MCKNIGHT
CONSULTING GROUP

Building Effective RAG Applications

Presented by: William McKnight

"#1 Global Influencer in Big Data" Thinkers360

President, McKnight Consulting Group

3 X Inc 5000

/in/wmcknight

www.mcknightcg.com
(214) 514-1444

McKnight Consulting Group Partial Technology Implementation Expertise

Big/Analytic/Vector/Mixed Data Management

databricks
Pinecone
Google BigQuery
snowflake
Azure Synapse
MySQL HeatWave
teradata.
amazon REDSHIFT
CLOUDERA
SAP Datasphere
Exasol
VERTICA
Starburst
ACTIAN activate your data
Azure HDInsight
SingleStore
Apache Impala
amazon EMR
Yellowbrick
Cloud Dataproc
ORACLE AUTONOMOUS DATABASE
ORACLE

Data Movement and APIs

Airbyte
Informatica
Fivetran
Qlik Replicate®
SYNADIA
CONFLUENT
talend
apigee
MATILLION
Stitch
Kong
Astera

Data Management

MONTE CARLO
Anomalo
SNOWPLOW
SAP Datasphere
alteryx DESIGNER CLOUD
splunk>
NGINX Part of F5
Collibra
IMMUTA
elasticsearch
Azure MLOps

Operational/Transactional Data Management

DIAMANTI
Cockroach
Couchbase
DataStax
SQLite
MongoDB.
PostgreSQL
yugabyteDB
Google Cloud Spanner
neo4j
ORACLE
redis
Azure Cosmos DB
Microsoft SQL Server
Amazon Aurora

MCKNIGHT CONSULTING GROUP

Analytics Architecture with William McKnight 2026

1. ~~2026 Trends in Analytic Architectures~~
2. ~~What Does Information Management Maturity Look Like in 2026~~
3. ~~The Data Product Revolution: Unlocking Business Value~~
4. Building Effective RAG Applications
5. Data Professionals in the AI Age: What's Next?
6. The Master Data Management Dilemma: To Buy or Build, That is the Question: Benchmark Completed
7. Promising AI Use Cases for the Enterprise in 2026
8. Data Mastery: William Answers Your Questions
9. Data Pipeline Engineering Strategies
10. How to Work with Open Table Formats
11. The ROI of Agentic AI: Strategies for Success
12. Data Architecture 2027: What enterprises are building today and why

RAG (Retrieval-Augmented Generation)

This is a technique that combines language models with vector databases. The process:

1. Prompt: The LM receives a question.
2. Vectorization: The system converts that question into a mathematical embedding.
3. Retrieval: The system searches the vector database for the most relevant data chunks.
4. Refinement: The retrieved data is re-ranked and filtered for maximum accuracy.
5. Generation: The LM uses that precise context to generate a grounded, accurate response.

AI Concepts

Retrieval-Augmented Generation (RAG)

RAG combines pre-trained language models with external knowledge retrieval. It dynamically fetches relevant information from databases or documents to enhance response accuracy and provide up-to-date information.

Data Access
External, Real-time

Memory
Unlimited (via retrieval)

Update Frequency
Real-time possible

Computational Cost
Higher per query

Small Language Models (SLMs)

SLMs are compact language models with fewer parameters, designed for efficiency and specific tasks. They offer faster inference, lower computational requirements, and can be deployed on edge devices while maintaining reasonable performance.

Model Size
1B-7B parameters

Deployment
Edge, Mobile, Local

Speed
Fast inference

Specialization
Domain-specific

Pre-training

Pre-training is the initial phase where models learn general language understanding from massive datasets. This foundational training creates versatile models that understand syntax, semantics, and world knowledge.

Data Scale
Billions of tokens

Learning Type
Unsupervised

Duration
Weeks to months

Purpose
General knowledge

Fine-tuning

Fine-tuning adapts pre-trained models for specific tasks or domains. It involves continued training on smaller, task-specific datasets to optimize performance for particular use cases while preserving general knowledge.

Data Scale
Thousands to millions

Learning Type
Supervised

Duration
Hours to days

Purpose
Task specialization

Use RAG When...; Use Small Language Models When...

Use RAG When...

- Business Intelligence: Building dashboards that need real-time data analysis and reporting
- Knowledge Base Q&A: Customer support systems accessing constantly updated documentation
- Best For: Dynamic information needs, factual accuracy, real-time updates

Use Small Language Models When...

- Mobile Apps: On-device text processing, autocomplete, and writing assistance
- Edge Computing: IoT devices requiring local language processing capabilities
- Privacy-Critical: Healthcare or financial apps needing on-premise processing
- Real-time Systems: Gaming, chatbots with strict latency requirements
- Cost-Sensitive: Startups or applications with tight computational budgets
- Offline Applications: Systems that must work without internet connectivity
- Best For: Resource constraints, speed requirements, privacy needs

Use Pre-training When...; Use Fine-tuning When...

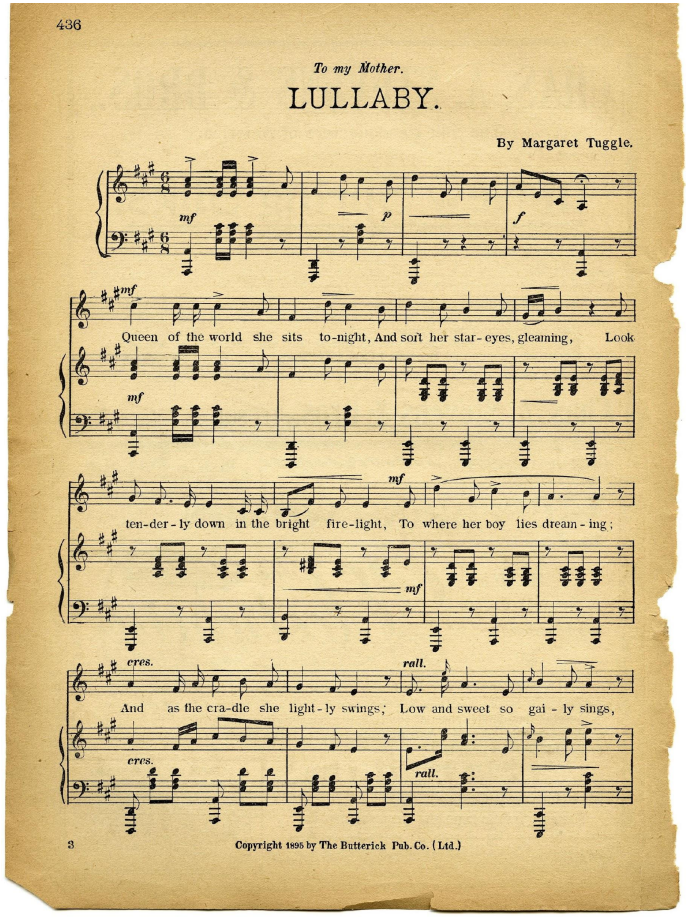
Use Pre-training When...

- New Languages: Creating models for under-resourced languages or dialects
- Enterprise Foundation: Companies building proprietary models from scratch
- Research Projects: Academic work exploring new architectures or techniques
- Massive Datasets: Applications with unique, large-scale text corpus
- Specialized Tasks: Applications requiring fundamentally different language understanding
- Best For: New languages, massive resources, research purposes
- You have a large budget

Use Fine-tuning When...

- Content Generation: Brand-specific writing styles, marketing copy, technical documentation
- Translation: Domain-specific translation (medical, legal, technical)
- Conversational AI: Customer service bots with specific personality traits
- Information Extraction: Named entity recognition, relationship extraction
- Summarization: News articles, research papers
- Best For: Task specialization, performance optimization, adapting new tasks

RAG and Language Models

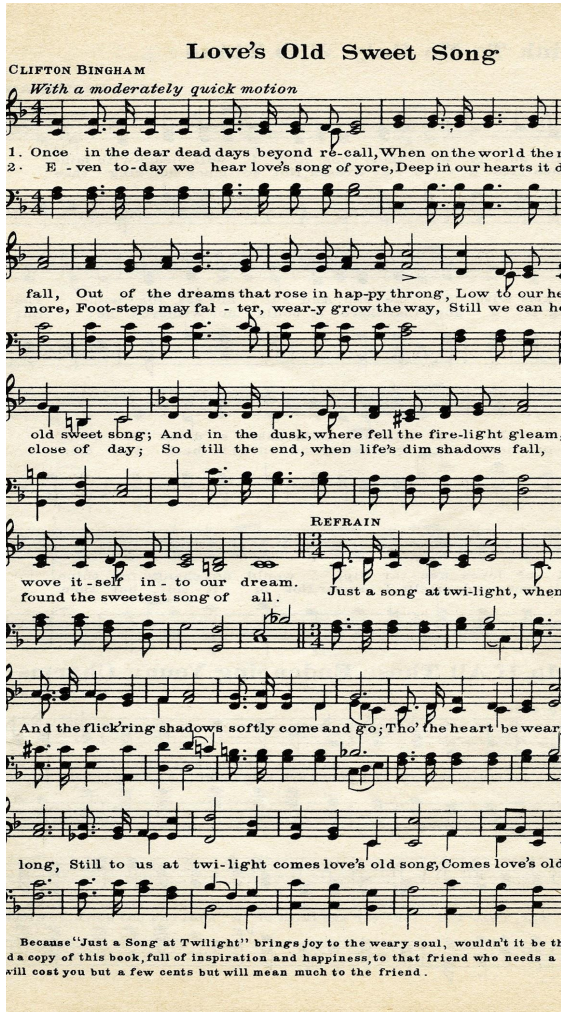


Retrieval-Augmented Generation combines the strengths of LMs and external knowledge sources

Enhances accuracy, relevance, and diversity of generated content

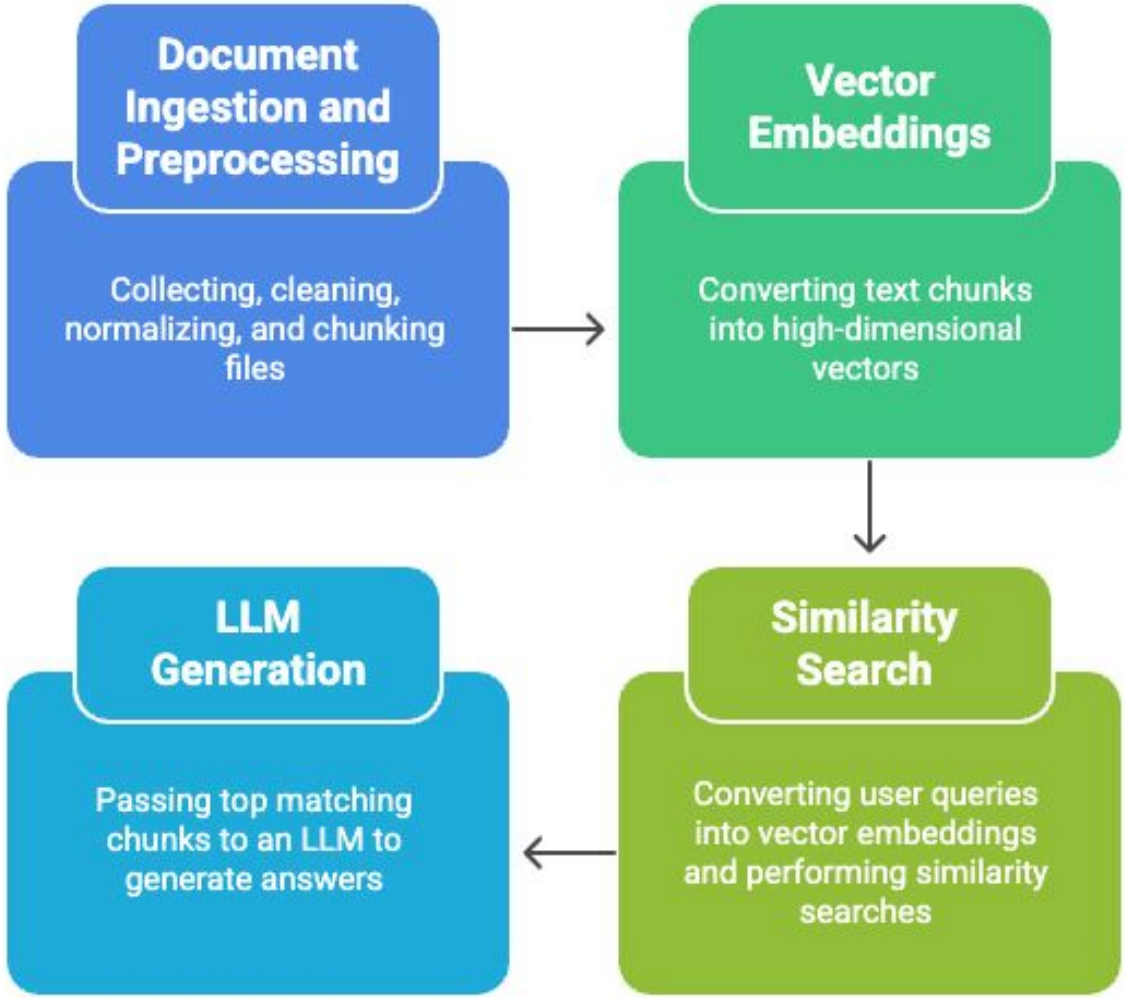
- LMs and RAG: A Powerful Duo:
- LMs provide context and language understanding
- RAG leverages external knowledge to augment LM capabilities
- Key Benefits:
- Improved accuracy and relevance
- Increased diversity and novelty
- Enhanced ability to handle nuanced and complex topics
- Applications:
- Content generation (text, chatbots, etc.)
- Question answering and information retrieval
- Summarization and knowledge graph construction

RAG Use Cases



- Customer Service
- Advisory
- Programming Assistants
- Summarization
- Language Translation
- Essay Writing
- Image Captioning

RAG Application Process

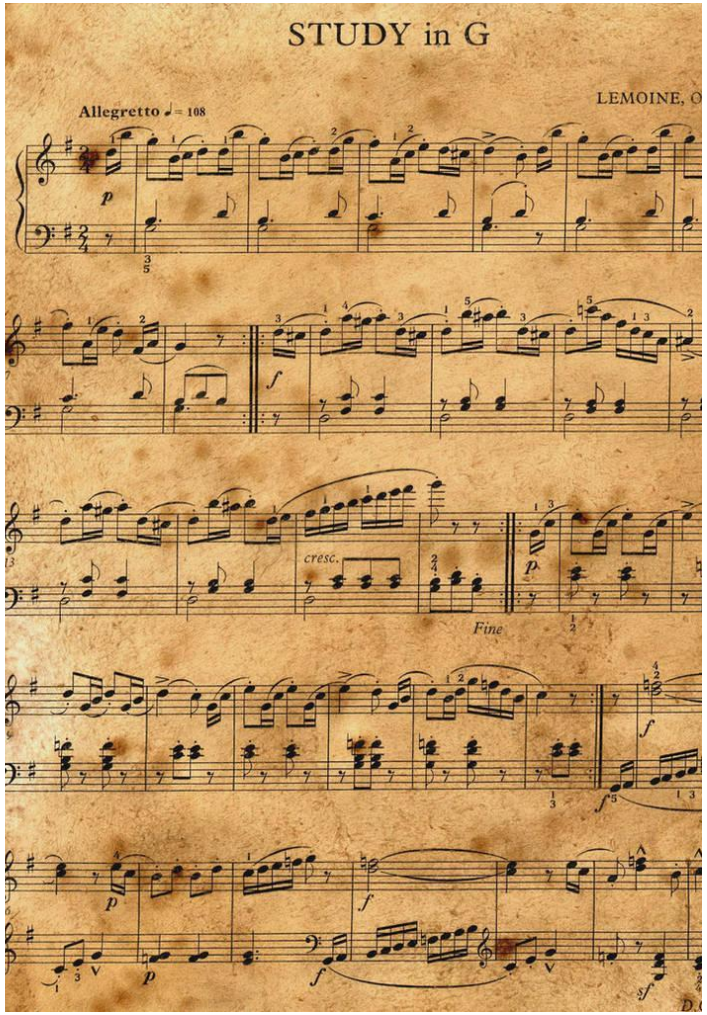


Effort and Complexity

DIY	Development Story Points (one-time)
Source Document Ingest	13
Document Preprocessing and Chunking	13
Embedding Microservice	21
LLM Generation Service	63
Retriever Service	39
Logs	32
Metrics, Dashboards, and Alarms	13
Access Control, Secrets & Encryption	5
DIY Total	**199**

The Enterprise RAG Blueprint

- Data Lake (The Raw Brain): Your source of truth (S3, Snowflake, local PDFs). Effective RAG starts here with clean data.
- Message Queue (The Synchronizer): Decouples the data ingestion from the embedding process. When data changes in the Lake, the Queue manages asynchronous updates to the Vector DB.
- Vector DB (The Indexed Memory): Stores data as semantic embeddings, optimized for similarity search (cosine, dot product).
- Distributed HA Microservices: The engine of scalability. We don’t run embeddings on a single server; we use microservices for chunking, embedding, and re-ranking to ensure high availability (HA).



Ingestion Pipeline and Microservices

- Optimized Chunking Strategies: The "one-size-fits-all" chunk is a myth. For technical manuals, we use smaller chunks; for policy docs, we need larger chunks.
- Metadata Tagging: This is critical. We tag chunks with date, source, department, and version. This allows for strict filtering during the retrieval step (e.g., "only show 2026 data").
- Distributed Workflow: Using tools like Celery and Airflow, our microservices manage the concurrent processing of millions of chunks. We embed asynchronously to avoid bottlenecks and use a model registry to manage embedding model versions.

Example RAG Stack: AWS

Database + AI	Amazon RDS for PostgreSQL
	Amazon KMS
	Amazon EC2 (Embedding)
Distributed HA	Amazon EC2 (Embedding)
Message Queue	Amazon MSK
Data Lake	Amazon MSK
Observability & Monitoring	Amazon CloudWatch
Security & Compliance	Amazon IAM

Example RAG Stack: Azure

Database + AI	Azure Database for PostgreSQL
	Azure Key Vault
	Azure ML (for Embedding)
Distributed HA	Azure Virtual Machines (With Avail Sets)
Message Queue	Azure Event Hubs
Data Lake	Azure Data Lake Storage Gen2
Observability & Monitoring	Azure Monitor
Security & Compliance	Microsoft Entra ID

Example RAG Stack: GCP

Database + AI	Cloud SQL for PostgreSQL
	Cloud KMS
	Vertex AI
Distributed HA	Compute Engine
Message Queue	Pub/Sub
Data Lake	BigQuery
Observability & Monitoring	Cloud Logging & Monitoring
Security & Compliance	Cloud IAM

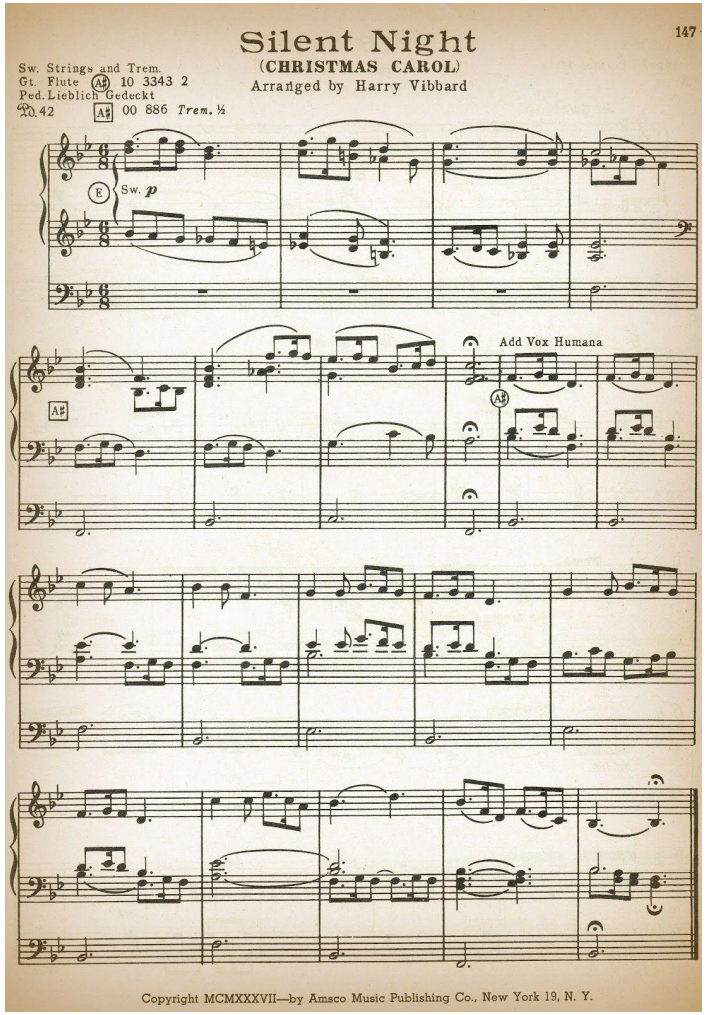
The Strategic RAG Framework

- I am using RAG because it is
  a) The less expensive way to meet the standards for the application
  b) Workable for the application need but leverageable to other applications, making them less expensive to do long-term
- This RAG application is being built
  a) Solely for this application with no consideration given to future applications
  b) With consideration of future expansion of this application and others
- This RAG application will deliver to the company
  a) An increase in sales
  b) A reduction in expenses
  c) A protection of sales
  d) A protection of expenses
- The process in this RAG application that I own is
  a) Data ingestion and pre-processing
  b) Embedding and vectorization
  c) Retrieval
  d) Refinement
  e) Generation
  f) Guardrails

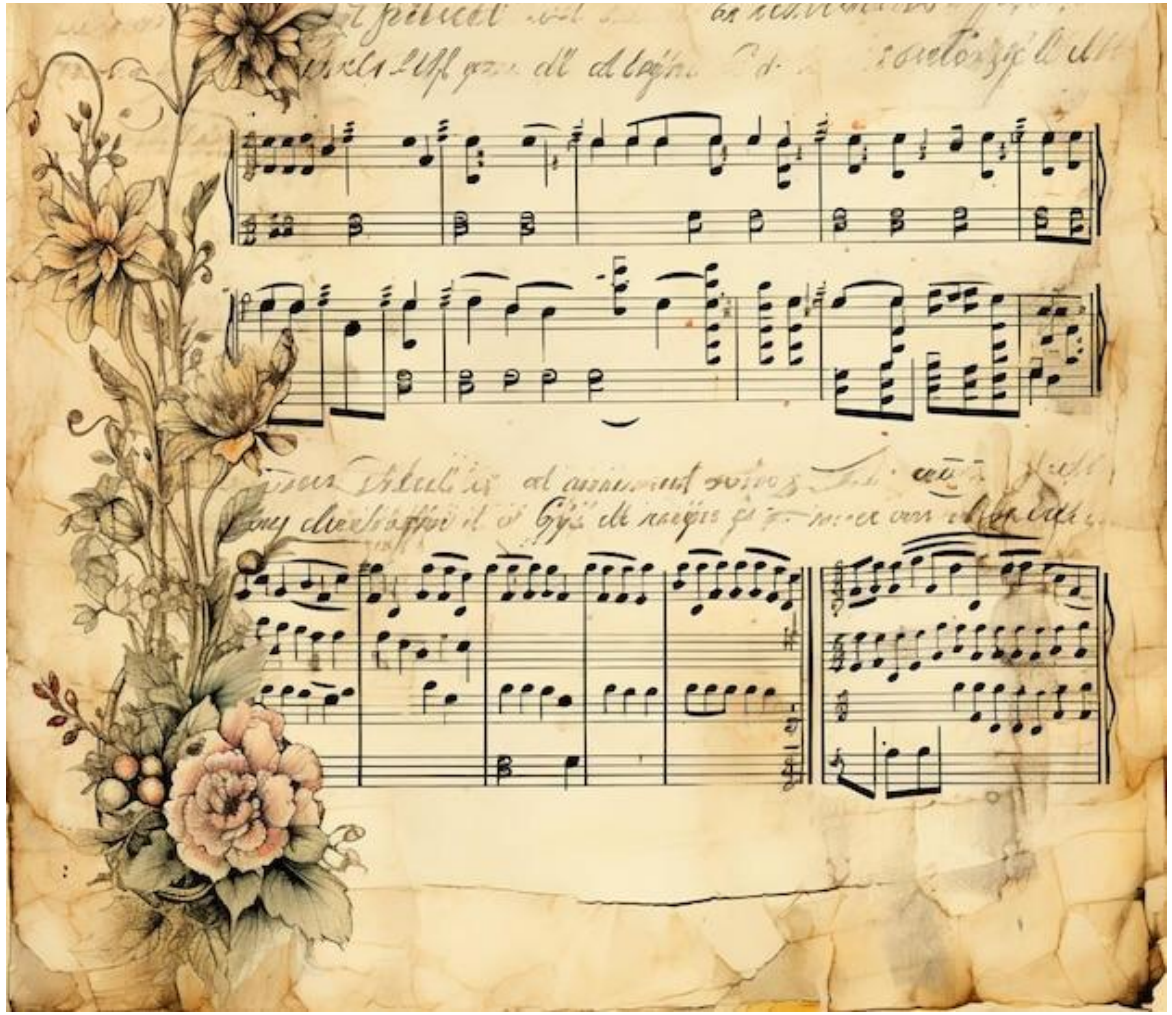
Vector Databases and RAG

Enter Vector Databases

- Conventional databases lack the structural capability to accommodate imprecise comparative inquiries such as "which items are comparable to this one?"
- The exploration of machine representations of datasets such as text, voice, image, and molecular structures is underway as ML and LLMs are applied to novel problems.
- Vector emerged to address new issues, much like the NoSQL generation of databases did.
- User inquiries evolved in tandem with the influx of machine representations of data.
- To address them, vector databases, a novel technology, were required.

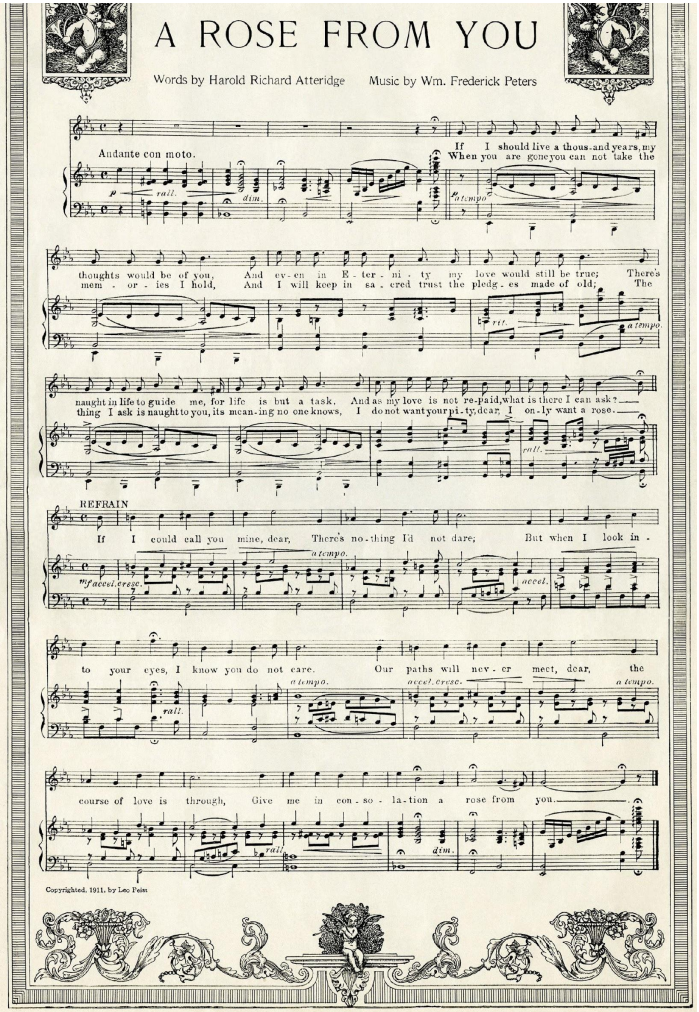


Imprecise/Similarity Search



- Ever searched for something with unclear detail?
- Struggling to search because you lack the exact keywords?
- Remembering actors but forgetting the movie title?
- Frustrated with outdated information in search results?
- If any of these sound familiar, then vector search can be your solution

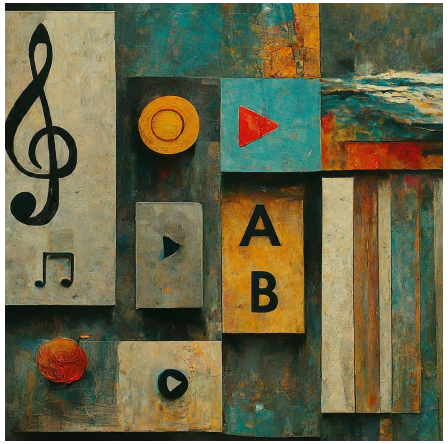
Vectors



- Numeric representations
- "The only thing we have to fear is fear itself" = [-2.345, 7.812, -1.009, 4.567, 0.123, 9.998,...]
- "To be or not to be, that is the question" = [-3.141, 2.718, -8.000, 1.618, 4.200, 7.539, ...]
- "I think, therefore I am" = [-5.827, 0.123, 9.000, -1.414, 3.142, 6.789, ...]

Vector Embeddings come from an Embedding Model

- Example Embedding Models are provided by OpenAI, Cohere, Google Vertex, HuggingFace



OpenAI

Hugging Face

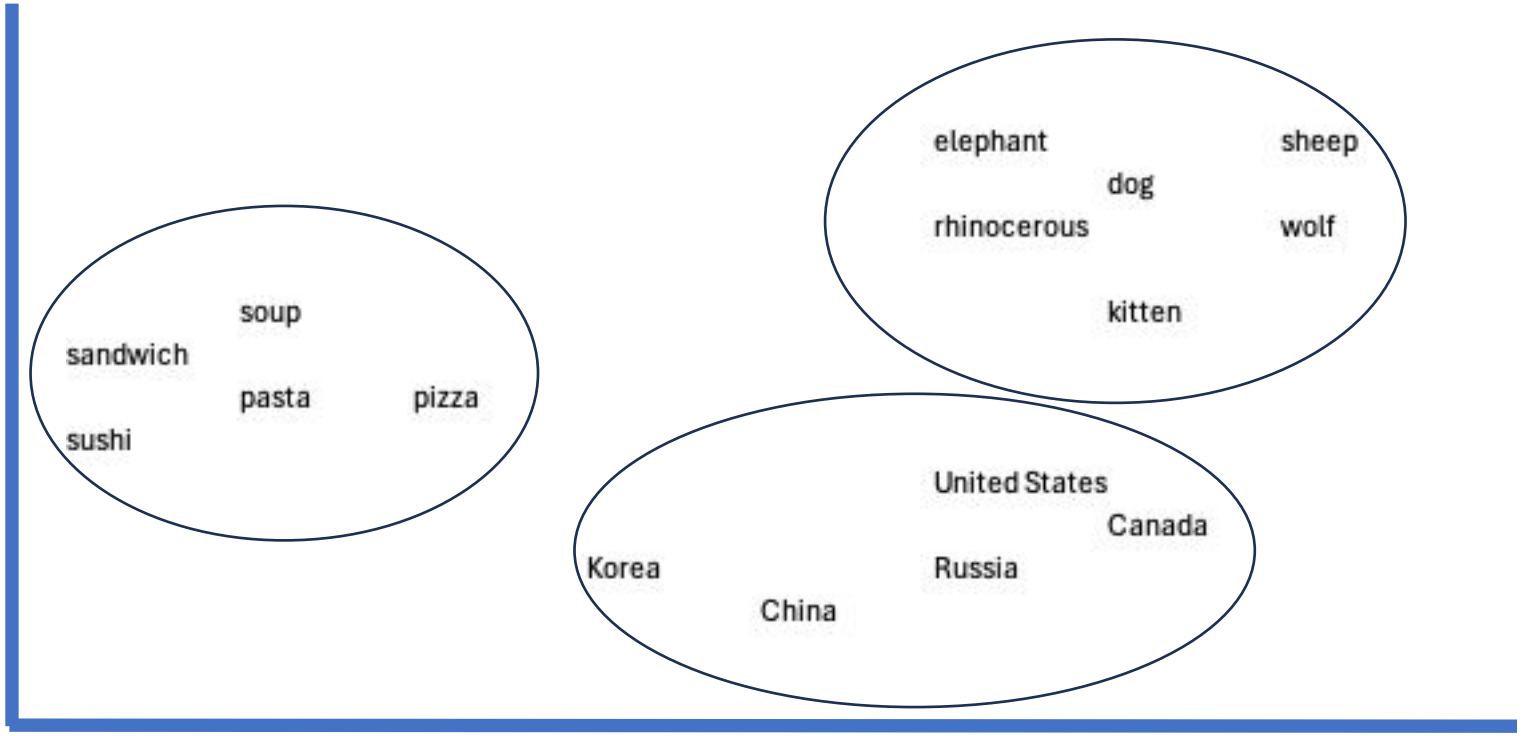
cohere

[-2.345, 7.812, -1.009, 4.567, 0.123, 9.998,…]
[-3.141, 2.718, -8.000, 1.618, 4.200, 7.539, …]
[-5.827, 0.123, 9.000, -1.414, 3.142, 6.789, …]

Embedding Models

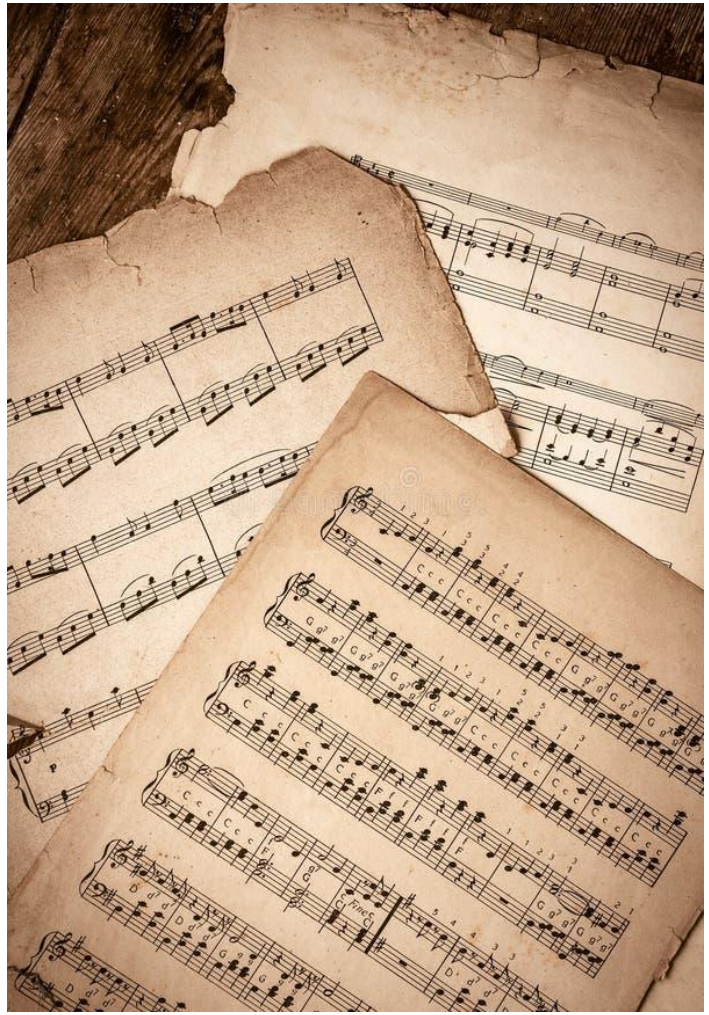
- Embedding Models are techniques for representing words as numerical vectors.
- These vectors capture the semantic meaning and relationships between words.
- Embedding Models are trained on massive datasets of text.
- The model analyzes the context in which words appear, learning their meaning based on surrounding words.
- Words with similar meanings have similar vector representations in the embedding space.
    - Imagine "king" and "queen" having vectors close together, while "king" and "car" would be farther apart.
- Embeddings fuel various applications:
    - Recommendation systems suggesting similar products based on user searches.
    - Machine translation finding the closest meaning in another language.
    - Chatbots understanding the intent behind user queries.

Data Clustering in Space

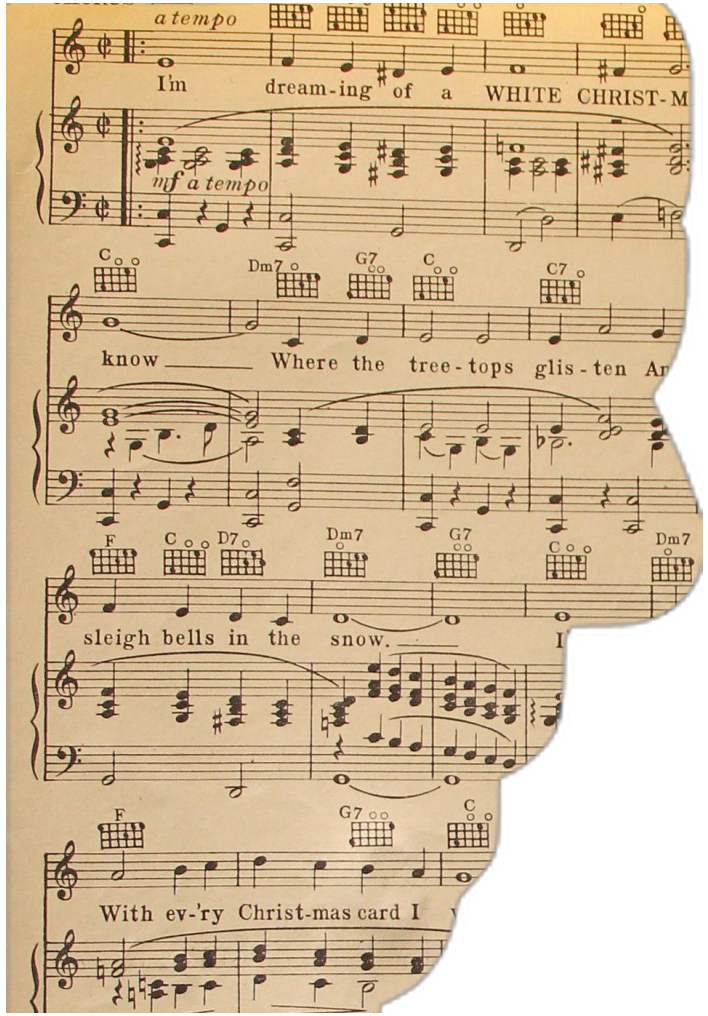


DBMS are adding Vector Support

- Vector data has a wide range of use cases, ultimately limited only by creativity and imagination.
- Isolated data sources (silos) can limit the accuracy and efficiency of analytics, including those involving vectors.
- Developers are overwhelmed by managing an abundance of individual tools and interfaces.
- Both developers and enterprises prioritize consolidation of their technology stacks for better manageability.
- The ideal scenario is a single interface offering broad capabilities across various data model problems, without sacrificing functionality.



Vector Search

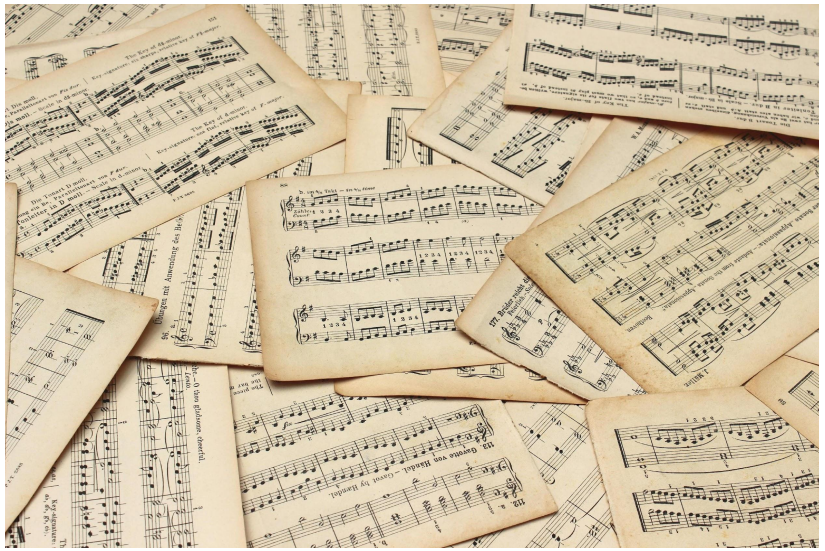


- By using the distance between high dimensional vectors, vector search enables us to search through data and find relevant results.
- When calculating the distance between your query and stored vectors at query time, it uses machine learning models to embed the data.

Vector Search

- Search based on meaning, not just keywords
- Leverages machine learning models called encoders for powerful results
- Embeddings
  - Text, audio, images transformed into a numeric string called "vectors"
  - High-dimensional arrays with semantic meaning
  - Makes data available to AI
- Benefits of Vector Search
  - Semantic Understanding
  - Scalable
  - Flexible – anything can be vectorized

- Example: [-0.0385810, -0.1348581, 0.0184810, -0.138542, -0.1984815, 0. 12498134, 0.0124897, -0.021858, -0.0002384, -0.024911, 0.199248284, ….]



Simple Vector_Distance Function

SELECT…

FROM house_for_sale

WHERE price <= (SELECT budget FROM customer …)

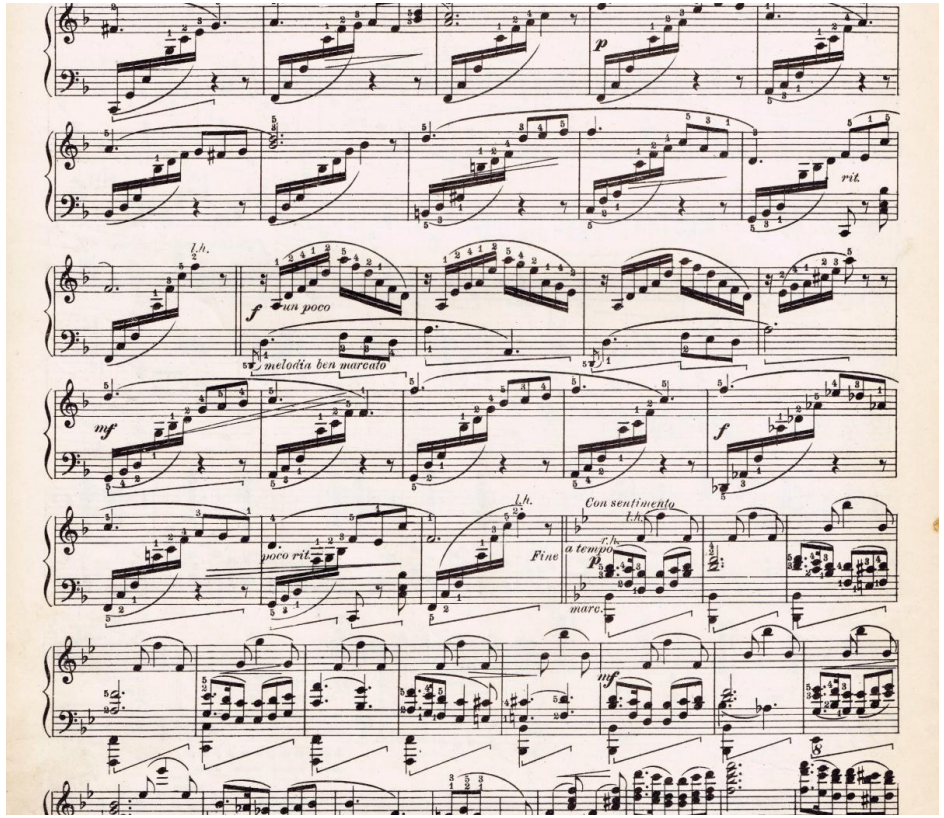
AND city in (SELECT search_city FROM customer …)

ORDER BY vector_distance(house_vectors, :input_vector);

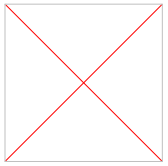
Hierarchical, Navigable Small World

- HNSW utilizes a graph-like structure with layers, enabling efficient traversal to find approximate nearest neighbors.
- Hierarchical Navigable Small World (HNSW) is an indexing strategy.
- HNSW enables fast retrieval of "mostly" nearest neighbors for K-Nearest Neighbors (KNN), improving efficiency for large datasets.
- HNSW utilizes a graph-like structure with layers, where similar data points are connected within layers.
- During a search, the algorithm traverses the layers of the graph, efficiently navigating towards potential nearest neighbors.
- HNSW offers a trade-off between finding the absolute closest neighbors (accuracy) and achieving high search speed.
- It is ideal for real-time applications with large datasets where KNN needs a performance boost.

K-Nearest Neighbor



- Closest thing to exact search for vectors; it finds the nearest neighbors.
- Typical approach to classification problems is k-nearest neighbors (KNN) Classification.
- KNN predicts the label (class) or value (regression) for a new data point by looking at its k nearest neighbors in the training data.
- Contrary to the notion of "exact search," KNN doesn't necessarily find perfect matches in the training data.
- It focuses on identifying the k data points most similar (closest) to the new point based on a chosen distance metric (e.g., Euclidean distance).



Similarity Functions

- Eucliden
  - Measures the straight-line distance between two points in a multidimensional space.
  - Commonly used for numerical data with real number values. Larger distance indicates less similarity.
- Cosine
  - Measures the directional similarity between two vectors. Useful for data where the magnitude (length) of the vectors may not be important.
  - Values range from -1 (completely opposite) to 1 (identical).
- Dot Product
  - Calculates the product of corresponding components of two vectors and then sums them up.
  - Closely related to cosine similarity, but it considers the magnitudes of the vectors as well.
  - Larger dot product indicates greater similarity.

Post-Retrieval: Reranking and Grounding

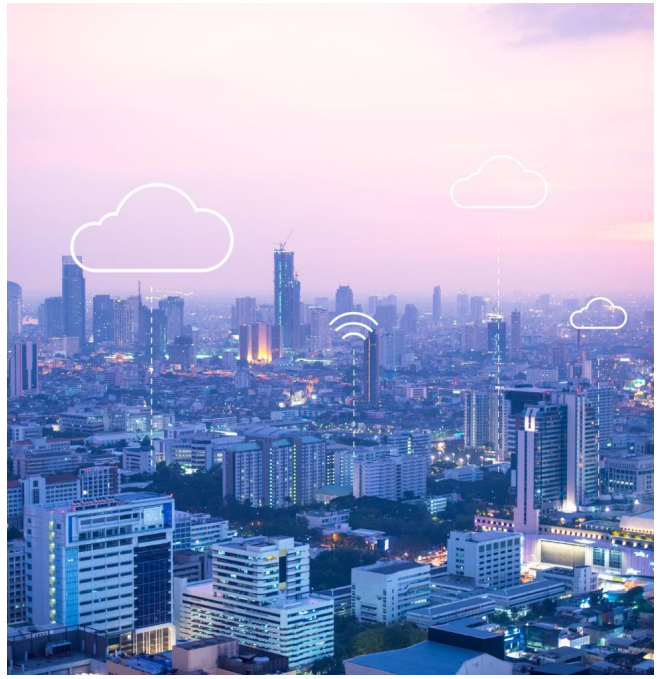
- Hybrid Search (Vector + Keyword): When we don’t trust vectors alone, we combine semantic relevance with exact keyword matching to catch specific part numbers or technical terms.
- Re-Ranking the Results: The most "similar" chunk might not be the most "relevant." We use a second-stage, specialized re-ranker model (like a Cross-Encoder) to re-order the top 10 retrieved chunks before they ever reach the LLM.
- Grounding the Answer: This is the most crucial step. We use a separate LLM call to perform "Fact Checking." We ask the second LLM: "Is the final answer fully supported by the retrieved chunks? If any part is not supported, flag it as a hallucination."

Agentic RAG: From Linear Pipeline to Intelligent Loop

- Evaluation Gap: Standard RAG cannot judge if retrieved data is actually "good" or just "similar." Agentic RAG evaluates relevance before generating.
- Context Gap: Standard RAG takes a query at face value. Agents can refine or rewrite ambiguous questions (e.g., clarifying "tax" as "personal" vs. "business") to improve search results.
- Source Gap: Agents act as intelligent routers, deciding whether to query a SQL database, a vector store, or a web API based on the specific needs of the query.

Summary

- RAG combines LLMs with vector databases for context-aware responses
- Choose RAG for dynamic data and fewer model hallucinations
- Use enterprise blueprints and microservices to manage RAG pipelines
- Embedding models convert data into high-dimensional semantic vectors
- They use k-NN and HNSW to optimize retrieval accuracy
- Compares RAG stacks across AWS, Azure, and Google Cloud
- RAG supports bots, coding assistants, and complex research tasks



MCKNIGHT
CONSULTING GROUP

Building Effective RAG Applications

Presented by: William McKnight

"#1 Global Influencer in Big Data" Thinkers360

President, McKnight Consulting Group

3 X Inc 5000

/in/wmcknight

www.mcknightcg.com
(214) 514-1444