zilliz

Learn / Vector Database 101: Everything You Need to Know

Introduction to Unstructured Data

Sep 29, 2022 12 min read

Buckle up for the first tutorial in our Vector Database 101 series and untangle the intricacy around Milvus with us every week.

By Zilliz

Read the entire series

Welcome to Vector Database 101.

Introduction

This is the first tutorial in the course *Vector Database 101*, and will be mostly a text-based overview of ***unstructured data***. I know, this doesn't sound like a very sexy topic, but before you press that little x button on your browser tab, hear us out.

New data is being generated every day, and is undoubtedly a key driver of both worldwide integration as well as the global economy. From heart rate monitors worn on wrists generating sensor data to GPS positions of a vehicle fleet to videos uploaded to social media, data is being generated at an exponentially increasing rate. The importance of this ever-increasing amount of data cannot be understated; data can help better serve existing customers, identify supply chain weaknesses, pinpoint workforce inefficiencies, and help companies identify and break into new markets, all factors that can enable a company (and you) to generate more revenue.

Not convinced yet? International Data Corporation - also known as *IDC* - predicts that the *global datasphere* - a measure of the total amount of new data created and stored on persistent storage all around the world - will grow to 400 zettabytes (a zettabyte = 1021 bytes) by 2028. At that time, over 30% of said data will be generated in real-time, while 80% of all generated data will be *unstructured data*.

Structured/ semi-structured/ unstructured data definition

So what exactly is unstructured data? As the name suggests, unstructured data refers to data that cannot be stored in a pre-defined format or fit into an existing data model. Human-generated data - images, video, audio, text files, etc - are great examples of unstructured data. But there are a variety of less mundane examples of unstructured data too. Protein structures, executable file hashes, and even human-readable code are three of a near-infinite set of examples of unstructured data.

Structured data, on the other hand, refers to data that can be stored in a table-based format, while semi-structured data refers to data that can be stored in single- or multi-level array/key-value stores. If none of this makes sense to you yet, don't fret. Bear with us and we'll provide examples to help solidify your understanding of the key differences between stuctured and unstructured data.

Some concrete examples of structured data

Still with us? Excellent - let's start by briefly describing structured/semi-structured data. In the simplest terms, traditional structured data can be stored via a relational model. Take, for example, a book database:

ISBN	Year	Name	Author
0767908171	2003	A Short History of Nearly Everything	Bill Bryson
039516611X	1962	Silent Spring	Rachel Carson
0374332657	1998	Holes	Louis Sachar
...			

Ahh, _Holes_. Brings back childhood memories.

In the example above, each row within the database represents a particular book (indexed by ISBN number), while the columns denote the corresponding category of information. Databases built on top of the relational model allow for multiple tables, each of which has its own unique set of columns. These tables are formally known as ***relations***, but we'll just call them tables to avoid confusing databases with friends and family members. Two of the most popular and well-known examples of relational databases are *MySQL* (released in 1995) and *PostgreSQL* (released in 1996).

Semi-structured data is the subset of structured data that does not conform to the traditional table-based model. Instead, semi-structured data usually comes with keys or markers which can be used to describe and index the data. Going back to the example of a book database, we can expand it to a semi-structured JSON format as so:

```
{
  ISBN: 0767908171
  Month: February
  Year: 2003
  Name: A Short History of Nearly Everything
  Author: Bill Bryson
  Tags: geology, biology, physics
},
{
  ISBN: 039516611X
  Name: Silent Spring
  Author: Rachel Carson
},
{
  ISBN: 0374332657
  Year: 1998
  Name: Holes
  Author: Louis Sachar
},
...
```

Note how the first element in our new JSON database now contains `Months` and `Tags` as two extra pieces of information, without impacting the two subsequent elements. With semi-structured data, this can be done without the extra overhead of two additional columns for all elements, thereby allowing for greater flexibility.

Semi-structured data is typically stored in a *NoSQL database* (wide-column store, object/document database, key-value store, etc), as their non-tabular nature prevents direct use in a relational database. *Cassandra* (released in 2008), *MongoDB* (released in 2009), and *Redis* (released in 2009) are three of the most popular databases for semi-structured data today. Note how these popular databases for semi-structured data were released a little over a decade after popular databases for structured data - keep this in mind as we'll get to it later.

A paradigm shift — Unstructured Data Definition

Now that we have a solid understanding of structured/semi-structured data, let's move to talking about unstructured data. Unlike structured/semi-structured data, unstructured data can take any form, be of an arbitrarily large or small size on disk, and can require vastly different runtimes to transform and index. Let's take images as an example: three front-facing successive images of the same German Shepherd are *semantically the same*.

Semantically the same? What on earth does that mean? Let's dive a bit deeper and unpack the idea of *semantic similarity*. Although these three photos may have vastly different pixel values, resolutions, file sizes, etc, all three photos are of the same German Shepherd in the same environment. Think about it - all three photos have identical or near-identical content but significantly different raw pixel values. This poses a new challenge for industries and companies

that use data[1]: how can we transform, store, and search unstructured data in a similar fashion to structured/semi-structured data?

At this point, you're probably wondering: how can we search and analyze unstructured data if it has no fixed size or format? The answer: machine learning (or more specifically, deep learning). In the past decade, the combination of big data and deep neural networks has fundamentally changed the way we approach data-driven applications; tasks ranging from spam email detection to realistic text-to-video synthesis have seen incredible strides, with accuracy metrics on certain tasks reaching superhuman levels. This may sound scary (hello, Skynet), but we're still many decades away from Elon Musk's vision of AI taking over the world.

[1]In essence, this is all industries, all companies, and all inviduals. Including you!

Examples of Unstructured data

Unstructured data can be generated by machines or by humans. Machine-generated unstructured data examples include:

- Sensor data: Data collected from sensors, such as temperature sensors, humidity sensors, GPS sensors, and motion sensors.
- Machine log data: Data generated by machines, devices, or applications, including system logs, application logs, and event logs.
- Internet of Things (IoT) data: Data collected from smart devices, such as smart thermostats, smart home assistants, and wearable devices.
- Computer vision data: This is unstructured data generated by computer vision technologies, such as image recognition, object detection, and video analysis.
- Natural Language Processing (NLP) data: This is data generated by NLP technologies, such as speech recognition, language translation, and sentiment analysis.
- Web and application data: Data generated by web servers, web applications, and mobile applications, including user behavior data, error logs, and application performance data.

Examples of human-generated unstructured data include:

- Emails: Email messages are often unstructured and can contain free-form text, images, and attachments.
- Text messages: Text messages can be informal, unstructured, and contain abbreviations or emojis.
- Social media posts: Social media posts can vary in structure and content, including text, images, videos, and hashtags.
- Audio recordings: Human-generated audio recordings can include phone calls, voicemails, audio files and audio notes are unstructured data.
- Handwritten notes: Handwritten notes can be unstructured and contain drawings, diagrams, and other visual elements.
- Meeting notes: Meeting notes can contain unstructured text, diagrams, and action items.

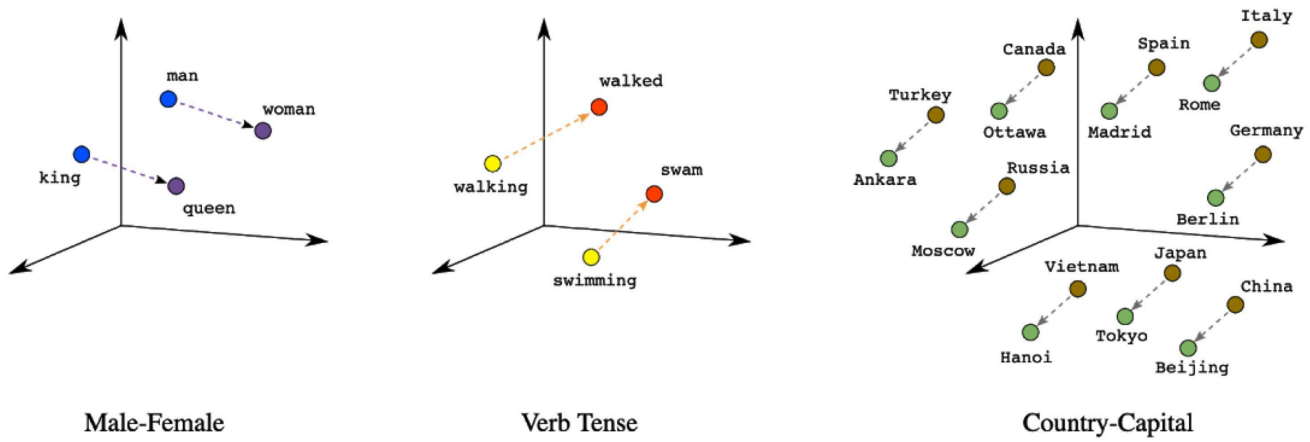
- Transcripts: Transcripts of speeches, interviews, and meetings can contain unstructured text with varying degrees of accuracy.
- User-generated content: User-generated content on websites and forums can be unstructured data and include free-form text, images, and video files.

A crash course on embeddings

Let's get back on track. The vast majority of neural network models are capable of turning a single piece of unstructured data into a list of floating point values, also known more commonly as an *embeddings* or *embedding vectors*. As it turns out, a properly trained neural network can output embeddings that represent the semantic content of the image2. In a future tutorial, we'll go over a vector database use case that uses a pre-determined algorithm to generate embeddings.

An Eastern Towhee. Photo by Patrice Bouchard.

The photo above provides an example of transforming a piece of unstructured data into a vector. With the preeminent ResNet-50 convolutional neural network, this image can be represented as a vector of length 2048 - here are the first three and last three elements: `[0.1392, 0.3572, 0.1988, ..., 0.2888, 0.6611, 0.2909]`. Embeddings generated by a properly trained neural network have mathematical properties which make them easy to search and analyze. We won't go too much into detail here, but know that, generally speaking, embedding vectors for semantically similar objects are *close to each other in terms of distance*. Therefore, searching across and understanding unstructured data boils down to vector arithmetic.



Embedding arithmetic in action.

As mentioned in the introduction, unstructured data will comprise a whopping 80% of all newly created data by the year 2028. This proportion will continue to increase beyond 80% as industries mature and implement methods for unstructured data processing. This impacts everybody - you, me, the companies that we work for, the organizations that we volunteer for, so on and so forth. Just as new user-facing applications from 2010 onward required databases for storing semi-structured data (as opposed to traditional tabular data), this decade necessitates databases purpose-built for indexing and searching across massive quantities (exabytes) of unstructured data.

The solution? A database for the AI era - a *vector database*. Welcome to our world; welcome to the world of *the open-source vector database, Milvus*.

2In most tutorials, we'll focus on embeddings generated by neural networks; do note, however, that embeddings can be generated through handcrafted algorithms as well.

Unstructured data processing

Excited yet? Excellent. But before we dive headfirst into vector databases and Milvus, let's take a minute to talk about how we process and analyze unstructured data. In the case of structured and semi-structured data, searching for or filtering items in the database is fairly straightforward. As a simple example, querying MongoDB for the first book from a particular author can be done with the following code snippet (using `pymongo`):

```
>>> document = collection.find_one({'Author': 'Bill Bryson'})
```

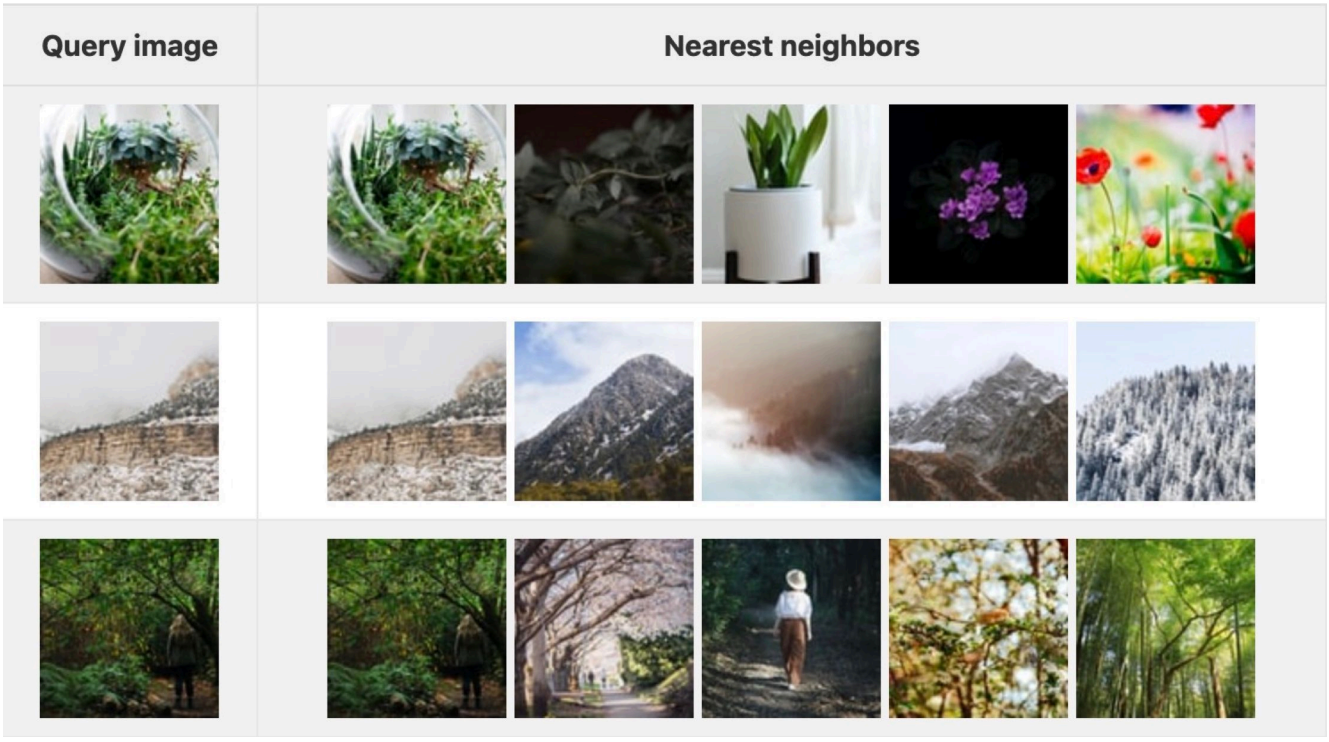
This type of querying methodology is not dissimilar to that of traditional relational databases, which rely on SQL statements to filter and fetch data. The concept is the same: databases for structured/semi-structured data perform filtering and querying using mathematical (e.g. `<=`, string distance) or logical (e.g. `EQUALS`, `NOT`) operators across numerical values and/or strings. For traditional relational databases, this is called *relational algebra*; for those of you unfamiliar with it, trust me when I say it's much worse than linear algebra. You may have seen examples of extremely

complex filters being constructed through relational algebra, but the core concept remains the same - traditional databases are *deterministic* systems that always return exact matches for a given set of filters.

Unlike databases for structured/semi-structured data, vector database queries are done by specifying an input *query vector* as opposed to SQL statement or data filters (such as `{'Author': 'Bill Bryson'}`). This vector is the embedding-based representation of the unstructured data. As a quick example, this can be done in Milvus with the following snippet (using `pymilvus`):

```
>>> results = collection.search(embedding, 'embedding', params, limit=10)
```

Internally, queries across large collections of unstructured data are performed using a suite of algorithms collectively known as *approximate nearest neighbor search*, or *ANN search* for short. In a nutshell, ANN search is a form of optimization that attempts to find the "closest" point or set of points to a given query vector. Note the "approximate" in ANN. By utilizing clever indexing methods, vector databases have a clear accuracy/performance tradeoff: increasing search runtimes will result in a more consistent database that performs closer to a deterministic system, always returning the absolute nearest neighbors given a query value. Conversely, reducing query times will improve throughput but may result in capturing fewer of a query's true nearest values. In this sense, unstructured data processing is a *probabilistic* process[3].



Approximate nearest neighbor search, visualized.

ANN search is a core component of vector databases and a massive research area in and of itself; as such, we'll dive deep into various ANN search methodologies available to you within Milvus in a future set of articles.

3Vector databases can be made deterministic by selecting a specific index.

Wrapping up

Thanks for making it this far! Here are the key takeaways for this tutorial:

- Structured/semi-structured data are limited to numeric, string, or time data types. Through the power of modern machine learning, unstructured data is represented as high-dimensional vectors of numerical values.
- These vectors, more commonly known as embeddings, are great for representing the semantic content of the unstructured data. Structured/semi-structured data, on the other hand, is semantically as-is, i.e. the content itself is equivalent to the semantics.
- Searching and analyzing unstructured data is done through ANN search, a process that is inherently probabilistic. Querying across structured/semi-structured data, on the other hand, is deterministic.
- Unstructured data processing is very different from semi-structured data processing, and requires a complete paradigm shift. This naturally necessitates a new type of database - the vector database.

This concludes part one of this introductory series - for those of you new to vector databases, welcome to Milvus! In the next tutorial, we'll cover vector databases in more detail:

- We'll first provide a birds-eye view of the the Milvus vector database.
- We'll then follow it up with how Milvus differs from vector search libraries (FAISS, ScaNN, DiskANN, etc).
- We'll also discuss how vector databases differ from vector search plugins (for traditional databases and search systems).
- We'll wrap up with technical challenges associated with modern vector databases.

See you in the next tutorial, What is a Vector Database?.

Take another look at the Vector Database 101 courses

01. Introduction to Unstructured Data
02. What is a Vector Database?
03. Comparing Vector Databases, Vector Search Libraries, and Vector Search Plugins
04. Introduction to Milvus
05. Milvus Quickstart
06. Introduction to Vector Similarity Search
07. Vector Index Basics and the Inverted File Index
08. Scalar Quantization and Product Quantization

Zilliz

Next: What is a Vector Database and How Does It Work? →

Zilliz Cloud BYOC
Free Tier
Milvus
GPTCache
Attu
Milvus CLI
Integrations

Blog
GitHub ↗
Benchmark ↗
Comparison
Resources
Glossary
What is a Vector Database
What is Retrieval Augmented Generation
GenAI Resource Hub

News
Partners
Events
Contact Sales

zilliz