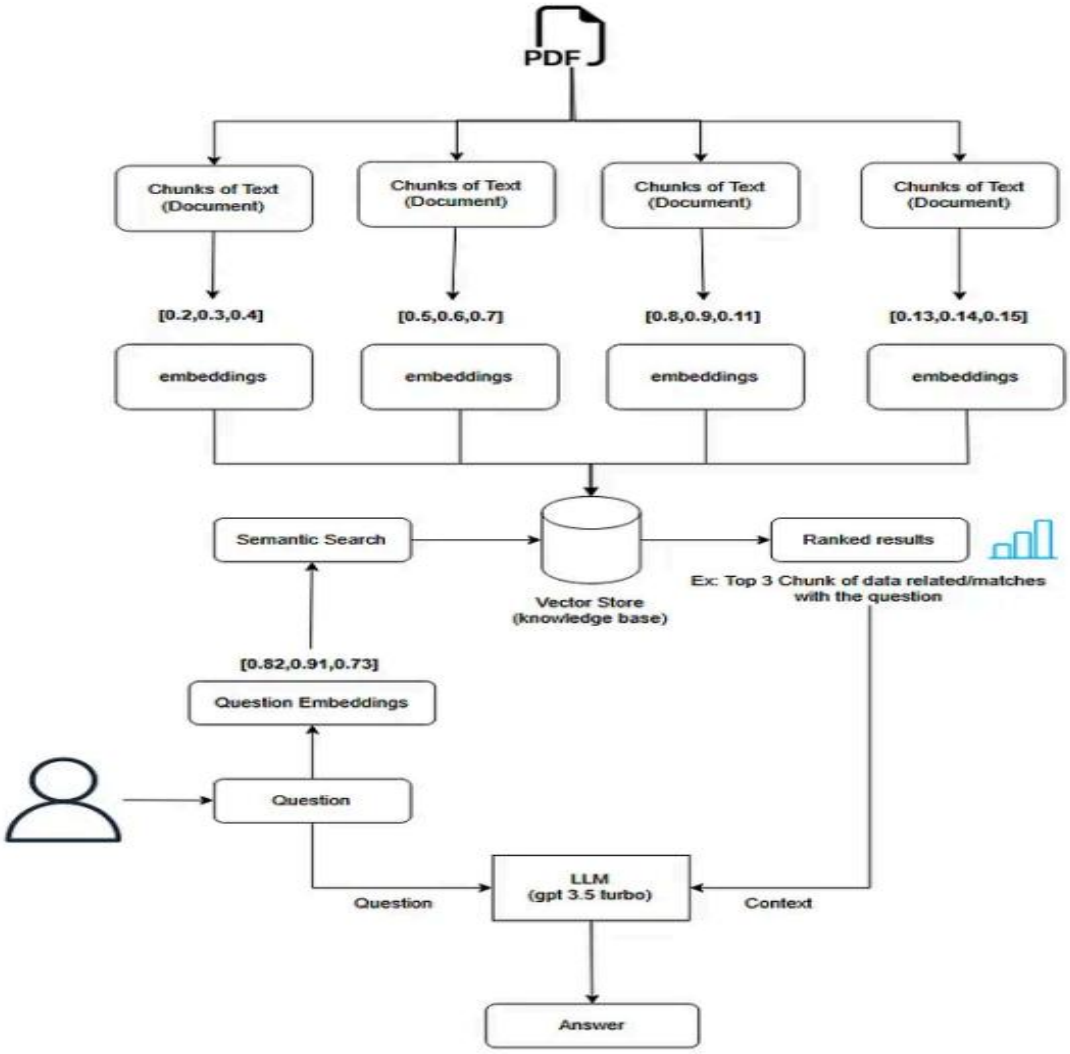
- **ServiceNow Virtual Agent (VA):** captures the document and runs a Q&A loop until the user types exit .
- **ConvertAPI:** extracts text from PDF (PDF → TXT) to enable downstream chunking and embedding .
- **OpenAI Embeddings:** generates vectors for each chunk and for the user's question (example model shown: text-embedding-ada-002) .
- **Qdrant (vector DB):** stores vectors and supports semantic search; collection setup uses size: 1536 with distance: "Cosine" for OpenAI text embeddings.
- **OpenAI Chat model:** generates the final answer using the retrieved context (example model shown: gpt-3.5-turbo) .
- **ServiceNow orchestration via Outbound REST + Script Include:** external calls are implemented as Outbound REST Messages , and the VA flow invokes a server-side utility (gptUtils().chatWithDocument(...)) to run semantic search and return an answer .

Combined Analysis: why embeddings + semantic search + LLM together

Embeddings + Qdrant provide **meaning-based retrieval** (semantic search) to locate the most relevant passages even when the user's wording differs from the document . The LLM then performs **language understanding and answer composition**, but is constrained by the retrieved "Context," which is explicitly constructed from the **top 3** matching chunks . This combination is the core methodological choice: retrieval improves relevance/grounding, while generation improves usability and readability.

3.11 Execution flow (how the prototype runs)

1. **Document intake:** user uploads a PDF using VA File Picker .
2. **Text extraction:** ConvertAPI converts the PDF into text .
3. **Chunking:** extracted text is split into smaller chunks to (a) focus context to the question and (b) respect LLM character limits .
4. **Embedding & storage:** chunks are embedded and stored in Qdrant .
5. **Question-time retrieval:** the user's question is embedded and searched against Qdrant; the REST "semantic_search" call is configured with limit: 3 , aligning with "retrieve top 3 text chunks" .
6. **Answer generation:** the retrieved chunks are combined into "Context" and sent with the question to the chat model .
7. **Conversation loop:** VA collects the next user input; a decision step exits on exit, otherwise it calls gptUtils().chatWithDocument(vaInputs.user_input) again and displays gptOutput .



3.12 Design considerations (method choices and trade-offs)

- **Chunking necessity:** required to keep only question-relevant context and to manage LLM character limits .
- **Top-k retrieval (k=3):** limits context size and reduces noise while still giving enough coverage for most questions .
- **Vector DB configuration:** Cosine distance and size: 1536 must match the embedding model output dimensionality .
- **ServiceNow implementability:** using Outbound REST Messages centralizes external integrations, and a Script Include allows the VA flow to call one reusable method (chatWithDocument) .

3.13 Proposed algorithm (ServiceNow "Chat with Your Document")

Following the demo paper's pattern of listing an explicit "Proposed Algorithm" with step-by-step strategy , the proposed algorithm here is:

Input: PDF document + user question (natural language)

Output: document-grounded answer displayed in ServiceNow VA

Steps:

1. Capture PDF from user using VA File Picker .
2. Convert PDF to text via ConvertAPI .
3. Split extracted text into chunks .
4. Create embeddings for all chunks using OpenAI embeddings .
5. Upsert chunk vectors + payload text into Qdrant (1536-dim, cosine) .
6. Embed the user question and run Qdrant semantic search to retrieve the **top 3** chunks .
7. Concatenate retrieved chunks into "Context," pass Context + question to the chat model (e.g., gpt-3.5-turbo) .
8. Return the generated answer to VA; repeat until the user types exit .

4. Research Methodology

4.1 Approach

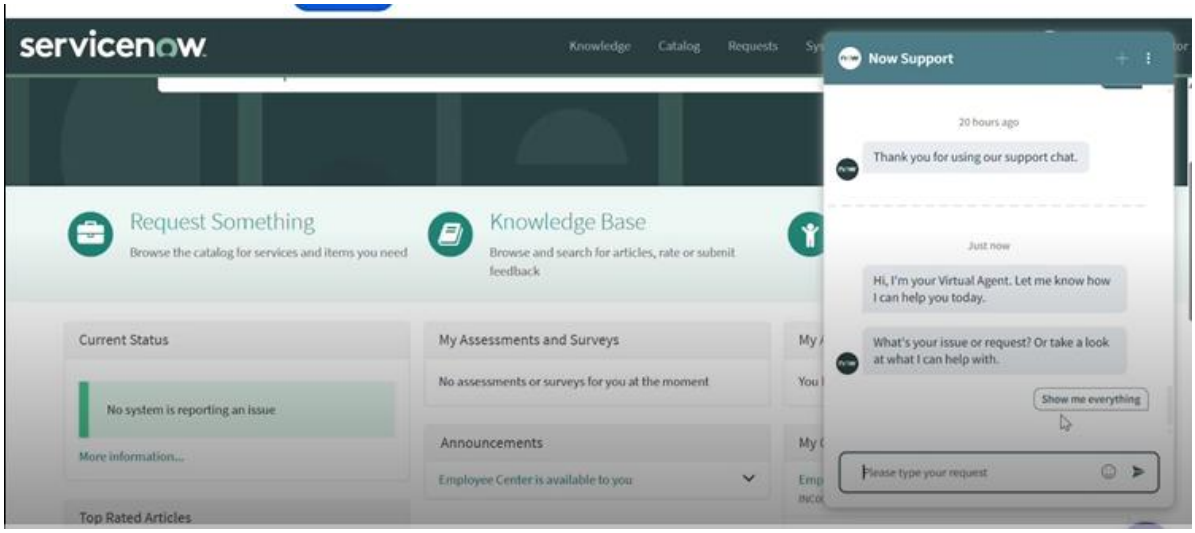
We use an implementation-based research approach by building and validating a ServiceNow prototype that enables "chat with your document" through a Retrieval-Augmented Generation (RAG) pipeline: semantic retrieval over embedded document chunks followed by LLM answer generation grounded in retrieved context .

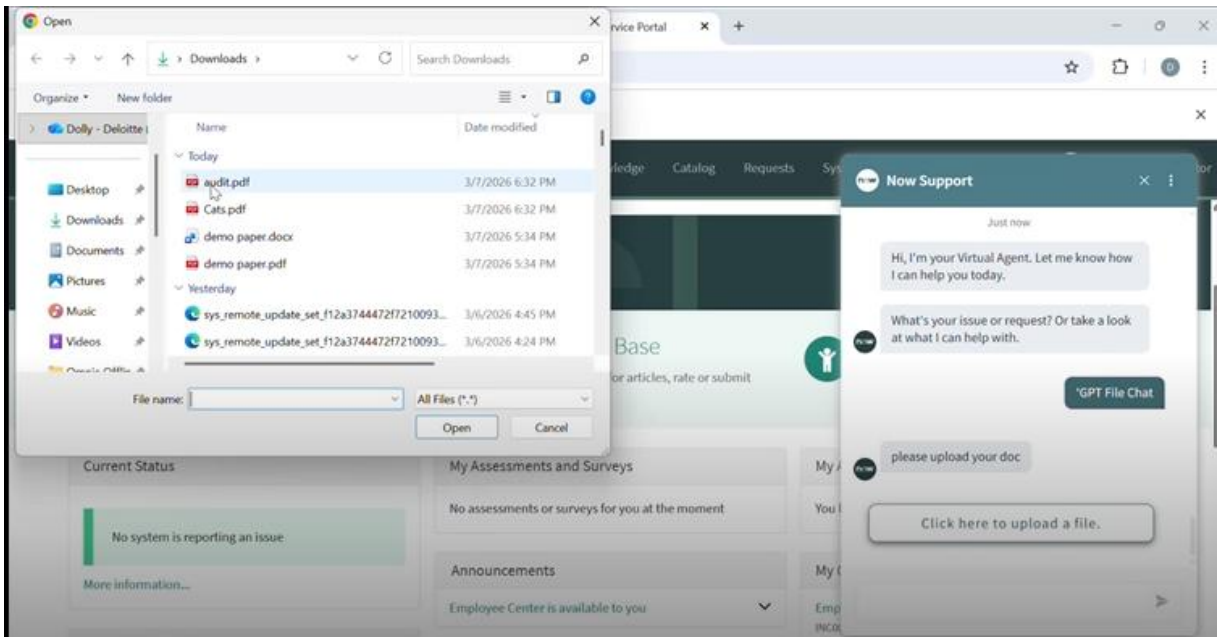
4.2 System overview

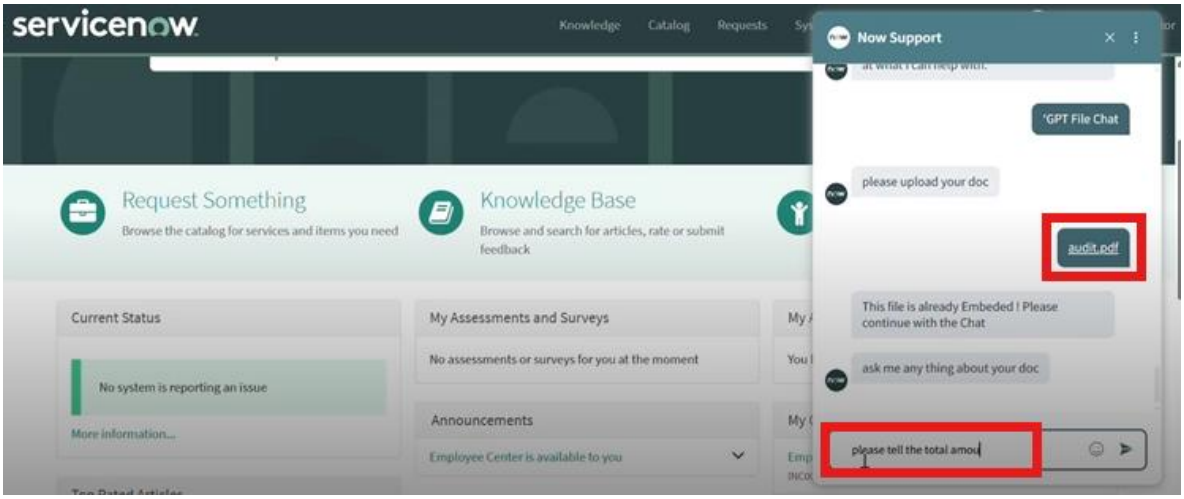
The end-to-end system consists of (1) ServiceNow Virtual Agent (VA) for user interaction and document intake via the **File Picker** (PDF-focused), (2) **ConvertAPI** to extract text from PDFs, (3) **OpenAI embeddings** to vectorize text chunks, (4) **Qdrant** as the vector store, and (5) an **OpenAI chat model** (e.g., gpt-3.5-turbo) to generate answers .

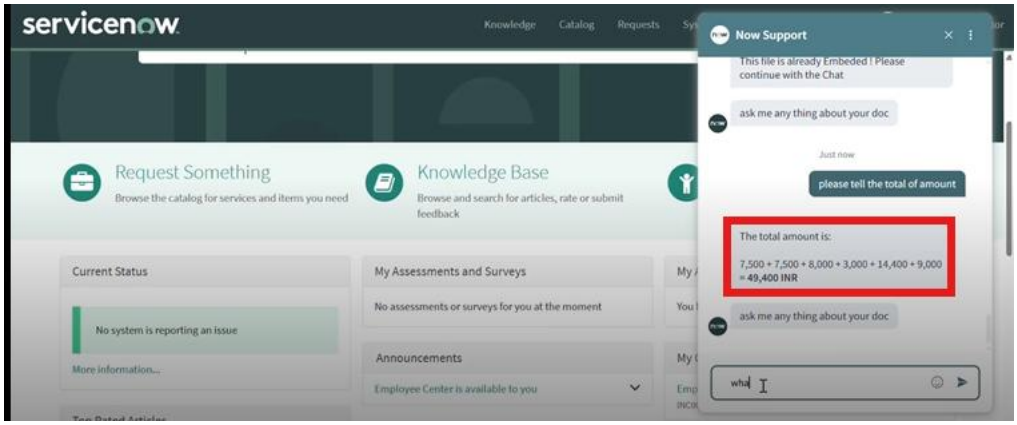
4.3 Document processing and indexing method

After a PDF is uploaded through VA, text is extracted using ConvertAPI (PDF → TXT) . The extracted text is split into smaller chunks to (a) focus context relevant to the user's question and (b) stay within LLM character limits . Each chunk is embedded with OpenAI embeddings (example model shown: text-embedding-ada-002) and stored alongside its source text in Qdrant .









4.4 Retrieval and answer generation method

For each user question, the question is embedded using the same embedding model and used to perform semantic search in Qdrant, retrieving the **top 3** highest-ranked chunks; these chunks are combined to form the "Context" . The question plus Context are then passed to an OpenAI chat

model (example shown: gpt-3.5-turbo) to generate a document-grounded response displayed back to the user .

4.5 ServiceNow implementation/orchestration

External integrations are executed from ServiceNow using Outbound REST Messages (OpenAI embeddings/chat and ConvertAPI), while vector operations are handled by Qdrant endpoints; Qdrant is configured to match OpenAI embedding dimensionality with cosine distance . The VA conversation runs a Q&A loop (exit-based) that repeatedly calls the server-side orchestration (e.g., a Script Include utility) to retrieve context and generate answers .

4.6 Proposed algorithm (summary)

1. Upload PDF in VA File Picker → 2) Convert PDF to text (ConvertAPI) → 3) Chunk text → 4) Embed chunks (OpenAI embeddings) → 5) Store vectors+payload in Qdrant → 6) Embed question → 7) Semantic search (top-3 chunks) → 8) Send question+context to chat model (e.g., gpt-3.5-turbo) → 9) Return answer to VA .

5. Security, Privacy, and Governance Considerations

This architecture transmits document content (extracted text and/or embeddings) to third-party services for conversion, embedding, storage, and generation . A production-grade implementation should define:

- data classification rules for allowed document types,
- retention and deletion policies for vectors and payload text,
- access controls around VA topics and vector collections,
- audit logging and incident response procedures,
- vendor risk approvals and contractual requirements.

6. Conclusion and Future Work

Integrating OpenAI embeddings and semantic search into ServiceNow VA enables a document-grounded conversational experience that reduces search friction and improves knowledge access at scale . The referenced implementation provides a practical blueprint—PDF upload, extraction, chunking, embeddings, Qdrant storage, semantic search, and chat completion—implemented through ServiceNow REST Messages, Script Includes, and a VA loop . Future extensions highlighted in the source include more advanced use cases such as fraud detection in invoices and richer document understanding beyond text-only extraction . From a research perspective, the next step is rigorous evaluation using a confusion-matrix-driven measurement approach (as emphasized in the demo paper) , augmented with modern RAG grounding and retrieval metrics.

This project successfully demonstrates the integration of Generative AI with ServiceNow to create an intelligent document interaction system. By combining OpenAI embeddings, semantic search, and virtual agent capabilities, the solution overcomes the limitations of traditional search mechanisms.

The system not only improves accessibility to information but also sets a strong foundation for future AI-driven innovations within enterprise platforms. As AI continues to evolve, such integrations will play a crucial role in transforming how organizations interact with data and knowledge.

In a world where time is of the essence, and the volume of data is overwhelming, the OpenAI Embedding and Semantic Search integration offers a game-changing solution. It empower users to extract insights and information swiftly, ultimately leading to improved productivity, better decision-making, and enhanced customer satisfaction. As we embark on this exciting journey of

revolutionizing the ServiceNow experience, the possibilities are boundless, and the future looks brighter than ever. This small implementation is just the starting of a lot more innovative ideas. Think if you can implement it in more advance way like detecting fraud in invoices, analyzing graphs data from reports and responding to work accordingly.

References

1. P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 9459–9474.
2. T. B. Brown *et al.*, "Language Models are Few-Shot Learners," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1877–1901.
3. J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, 2019, pp. 4171–4186.
4. ServiceNow Inc., "ServiceNow Virtual Agent Documentation," 2024. [Online]. Available: https://docs.servicenow.com
5. Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," 2024. [Online]. Available: https://deepmind.google/technologies/gemini
6. A. Johnson, J. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, Sept. 2021.
7. Qdrant Team, "Qdrant: Vector Database for the Next Generation of AI Applications," 2024. [Online]. Available: https://qdrant.tech/documentation
8. R. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.
9. T. Mikolov *et al.*, "Efficient estimation of word representations in vector space," in *Proc. International Conference on Learning Representations (ICLR)*, 2013.
10. M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation," in *Proc. ACL*, 2020, pp. 7871–7880.
11. Usha Kosarkar, Gopal Sakarkar, Shilpa Gedam , "An Analytical Perspective on Various Deep Learning Techniques for Deepfake Detection", 1st International Conference on Artificial Intelligence and Big Data Analytics (ICAIBDA), 10th & 11th June 2022, 2456-3463, Volume 7, PP. 25-30, https://doi.org/10.46335/IJIES.2022.7.8.5