Embedding Generative AI and Advanced Search into your Apps with MongoDB

Building AI-Powered Applications

December 2023

US 866-237-8815 • INTL +1-650-440-4474 • info@mongodb.com.

Table of Content

Introduction

Never before has the introduction of a new technology so rapidly captured the attention of enterprises, governments, and consumers alike. The arrival of ChatGPT in November 2022 showcased the potential of Generative AI powered by Large Language Models (LLMs) in addressing a vast array of new use cases. These use cases were previously unimaginable with conventional computing and analytical AI (now sometimes described as “traditional” or “classic” AI).

All it seems to take is a few well crafted prompts to automate a whole range of things. Generate professional quality text, images, audio, video and programming code. Better support customers. All the way through to modeling climate change, discovering new drugs or designing new materials, predicting the movements of financial markets, and many, many more.

Overnight one question appeared at the top of every boardroom agenda – *“how can we use Generative AI to disrupt our markets while not being disrupted ourselves”?*

However technology leaders have quickly recognized that alongside the potential benefits of GenAI, there are also risks from the technology’s immaturity. They can’t just discard years of operational best practices and institutional knowledge. Instead they need to make sure both their existing systems as well as new applications in development are capable of harnessing Generative AI in ways that are safe, reliable, and accurate.

In this paper, we will discuss how MongoDB can put you on the path to achieving those goals while using your own data to power compelling new GenAI-powered applications and experiences.

Context is everything

When everyone has access to GenAI models, your “superpower” differentiation comes from giving those models access to one of your most important enterprise assets – your data. Some of this data will be proprietary to the organization and some will be public – but fresher – than that used to train the original base foundation models. Together this data provides responses that better reflect today’s “ground truth”.

Providing models with your own data is accomplished by a new architectural pattern called Retrieval-Augmented Generation or RAG. Using RAG presents your developers with a potent combination. They can take the incredible knowledge and reasoning

capabilities of pre-trained, general purpose GenAI models and feed them with accurate and up-to-date company-specific data.

The results are GenAI outputs that are accurate, up-to-date, relevant, and harness all of your data, no matter its structure. Your GenAI-powered apps better serve your customers, boost employee productivity, and out innovate competitors. Your developers can unlock all of these outcomes without having to turn to specialized data science teams to train or fine tune models — a complex, time-consuming, and expensive process.

Using your own data sources are one important piece to making generative AI work for the business. But on its own, it's not enough. As we discuss later in the paper, developers also need to consider how to deploy their application around an informed large language model with the right security controls in place, and at the scale and performance users expect.

The rise of vectors and similarity search

To feed AI models with our own data, we need to first turn it into vector embeddings. These vectors provide multi-dimensional numerical encodings of our data that captures its patterns, relationships and structures. Vector embeddings give our data semantic meaning; calculating the distance between vectors makes it easy for our applications to understand the relationships and similarities between different data objects. This opens our data up to a whole new range of applications that we discuss below.

Data in any digital format and of any structure – i.e., text, video, audio, images, code, tables – can be transformed into a vector by processing it with a suitable vector embedding model. For example, OpenAI's `text-embedding-ada-002` is one of the most popular models for vectorizing textual content. The beauty of vector embeddings is that data that is unstructured and therefore completely opaque to a computer can now have its meaning and structure inferred and represented via these embeddings. This means we can start to search and compute unstructured data in the same way we've always been able to with structured business data. Considering that 80%+ of the data we create everyday is unstructured, then we start to appreciate how transformational vector search combined with GenAI really is.

As shown in Figure 1 below, once our data has been transformed into vector embeddings, it is persisted and indexed in a vector store such as MongoDB Atlas Vector Search. To retrieve similar vectors, the store is queried with an Approximate Nearest Neighbor (ANN) algorithm to perform a K Nearest Neighbor (KNN) search using an algorithm such as 'Hierarchical Navigable Small Worlds' (HNSW).

Querying these vectors allows us to do things with data that we could only previously accomplish with expensive data science skills and infrastructure. Firstly, we can extend information search and discovery beyond keyword matching to context-aware semantic search which is capable of inferring meaning and intent from a user's search term. Secondly, we can retrieve our own data – encoded as vectors – to provide the GenAI model with the context necessary to generate more reliable and accurate outputs. These outputs can include:

- Natural Language Processing (NLP) for tasks such as chatbots and Question-Answering to text summarization and sentiment analysis.
- Computer vision and audio processing for image classification and object detection through to speech recognition and translation.
- Content generation, including creating text-based documentation and SEO-optimized web pages, computer code, or converting text to an image or video.

Vector search and the LLM workflow

Figure 1 brings together the workflow enabling "Retrieval Augmented Generation" for a LLM.

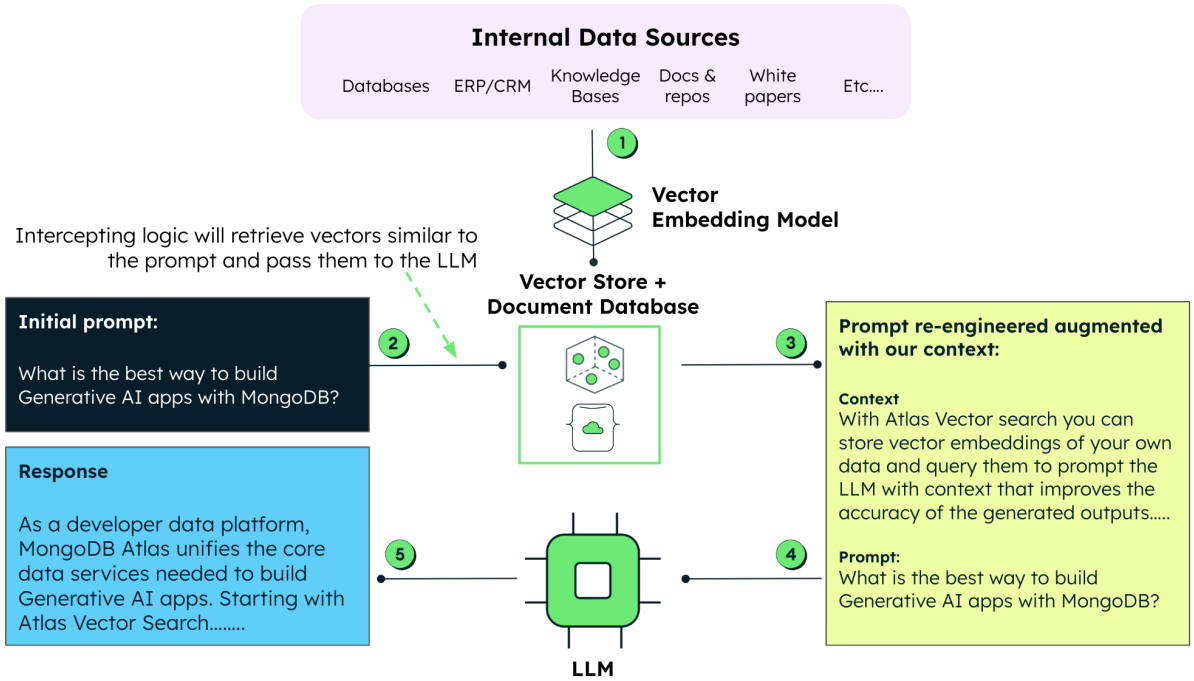

Figure 1: *Dynamically combining your custom data with the LLM to generate reliable and relevant outputs*

Ahead of time, our data is transformed by a vector embedding model and stored in a vector store. Ideally the vectors' metadata and raw "chunked" data are stored alongside the vectors themselves in a flexible document database which also stores our regular application data. This allows our application to query data in multiple ways,

improve relevance (e.g. score more recent data higher) and provides long term memory for the LLM. Prompts to the LLM are intercepted by logic that retrieves similar vectors from the vector store. These are then used to reengineer the initial prompt. The new prompt is sent to the LLM which is able to use the provided context to generate higher quality and accurate responses using fresher data.

Later in this paper you will find examples that demonstrate the workflow above and show how the resulting capabilities can be applied to different classes of applications.

The promise and reality of a vibrant AI ecosystem

Vector stores are one part of a rapidly evolving ecosystem of AI-enabling technologies extending from creation of embeddings, to prompt engineering, LLMs, model fine-tuning, orchestration, logging, infrastructure automation, and more.

Within this ecosystem there are no end of interesting and promising projects and vendors to work with. Some show the “art of the possible” through demos and prototypes. But the fear enterprise decision makers and developers face is how easily these prototypes can be adapted for their specific business needs. And whether some of the newer technologies can really sustain production load with reliability, scalability, and security, day in and day out, under any operational environment. A further consideration is how to integrate the organization's own databases to feed real, ground truth business data into the model.

The AI ecosystem does not exist in isolation. All of these technologies need to be embedded into real world applications to be truly useful to the business. For example, vector stores are essential to enabling context-aware generative AI and semantic search. But these are just one part of a broader application that also has to manage regular, non-vectorized business data.

This data can be anything – customer records, orders and inventory, trades and transactions, quotations, geospatial coordinates, product details and pricing, time series measurements and sensor readings, clickstreams and social feeds, text descriptions, and more.

All of this data needs to be queried to power application functionality. Not just to retrieve approximate nearest neighbors between vectors, but also to accomplish regular operations such as retrieving specific records, handling a firehose of updates to the data, and running sophisticated aggregations and transformations supporting analytics processing. These queries power application features outside of any generative AI use cases. But they become even more important when we can use them

alongside in-context prompts to our models, improving accuracy and relevance of the GenAI model's outputs.

Beyond working with our application data and vector embeddings, we need to do the non-functional things – meeting uptime, performance and scalability SLAs, integrating new features, securing and backing data up, and auditing it. Some of this stuff can sound boring. That is until it fails. Then suddenly it isn't boring any more.....

Bringing together the technologies to power new AI-driven experiences and fusing them into your applications risks creating a sprawl of point products and complexity that places enormous overhead on your teams. All of these challenges add up to fragmented and inefficient developer experiences, a multitude of operational and security models to deal with, a ton of data wrangling and integration work, and lots of data duplication. All of this slows the speed of bringing your new AI-driven experiences to market, while increasing your costs and risk.

Using a developer data platform built on MongoDB Atlas gives you a better way.

A developer data platform: the smart way to build intelligent applications

MongoDB's developer data platform, built on MongoDB Atlas, unifies operational, analytical, and generative AI data services to streamline building intelligent applications. However you are harnessing AI – from training and serving your own machine learning models to embedding the latest generative AI in your apps – Atlas is a critical part of your stack. From prototype to production, with Atlas you can ensure your apps are grounded in truth with the most up-to-date operational data, while meeting the scale, security, and performance users expect.

At the core of MongoDB Atlas is its flexible document data model and developer-native query API. Together, they enable your developers to dramatically accelerate the speed of innovation, outpacing competitors and capitalizing on new market opportunities presented by generative AI.

Documents are the best way for developers to work with data because they map to objects in code making them intuitive and easy to reason about. Documents can model data of any structure – from the vast diversity of regular application data we discussed earlier to vector embeddings composed of several thousands of dimensions. Any of these structures can be modified at any time to support the addition of new data types and application features. Documents give you the flexibility to rationalize and harness that data in ways that traditional tabular data models of relational databases simply can't.

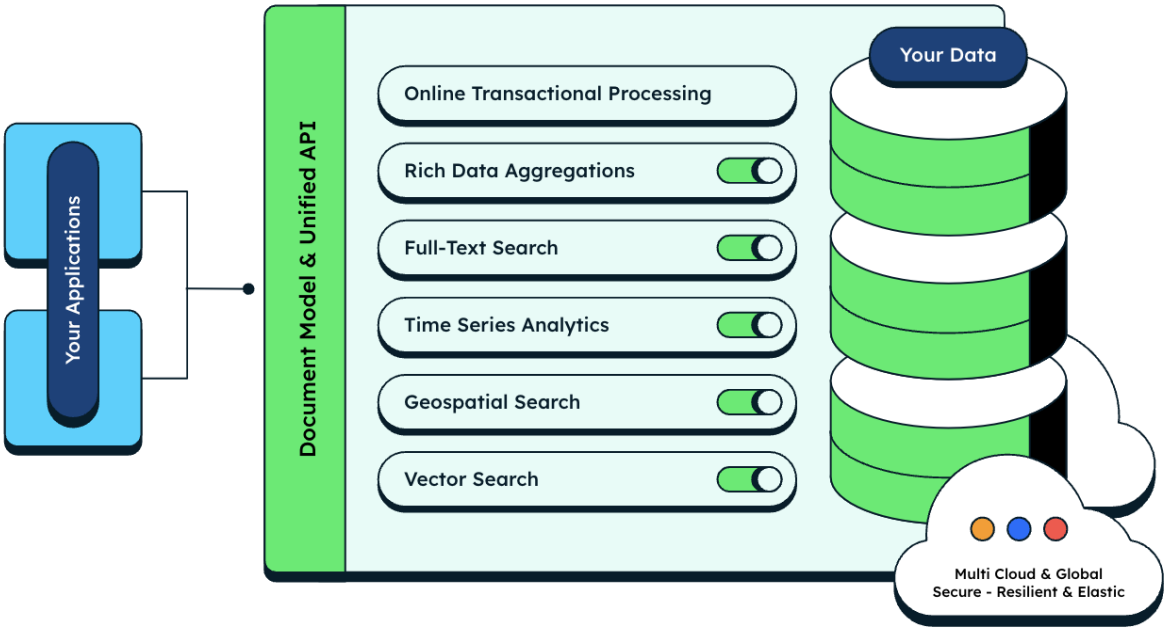

Figure 2: *MongoDB Atlas integrates the data services needed to bring AI into your applications*

In conjunction with the document model, the MongoDB Query API gives developers a unified and consistent way of working with data across any data service. From simple CRUD operations to keyword and vector similarity search through to sophisticated aggregation pipelines for analytics and stream processing, the MongoDB Query API provides developers the flexibility to query and compute data anyway the application needs. In the context of GenAI, this provides extremely flexible and powerful ways of defining additional filters on vector-based queries, for example:

- Combining with metadata for filtering: "Find me content matching the user's query but only content published in years X, Y, and Z".
- Combining with aggregations: "Find me all images that are similar to the query image and group them by photographer id".
- Combining with geo-spatial search: "Find me real-estate listings for houses that are similar to the house in this photograph that is within N miles of my location".

No other database is able to offer such a breadth of query functionality in a single, unified, query experience. This allows developers to build end-user functionality with greater ease and less complexity. Developers no longer have to manually stitch together query results from multiple databases, which is a process that is complex, error prone, costly, and slow. At the same time, it also keeps your technology footprint compact and agile.

“MongoDB was already storing metadata about artifacts in our system. With the introduction of Atlas Vector Search, we now have a comprehensive vector-metadata database that’s been battle-tested over a decade and that solves our dense retrieval needs. No need to deploy a new database we'd have to manage and learn. Our vectors and artifact metadata can be stored right next to each other.”

Pierce Lamb, Senior Software Engineer on the Data and Machine Learning team at VISO TRUST.

Show, don’t tell. Generative AI-enhanced apps on a developer data platform

We will focus on three popular use cases to showcase how developers are using MongoDB Atlas to build AI-enriched apps:

- Chatbot and Question-Answering (Q-A) for customer self-service.
- Advanced ecommerce search and user recommendations.
- Rich media (multimodal) analysis and generation.

Each of these examples rely on Generative AI and advanced semantic search to create amazing user experiences and unlock capabilities that were previously out of the reach of most organizations. To be truly transformative however, these AI enhancements need to be delivered as part of a larger application that itself is powering critical business functionality.

We will step through each use case in turn, showing an architectural design pattern that supports it, along with the relevant capabilities provided by MongoDB Atlas.

Chatbot and Q-A for customer self service

MongoDB sits at the heart of many customer support applications. This is because MongoDB’s flexible data model makes it easy to build a single, 360-degree view of the customer. It does this by dynamically ingesting diverse and rapidly changing customer data from the myriad of siloed backend source systems typical in most organizations. The single, consolidated real-time customer view powered by MongoDB is therefore the ideal platform against which we can train and deliver chatbot and Q-A assistance features for customer self-service.

In our example shown in Figure 2, the customer database stored in MongoDB is exported as a JSON file to an embedding model that chunks the data (using tools like LangChain or LlamaIndex) and creates vector embeddings from it. Other internal data

sources such as knowledge bases and documentation can also be vectorized for use in the app. The data is then imported back into the MongoDB database.

We need to make sure our vectors are constantly updated with the freshest customer data, so we use Atlas Triggers to watch for any data changes in our single view. As soon as new customer records are inserted or existing records updated in the database, Atlas Triggers calls the embedding model's API to generate the corresponding vectors and load them back into Atlas.

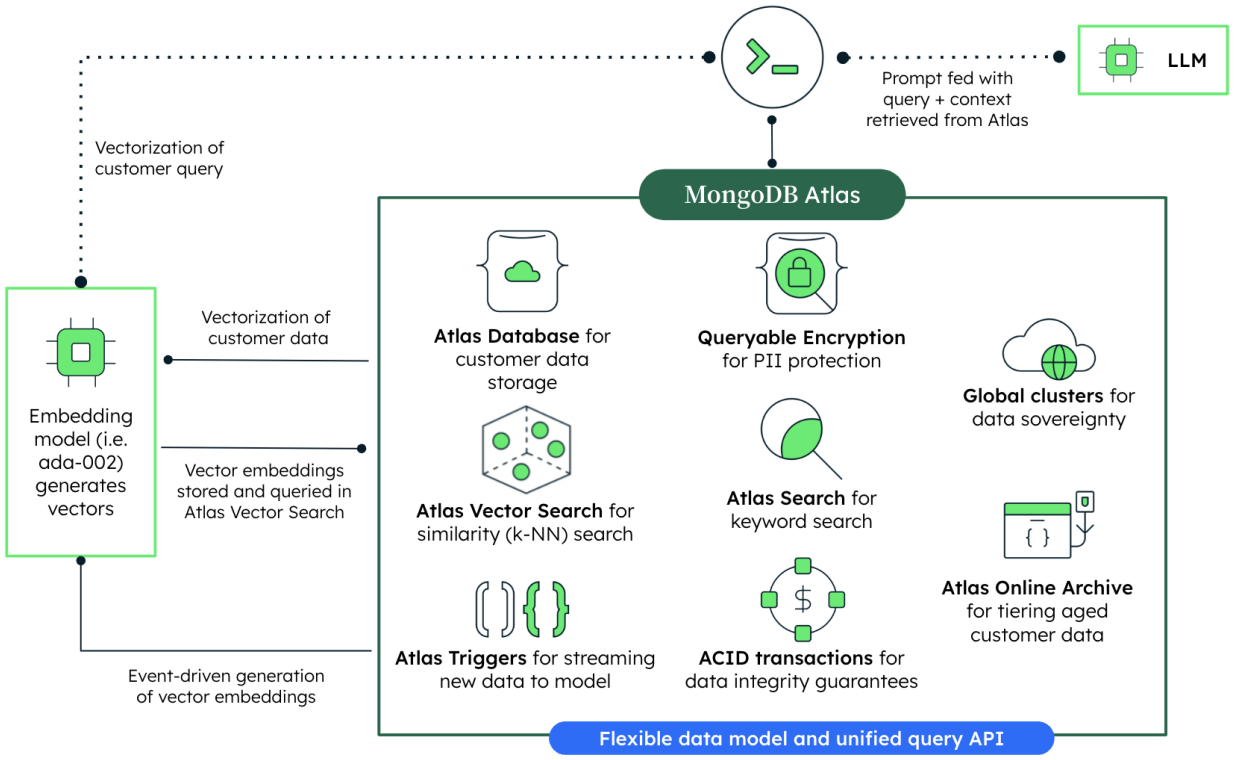

Figure 3: *Chatbot & Q-A generative AI features built into a customer self-service application powered by MongoDB Atlas*

By using Atlas, developers take advantage of MongoDB's flexible data model. They can store the source customer data, metadata, and chunks alongside the vector embeddings, all synchronized and sitting side by side in a single storage layer, and accessed by a single query API and driver.

Queries can efficiently filter data using the indexed vectors alongside keyword indexes of regular fields in your documents. This integration means the app can support a much broader range of user functionality with lower developer overhead:

- Atlas Vector Search returns matching documents by performing a similarity search on its indexed embeddings data. To reduce the risk of returning outdated data, our queries can use a vector's metadata – such as "date created" stored in the Atlas database – to filter out older content.
- Atlas Search returns results based on matching keywords in the source and chunked customer data. It uses features such as fuzzy search to correct for

typos in user input and autocomplete to provide suggested search terms. It also uses index intersection to efficiently service complex ad-hoc queries against the customer data.

Queries to the Atlas database, Vector Search, and Atlas Search all use the same query interface and driver, massively simplifying the developer workflow. The data retrieved from MongoDB Atlas is supplied as context augmenting the prompt to the LLM, allowing it to generate relevant responses to chats and questions. Context and prompts, along with any associated reasoning steps used to answer complex questions are persisted to Atlas, providing the LLM with long term memory and continually improving its outputs.

Customer data is some of the most valuable any organization manages. While generative AI helps us innovate in how we serve our customers, protecting their data remains paramount. Atlas provides a range of capabilities to help us do this, freeing developers to focus on AI-driven functionality:

- Converged infrastructure powering data storage, query and analytics, keyword search, and vector search. This unification behind a single API and data model dramatically reduces the number of moving parts developers have to integrate and build against.
- Queryable Encryption is an industry first in securing customer data. MongoDB drivers encrypt sensitive data fields client-side with the database only ever working with them as fully randomized encrypted data. Even with the data encrypted, applications can still run expressive queries against it without ever having to decrypt data in the database. Note that typically only those fields that contain the most sensitive data uniquely identifying an individual, such as an SSN, are protected with Queryable Encryption. Searching can therefore be performed on the remaining clear text fields.
- Multi-document ACID transactions in the Atlas database guarantee the integrity of our customer data whenever it is accessed and modified by the application.
- With Atlas Global Clusters, customer data can be pinned to its region of residence, complying with modern data sovereignty regulations.
- Complete data lifecycle management is provided by Atlas Online Archive. The service automatically tiers aged customer data out of active databases into lower cost cloud object storage, while keeping the data accessible for querying. This is important for customer data managed within apps operating in regulated industries where it must be retained and accessible for multiple years.
- Customer data is protected from corruption and ransomware with backups and point-in-time restore.

Atlas is fully managed for you on the major hyperscale clouds, backed by a 99.995% uptime SLA.

Advanced ecommerce search and recommendations

Ecommerce product catalogs are a common use case for MongoDB:

- The diversity of different products and their attributes naturally map to MongoDB's flexible document data model.
- Atlas' distributed architecture with elastic scale out allows developers to size and dynamically adjust database capacity in response to application demand (i.e. for shopping seasonality and sales promotions).
- With Atlas Search, keyword matching features like fuzzy search, autocomplete, faceting, highlighting and custom scoring allow shoppers to quickly browse and navigate the product catalog, driving clickthrough rates (CTRs) and buying conversions.

However, keyword search relies on matching specific words in indexed text fields in order to return relevant results. Without extensive and laborious synonym mapping (for example, mapping bikes to cycling or sneakers to trainers) users will quickly get frustrated when their search queries fail to return relevant products. This frustration translates to lost sales and damaged brand reputation.

An additional challenge is providing recommendations to users. Developers either have to write complex rules-based engines or turn to specialized and scarce data science resources. Typically data first has to be ETLed (Extract, Transform, Load) from the operational database into an offline data warehouse or data lake. Only then can traditional analytical AI models generate a set of recommendations that then have to be loaded back into the operational database. The process is complex, expensive, and generates recommendations that are instantly stale as they don't reflect the user's latest browsing behavior or purchases.

Enhancing our product catalog with vector embeddings eliminates these challenges. Vectors provide semantic meaning to the products in our catalog, making it easy to understand similarities and the relationships between products. This allows merchandisers to surface relevant and related products to users with way lower effort, complexity, and cost. Common search terms can be cached in MongoDB Atlas, serving relevant results to users, faster.

Extending vectorization to customer data – as demonstrated in the customer self-service app earlier – allows us to build even more sophisticated recommendations by combining product and customer similarity search to fine tune suggestions.

Figure 4 shows a high level design pattern for advanced search and recommendations. Creating and maintaining our vector embeddings follows the same workflow described earlier for chatbots and Q-A in our customer self-service application.

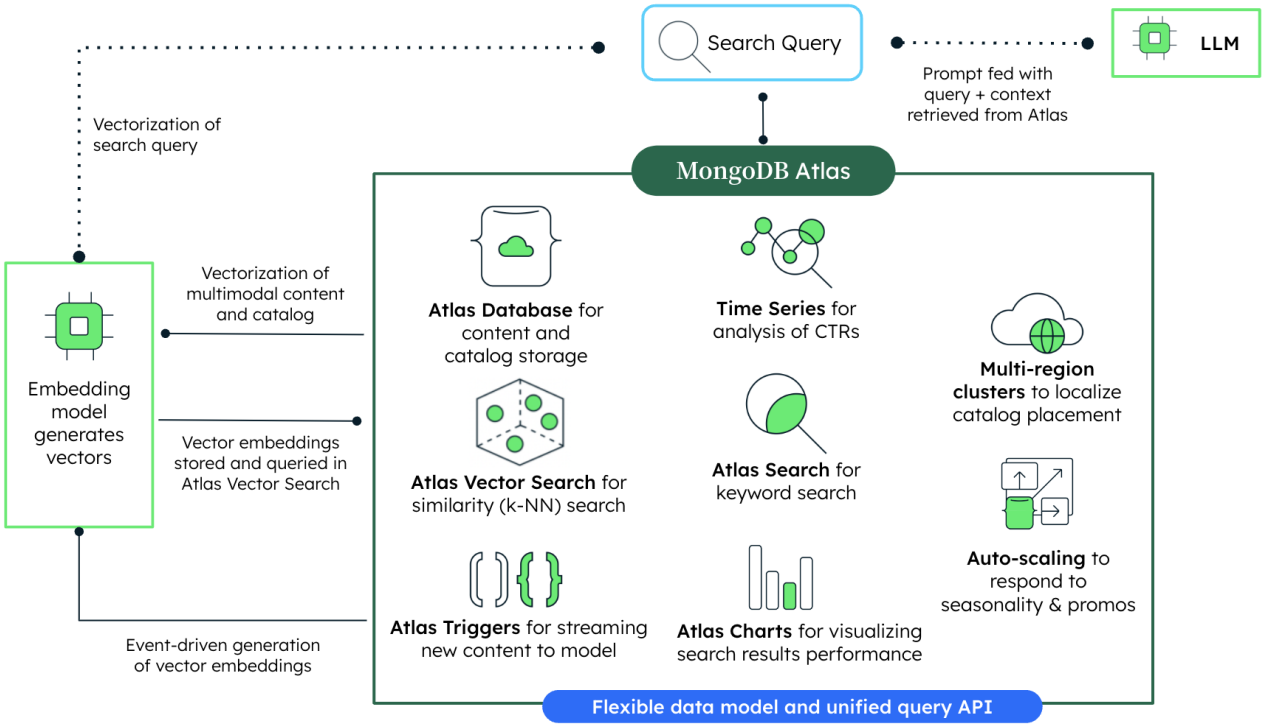

Figure 4: *Advanced semantic search on our product catalog drives higher sales conversions and upsell*

It is easy to see how vector search dramatically improves product search and recommendations. Integrating an LLM takes this experience even further. Now customers can ask live questions and get instant answers about the products they are evaluating, helping accelerate the purchase cycle.

Merchandisers can also use the LLM for a range of tasks that previously would have been laborious, freeing them up to develop even more creative ways to engage customers. For example, the LLM can be used to generate different variations of product copy and SEO keywords which can then be A/B tested to quantify which drives higher conversions. The LLM could be used to summarize multiple user reviews and infer sentiment, helping synthesize feedback that informs product roadmaps.

Organizations can use Atlas to manage the complete ecommerce lifecycle. Beyond using AI to make our search experience more intelligent and predictive, business owners can track user clickthrough rates and sales conversions from search results. Time series collections can efficiently ingest and store high velocity and voluminous clickstreams from user sessions, making that data available for analytics to measure search performance, including live visualizations of results using Atlas Charts. With these insights, merchandisers can continuously tune and optimize product data and relevance scoring to maximize sales from the ecommerce site.

Rich media (multimodal) analysis and generation

Regular text search is well served with conventional keyword search. However, working with richer media (sometimes called multimodal) assets such as images, speech, and video requires highly complex data science technology and skills. Up until now.

As noted earlier, any piece of digital content can be vectorized with the appropriate vector embedding model. AI hubs such as Hugging Face and those from the cloud hyperscalers provide a wealth of models tuned for different content modalities. The embeddings from these models can be stored in Atlas Vector Search to power a whole range of new functionality. As discussed earlier, generating images from text, transcribing videos for speech recognition and sentiment analysis, classifying images and detecting objects are just some examples of what's possible. Vectors from different media can be combined – for example comparing a text and image embedding to check whether a given sentence accurately describes an image.

This multimodal functionality can be used across a range of use cases. For example enriching product catalogs such as those described above, or enhancing discovery from analysis of images and videos. They could be used for streamlining design, manufacturing, and publishing processes, or to create completely new classes of applications in domains such as security and surveillance or augmented reality (AR).

The architectural design pattern and MongoDB Atlas capabilities described for the advanced ecommerce search and recommendations above equally apply to multimodal content generation.

MongoDB Vector Search in Action

MongoDB has already seen widespread adoption for traditional AI use cases. Continental selected MongoDB for the feature engineering platform in its Vision Zero autonomous driving initiative. Both Bosch and Telefonica use MongoDB in their AI-enhanced IoT platforms. Kronos trades billions of dollars of cryptocurrency every day using ML models configured and built with data from MongoDB. Iguazio uses MongoDB as the persistence layer for its data science and MLOps platform, while H2O.ai and Featureform support MongoDB as feature stores in their respective platforms.

Building on this foundation, MongoDB Atlas is already used today in a range of applications that are pushing the boundaries of what's possible with GenAI. Take a look at our case studies page to learn more about the breadth of use cases served by MongoDB Atlas. A selection of specific examples include:

- Ada: helps companies like Meta, ATT, and Verizon better support their customers through AI-driven automation and conversational AI.
- ExTrac: identifies and classifies emerging physical and digital risks from the analysis of real-time data streams.
- Eni: unlocks geological data and makes it actionable for better decision-making and accelerating the company's path to net zero.
- Inovaare: continuously monitors, extracts, and classifies data across the healthcare lifecycle for regulatory compliance reporting, auditing, and risk assessments.
- Source Digital: achieves 7x cost reduction after migrating from PostgreSQL to MongDB Atlas for its video detection platform.
- Catylex: automatically extracts, classifies and analyzes contract terms to identify rights, obligations, and risks
- Robust Intelligence: protects large language models (LLMs) in production by validating inputs and outputs in real-time with its AI Firewall offering.
- Potion: regenerates video and audio streams using custom vision and audio models.

Figure 5: *Retool State of AI Survey – the industry's top vector databases*

Reflecting MongoDB's popularity among AI developers, software tools vendor Retool concluded from its State of AI Survey that MongoDB Atlas Vector Search:

1. Commands the highest Net Promoter Score (NPS) of all surveyed vector databases.
2. Had risen to the second most widely used vector database within just months of its release, placing it ahead of alternative solutions that have been around for years.

"Atlas Vector Search is robust, cost-effective, and blazingly fast!"
Saravana Kumar, CEO, Kovai discussing the development of his company's AI assistant.

Getting started

Whether you are building the next big thing at a startup or enterprise, with MongoDB Atlas you can:

- Accelerate building your generative AI-enriched applications that are grounded in the truth of operational data.
- Simplify your technology stack by leveraging a single platform that allows your app to store operational data and vector embeddings in the same place, react to changes in source data with serverless functions, and search across multiple data modalities – improving relevance and accuracy in responses that the apps generate.
- Easily evolve your generative AI-enriched apps with the flexibility of the document model while maintaining a simple, elegant developer experience.
- Seamlessly integrate leading AI services and systems such as the hyperscalers and open source LLMs and frameworks to stay competitive in dynamic markets.
- Build GenAI-enriched applications on a high performance, highly scalable operational database that's had a decade of validation over a wide variety of AI use cases.

You can learn more about building AI-powered apps with MongoDB by visiting our AI/ML resource center.

The best way for developers to get started is to sign up for an account on MongoDB Atlas. From there, they can create a free MongoDB instance with the Atlas database, Atlas Vector Search and Atlas Search, load their own data or our sample data sets, and explore what's possible within the platform.

Safe Harbor

The development, release, and timing of any features or functionality described for our products remains at our sole discretion. This information is merely intended to outline our general product direction and it should not be relied on in making a purchasing decision nor is this a commitment, promise or legal obligation to deliver any material, code, or functionality.

US 866-237-8815 • INTL +1-650-440-4474 • info@mongodb.com.