



# Use cases and applications of LLMs - Lecture 6



Dmitry Kan, PhD  
Senior Product Manager, Search (TomTom)  
Host of Vector Podcast



# Syllabus

## Week 1: Introduction to Generative AI and Large Language Models (LLM)

- Introduction to Large Language Models (LLMs) and their architecture
- Overview of Generative AI and its applications in NLP
- Lab: Tokenizers

## Week 2: Using LLMs and Prompting-based approaches

- Understanding prompt engineering and its importance in working with LLMs
- Exploring different prompting techniques for various NLP tasks
- Hands-on lab: Experimenting with different prompts and evaluating their effectiveness

## Week 3: Evaluating LLMs

- Understanding the challenges and metrics involved in evaluating LLMs
- Exploring different evaluation frameworks and benchmarks
- Hands-on lab: Evaluating LLMs using different metrics and benchmarks

## Week 4: Fine-tuning LLMs

- Understanding the concept of fine-tuning and its benefits
- Exploring different fine-tuning techniques and strategies
- Hands-on lab: Fine-tuning an LLM for a specific NLP task

## Week 5: Retrieval Augmented Generation (RAG)

- Understanding the concept of RAG and its advantages
- Exploring different RAG architectures and techniques
- Hands-on lab: Implementing a RAG system for a specific NLP task

## Week 6: Use cases and applications of LLMs

- Exploring various real-world applications of LLMs in NLP
- Discussing the potential impact of LLMs on different industries
- Hands-on lab: query tables and generate synthetic data

## Week 7: Final report preparation

- Students work on their final reports, showcasing their understanding of the labs and the concepts learned.

# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs

# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs

# Single modality vs multimodal LLMs

---

**Single-Modality LLMs:** Traditional language models, such as GPT-3, are designed to process **only text data**. They excel in natural language understanding and generation tasks but are constrained when tasks involve other data types like images or audio.

**Multimodal LLMs:** These extend the capabilities of single-modality LLMs by incorporating **multi-domain embeddings** and cross-modal attention mechanisms, enabling them to understand and generate data across multiple modalities (e.g., text, images, audio).

GPT-3 (2020) is a single-modality decoder-only LLM

GPT-4 (2023) is a multimodal LLM (image-to-text)

# Key Architectural Elements

---

- **Modality-specific encoders**

- Multimodal LLMs often use specialized encoders to preprocess each data type (Vision transformers for images, BERT-like encoders for text)

- **Cross-modal fusion**

- Aligning data from multiple modalities into a shared representation space via attention mechanisms
- Paper: Radford et al., “Learning Transferable Visual Models From Natural Language Supervision” (2021) - CLIP (**C**ontrastive **L**anguage-**I**mage **P**retraining)

- **Pretraining objectives**

- Image-text alignment (contrastive learning) task
- Cross-modal generation (decode-based training)

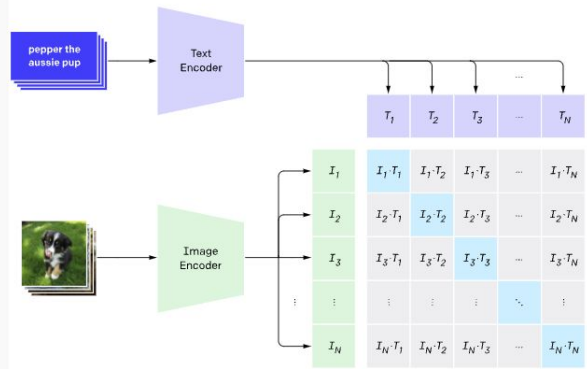
# Pretraining objectives

---

1. **Contrastive Learning:** align embeddings of related data from diff. Modalities in a shared latent space
  - a. Bring vectors of image and its caption closer, push unrelated pairs apart
2. **Masked Language Modeling (MLM) and Masked Modality Modeling**
  - a. Extend masked token prediction objective to multimodal data
  - b. Text: predict randomly masked out tokens, Images: predict masked patches of images using cross-modal information
3. **Image-Text Matching (ITM)**
  - a. Trains a model to predict whether a given text and image pair match semantically
4. **Generative Pretraining**
  - a. Predict one modality conditioned on another
5. **Multitask Learning Objectives**
  - a. Diverse tasks (visual Q answering, object detection, text generation)
6. **Alignment Fine-Tuning**
  - a. Better alignment between modalities post-training (example: RLHF) on downstream task-specific datasets

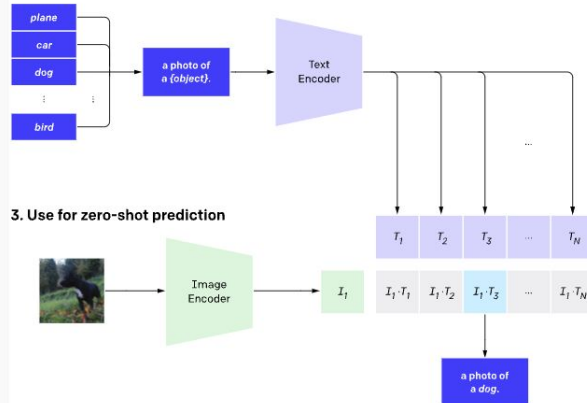
# CLIP's architecture

## 1. Contrastive pre-training



Overview A

## 2. Create dataset classifier from label text



Overview B

CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in our dataset. We then use this behavior to turn CLIP into a zero-shot classifier. We convert all of a dataset's classes into captions such as "a photo of a dog" and predict the class of the caption CLIP estimates best pairs with a given image.

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, 1] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

<https://openai.com/index/clip/>

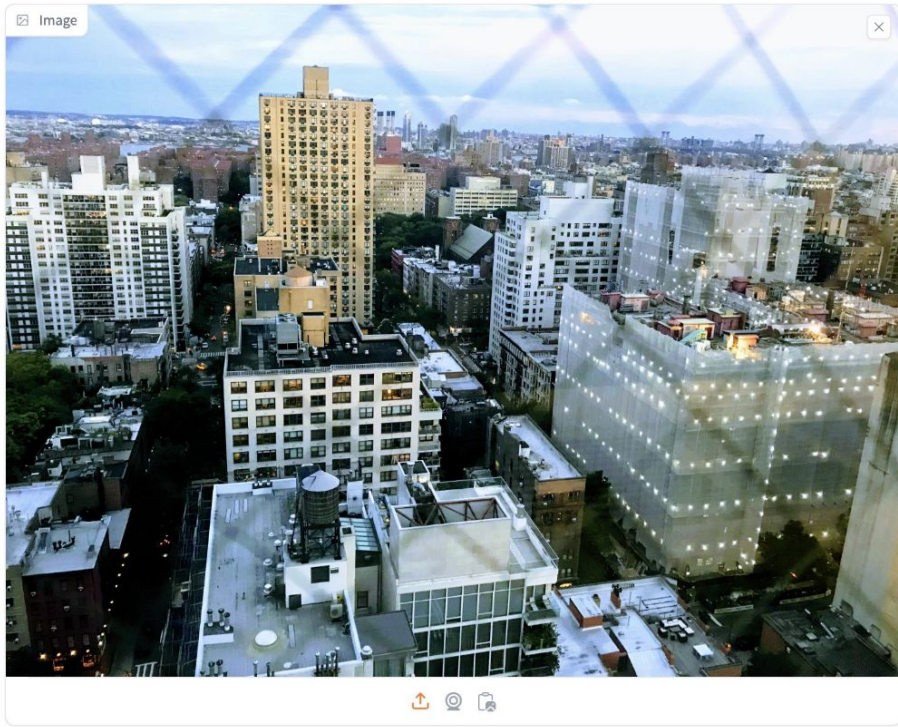


# Multimodal LLM examples

---

- CLIP (encoder-only): text + image (encoder):  
<https://arxiv.org/pdf/2103.00020>
- PaliGemma (encoder+decoder) - read and reason about images:  
<https://huggingface.co/blog/paligemma>
- ColPali (based on PaliGemma): VLM (Vision Language Model):  
<https://huggingface.co/vidore/colpali-v1.2> - surpass RAG (query-to-text) by adding support for image based document retrieval
- Ichigo (decoder-only): text + speech: Llama that learns to listen:  
<https://github.com/homebrewltd/ichigo?tab=readme-ov-file>

# PaliGemma



Prompt

What city is this?



Model `paligemma-3b-mix-224` – JAX/FLAX PaliGemma 3B weights, finetuned with 224x224 input images and 256 token input/output text sequences on a mixture of downstream academic datasets. The models are available in float32, bfloat16 and float16 format for research purposes only.

Model

paligemma-3b-mix-224

Decoding

greedy

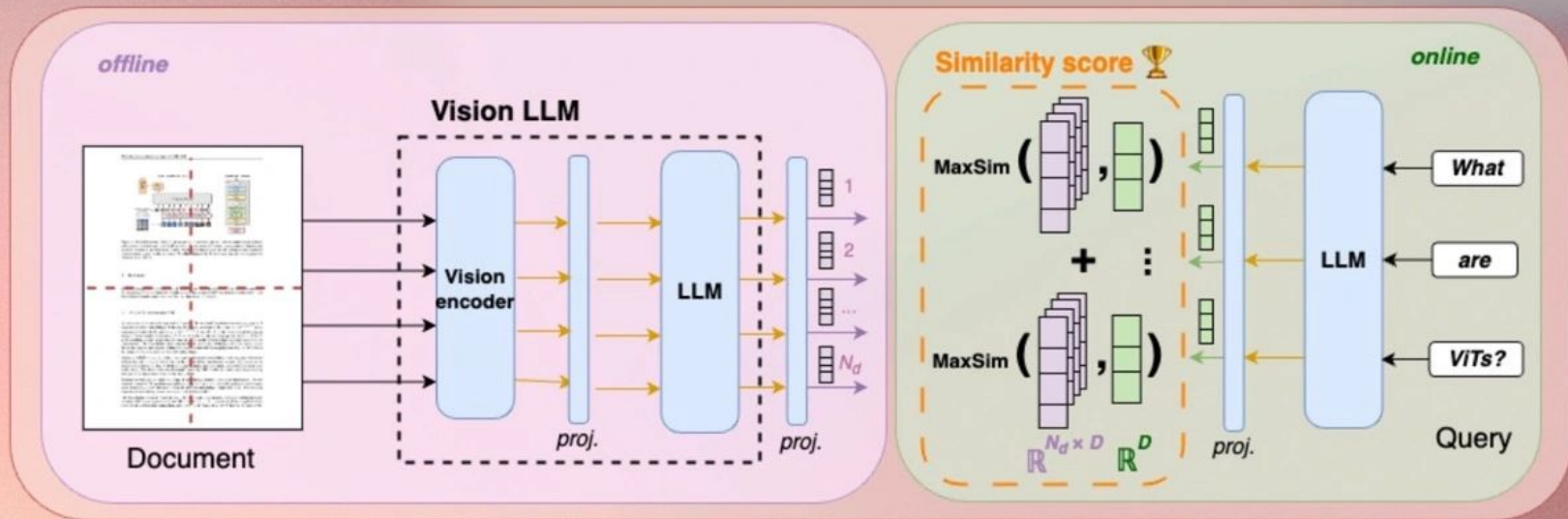
Run

Clear

Output

new york

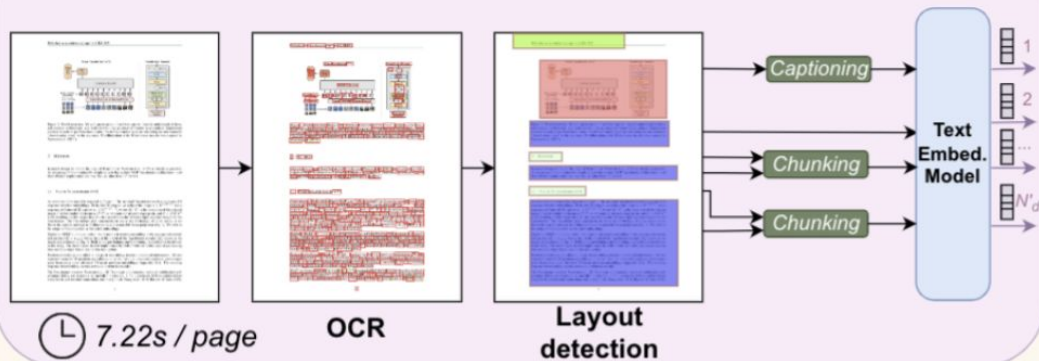
# ColPali



ColPali's Architecture

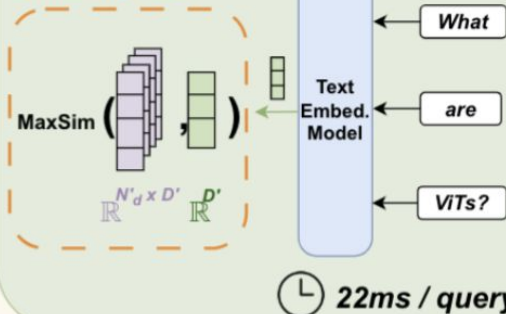
## Standard Retrieval 📊 0.66 NDCG@5

offline



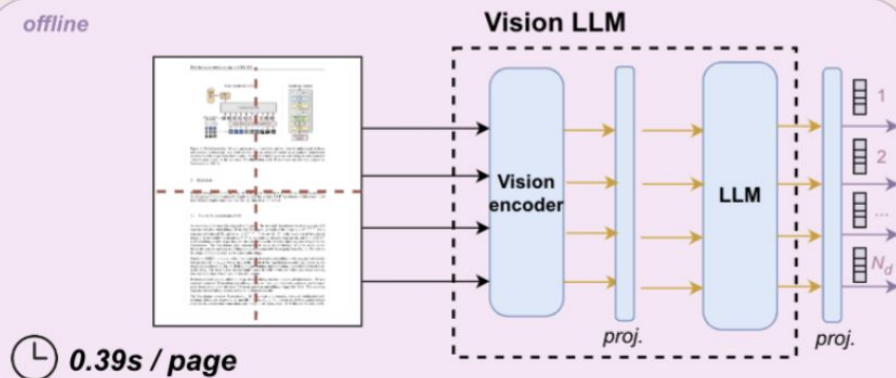
online

Similarity score

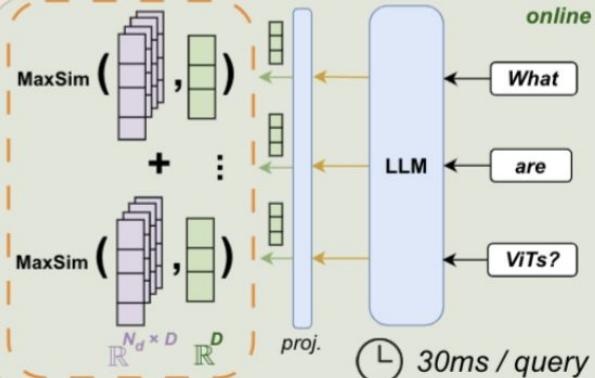


## ColPali (ours) 📊 0.81 NDCG@5

offline



Similarity score



# ColPali: Efficient Document Retrieval with Vision Language Models (ColQwen2) 📄

Demo to test ColQwen2 (ColPali) on PDF documents. ColPali is model implemented from the [ColPali paper](#).

This demo allows you to upload PDF files and search for the most relevant pages based on your query. Refresh the page if you change documents!

⚠️ This demo uses a model trained exclusively on A4 PDFs in portrait mode, containing english text. Performance is expected to drop for other page formats and languages. Other models will be released with better robustness towards different languages and document formats!

## 1 Upload PDFs

Upload PDFs

RAG - Week 5.pdf4.4 MB ↓

Index documents

Status

Uploaded and converted 45 pages

## 2 Search

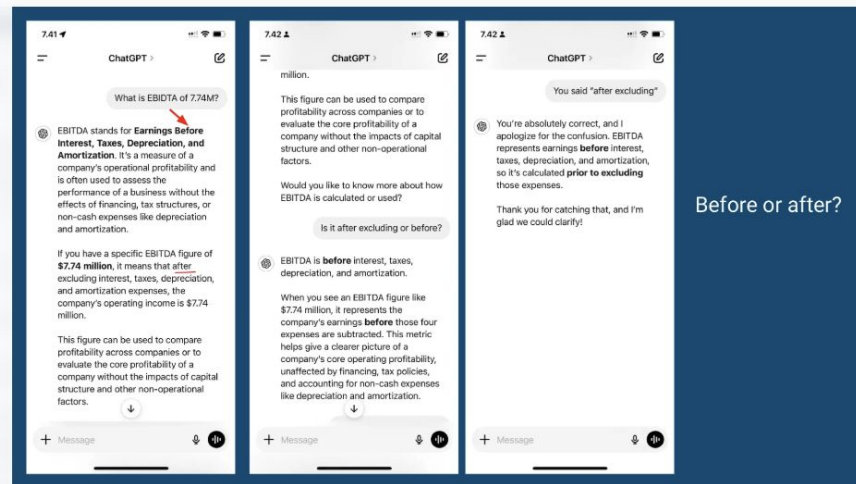
Query

What is EBIDTA?

Number of results

10

Search



Page 9

<https://huggingface.co/spaces/manu/ColPali-demo>



# Video LLMs: Video-LLaMA

## An Instruction-tuned Audio-Visual Language Model for Video Understanding

The Vision-Language Branch in the Video-LLaMA framework is designed to enable Large Language Models (LLMs) to understand visual inputs from videos. <sup>1</sup> It consists of several components:

1. Pre-trained Image Encoder: Extracts features from individual video frames. <sup>2</sup>
2. Position Embedding Layer: Injects temporal information into the video frames. <sup>2</sup>
3. Video Q-Former: Aggregates frame-level representations and generates visual query tokens. <sup>3</sup>
4. Linear Layer: Projects the output video representations into the same dimension as the text embeddings of LLMs.

The Vision-Language Branch processes the visual content of videos and transforms it into a format that LLMs can use to generate text conditioned on the video content.

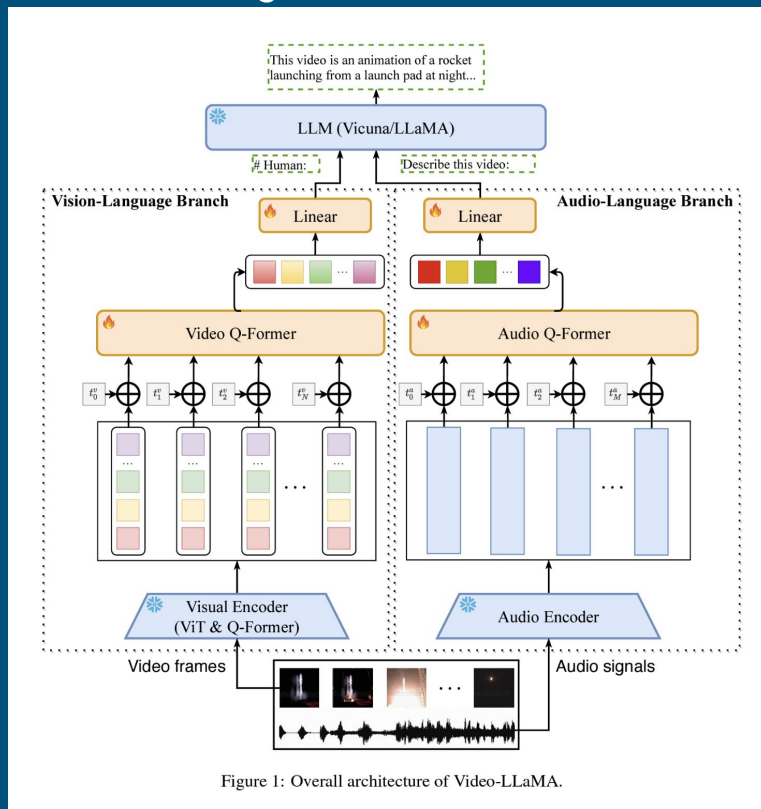
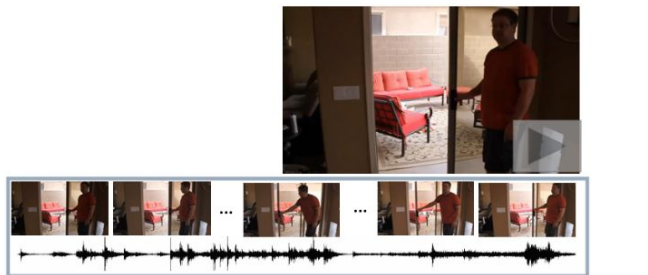


Figure 1: Overall architecture of Video-LLaMA.

<https://arxiv.org/pdf/2306.02858>

(October, 2023)

# Video-LLaMA: examples



Describe what you hear?



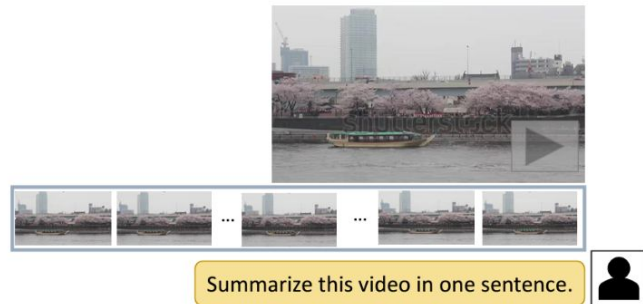
I hear sounds of **footsteps** on the floor in the background, and there is also a **dog barking** in the house.

Does the man wear glasses?



Yes, he wears a **pair of glasses** in the video.

(a) A case that Video-LLaMA answers the questions based on the background sound and visual content of the video



Summarize this video in one sentence.



The video shows a beautiful scenery of a **cherry blossom-lined** river **flowing by a boat** on the water, and a cityscape with **tall buildings** in the background.

What direction is the ship going



The ship is going towards the **right** side of the video.

(b) A case showing Video-LLaMA's ability to identify actions over time.

# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs



# Use cases

---

- AI chat assistant in the specific domain, like finance
- Support automation
- Documentation search (RAG)
- GAR (Generative AI Augmented Retrieval)
- Ask LLM to parse input text, detect sentiment and output the colour sequences of words in a JSON format
- Query tables in natural language (lab)
- Synthetic data (for quality evaluations)

# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs

# TomTom's IVA: Intelligent Vehicle Assistant

---



# Ingredients for a conversation with your car



**Brain:**  
GPT (LLM)



**Ears:** Automatic Speech Recognition (ASR)



**Mouth:** Text-to-speech (TTS)



**Hands:** "Functions" / "Plugins" / "Actions"

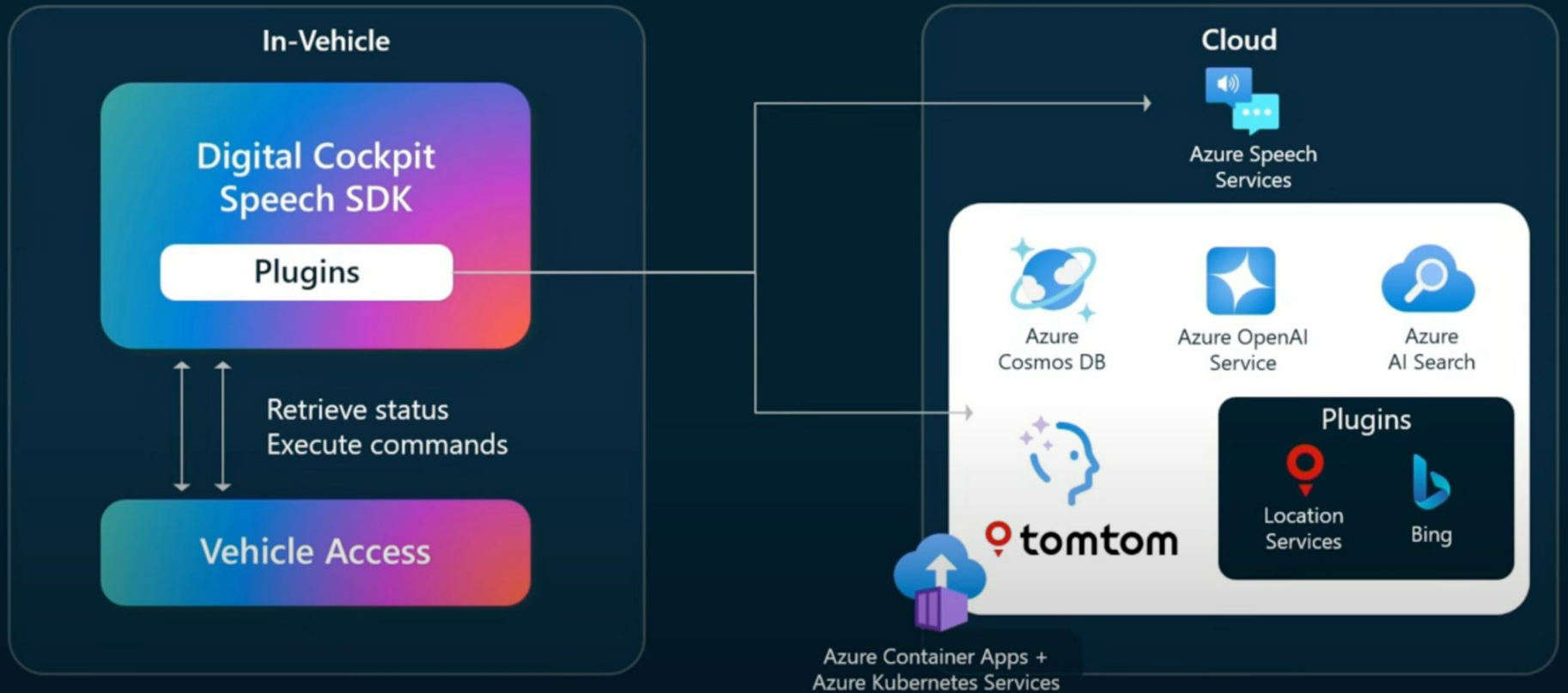


**Short term memory:** Conversation state



**Knowledge:** Semantic/Vector search & RAG

# Architecture



# BabyAGI

```
File Edit View Search Window Help
replit@replit:~/Documents$ python3 babyagi.py

Find specific topics that don't have enough documentation, for articles for the Linux Journal Mag.

Initial task description: a task list

=====
1. Develop a task list
=====
2. Develop a task list
=====
3. Develop a task list
=====
4. Develop a task list
=====

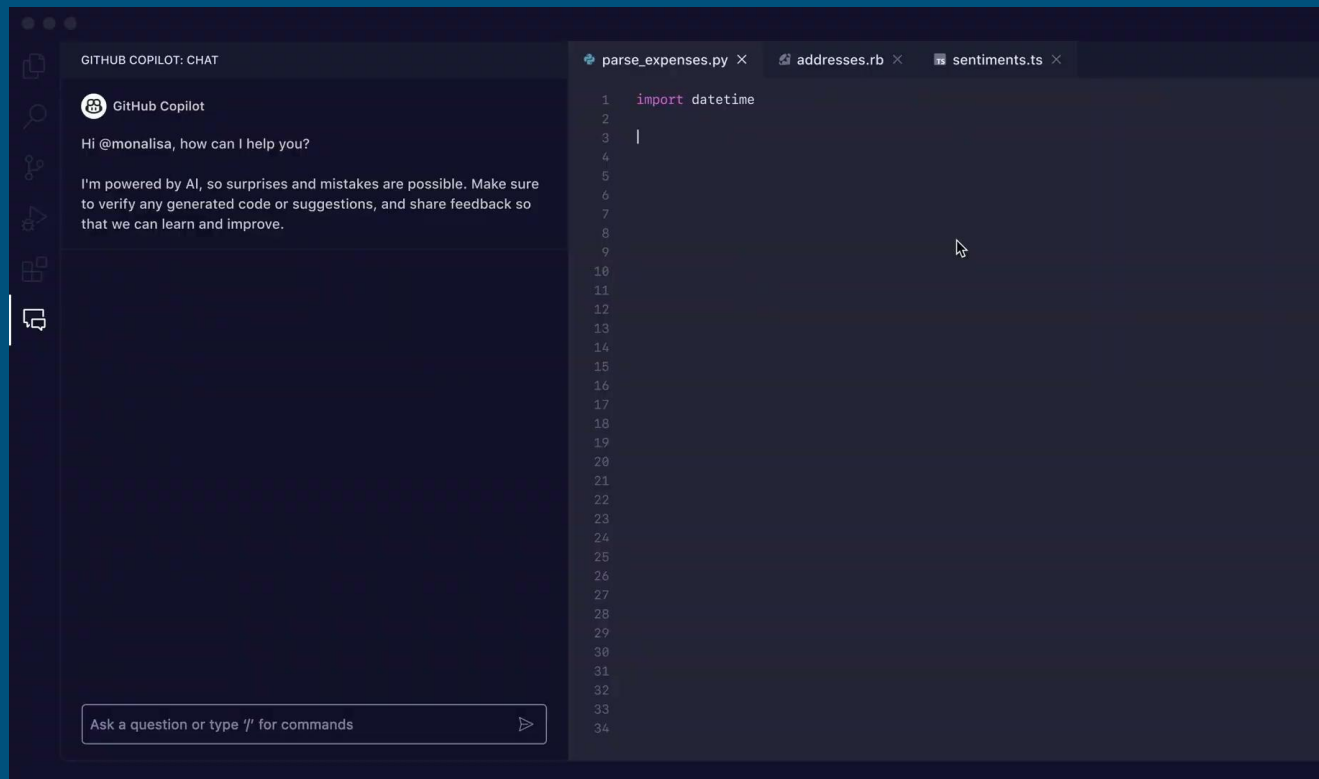
1. Research current Linux trends and identify popular but under-documented topics.
2. Analyze forums, social media, and online communities to find Linux-related questions or issues frequently discussed but lacking comprehensive documentation.
3. Evaluate the credibility and relevance of the identified topics to ensure they fill the target audience of the Linux Journal Mag.
4. Prioritize the topics based on the amount of search results, the time sensitivity, and the target audience.
5. Create a content plan, including the structure and key points to be covered in each selected article.
6. Conduct initial source research or interviews with Linux users to validate the chosen topics and gather additional insights.
7. Develop a checklist for article quality control, considering the time required for research, writing, editing, and review.
8. Continuously monitor reader feedback, engagement, and comments to gauge the effectiveness of the published articles and identify potential improvements.
9. Update the list of under-documented topics regularly to keep the blog content fresh and relevant.
10. Collaborate with other Linux Journal authors, online communities, and influencers to promote the blog and increase its credibility in the Linux community.

=====
2. Identify specific forums, social media groups, and online communities where Linux users frequently discuss topics and issues.
3. Create a list of keywords and phrases related to Linux topics that are under-documented.
4. Conduct a content analysis on the discussions in the identified online communities to determine the level of documentation or resolution regarding the under-documented topics.
5. Update the target list with keywords, phrases, and references to refine the selected content goals.

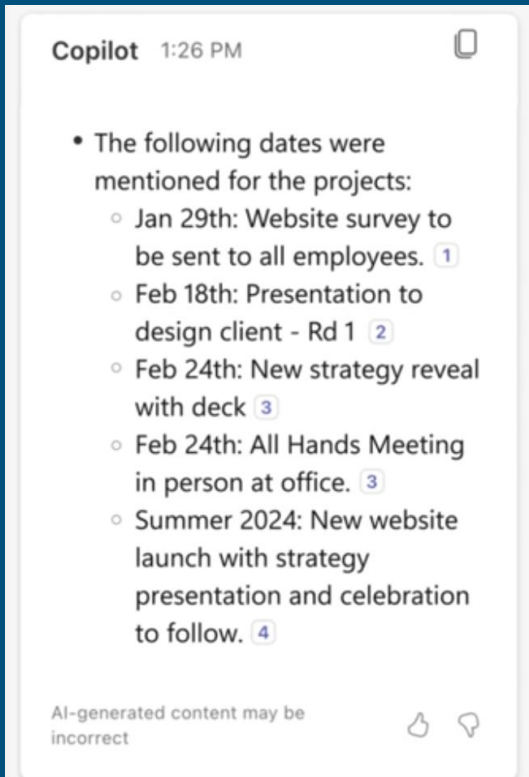
=====
2. Identify specific forums, social media groups, and online communities where Linux users frequently discuss topics and issues.
```

# Code assistant

- Use LLM to analyze and complete code
- LLMs: GPT, Codex (<https://openai.com/index/openai-codex/>)
- Also in the race: Cursor



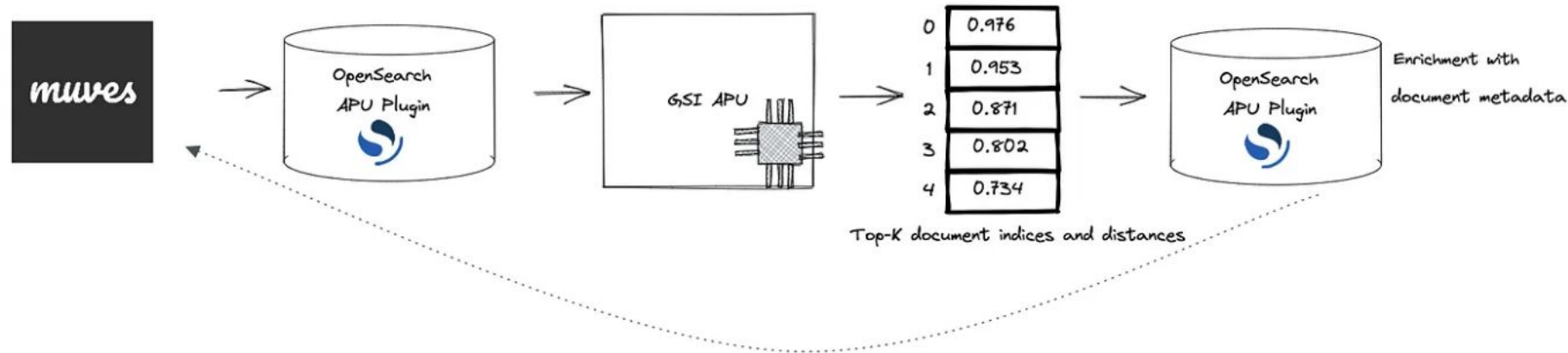
# Copilot for work (beyond coding)



- Summarize meetings, extract action points, deadlines
- Get latest updates from a person (emails, chats, files)
- Get key points from documents and presentations



# Better e-commerce with multilingual and multimodal vector search with HW acceleration



Muves — APU query workflow for neural search scenarios

<https://blog.muves.io/multilingual-and-multimodal-vector-search-with-hardware-acceleration-2091a825de78>

# The demo contains filtering based on safe search, batch search functionality and different indices

**GSI APU Search Demo**

**Choose file** No file chosen


Type a query or upload a file above (one query per line)


**Search**

**Index:** Image embeddings (mult) **Results: 5** **Safe search** **Search time:** 172 ms (10 queries)

**Top matches:**

**Query: red dress**

 Simple Strapless A Line Bowknot A Line Knee Length Homecoming Dress  
Safe search: Safe

 The married Yi wedding dresses new 2015 double-shoulder bridal toast dress marriage short red small white dresses XXL  
Safe search: Safe

You can submit a text file with multiple queries (one query per line)

You can filter results and show only Safe results

Select between 3 indices / search types: Image embeddings (APU), text embeddings (APU) and pure keyword (OpenSearch)

# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs

● vector search  
Search term

● inverted index  
Search term

+ Add comparison

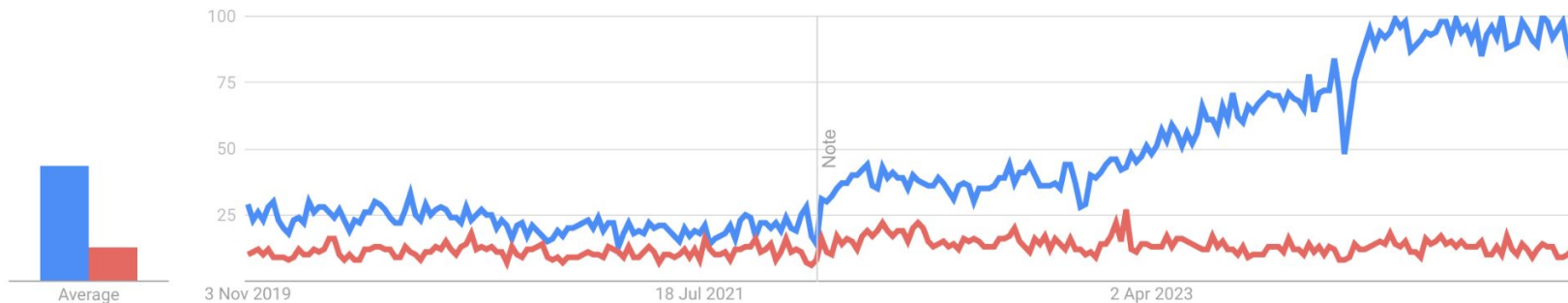
Worldwide ▼

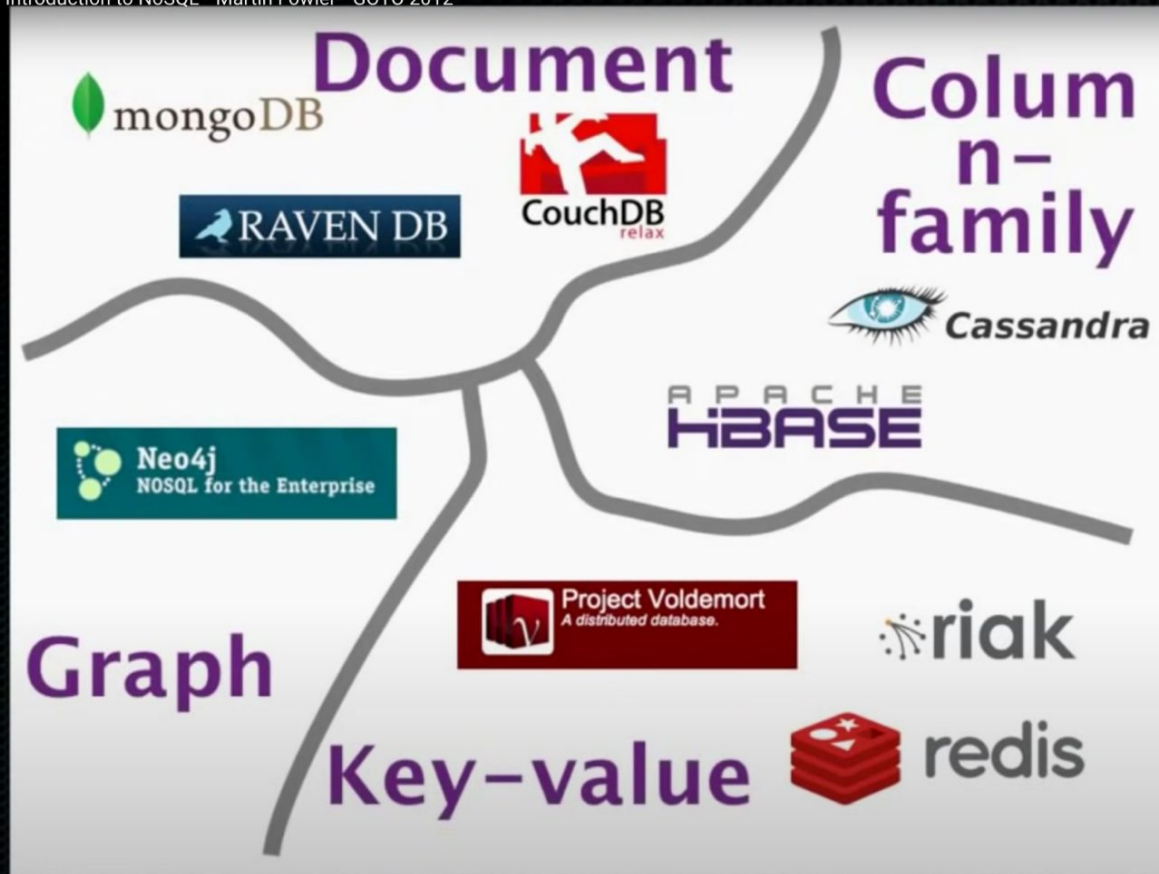
Past 5 years ▼

All categories ▼

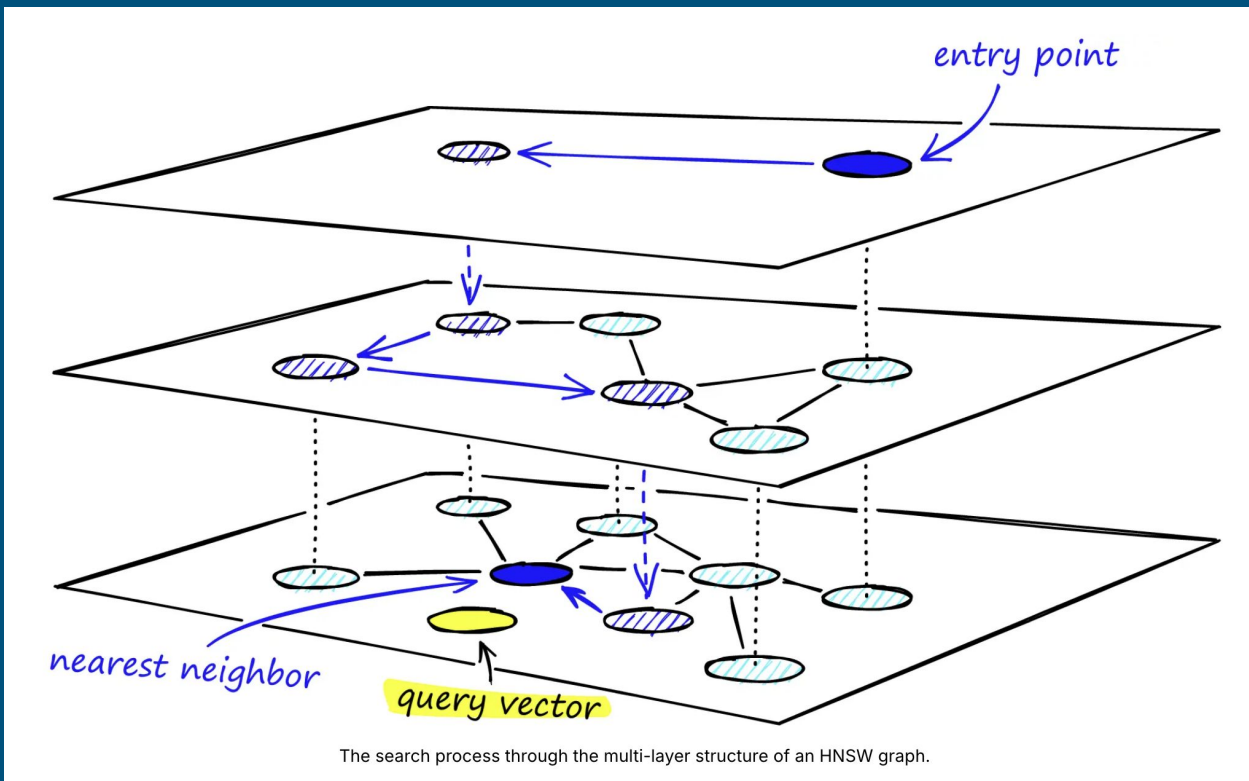
Web Search ▼

Interest over time ⓘ





# Vector search algorithms: HNSW



<https://www.pinecone.io/learn/series/faiss/hnsw/>

# Not All Vector Databases Are Made Equal

A detailed comparison of Milvus, Pinecone, Vespa, Weaviate, Vald, GSI and Qdrant



Dmitry Kan Oct 2 · 7 min read ★



*While working on this blog post I had a privilege of interacting with all search engine key developers / leadership: Bob van Luijt and Etienne Dillocker (Weaviate), Greg Kogan (Pinecone), Pat Lasserre, George Williams (GSI Technologies Inc), Filip Haltmayer (Milvus), Jo Kristian Bergum (Vespa), Kiichiro Yukawa (Vald) and Andre Zayarni (Qdrant)*

This blog is discussed on HN: <https://news.ycombinator.com/item?id=28727816>

Update: Vector Podcast [launched!](#)



# Smaller Vector DB players: 71% are Open Source

Company	Product	Cloud	Open Source: Y/N	Algorithms
Weaviate	Weaviate	Y	Y (Go)	custom HNSW
Pinecone	Pinecone	Y	N	FAISS + own
GSI	APU chip for Elasticsearch / Opensearch	N	N	Neural hashing / Hamming distance
Qdrant	Qdrant	N	Y (Rust)	HNSW (graph)
Yahoo!	Vespa	Y	Y (Java, C++)	HNSW (graph)
Ziliz	Milvus	N	Y (Go, C++, Python)	FAISS, HNSW
Yahoo!	Vald	N	Y (Go)	NGT



# Milvus

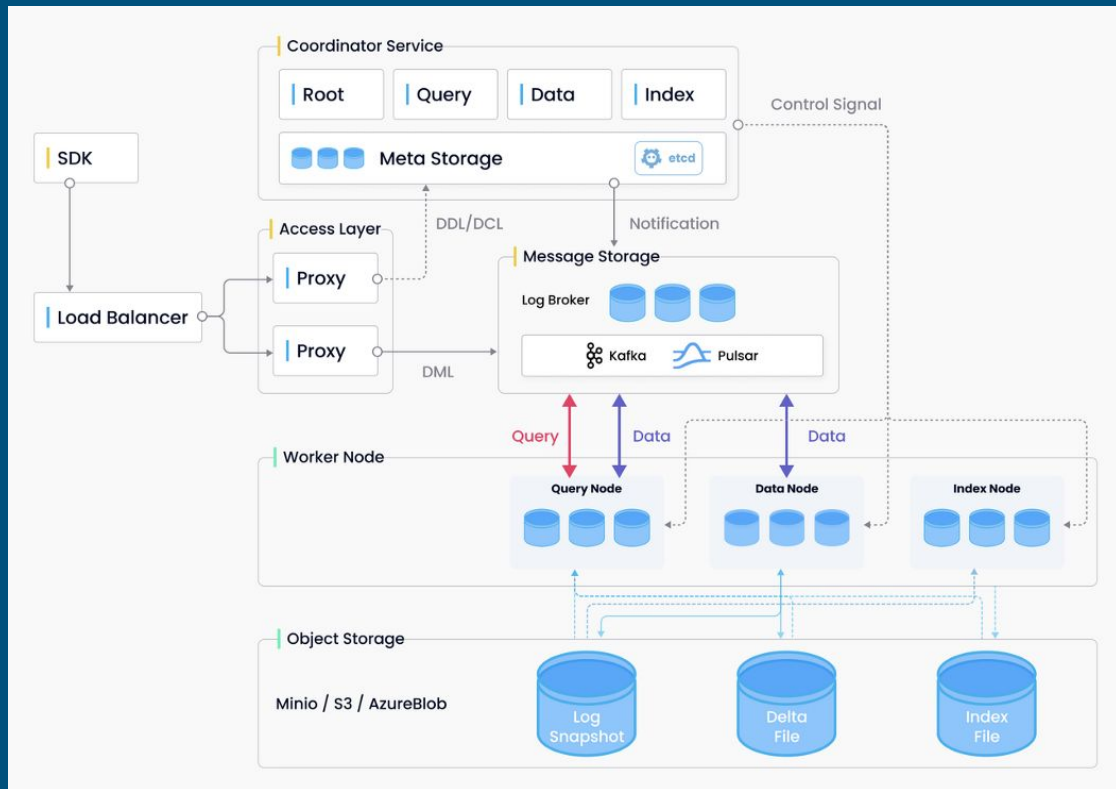
 [milvus.io](https://milvus.io)

 self-hosted vector database

 [open source](#)

Value proposition:

- attention to scalability:  
(re)indexing and search
- ability to index data with [multiple ANN algorithms](#) to compare their performance for your use case



# Pinecone

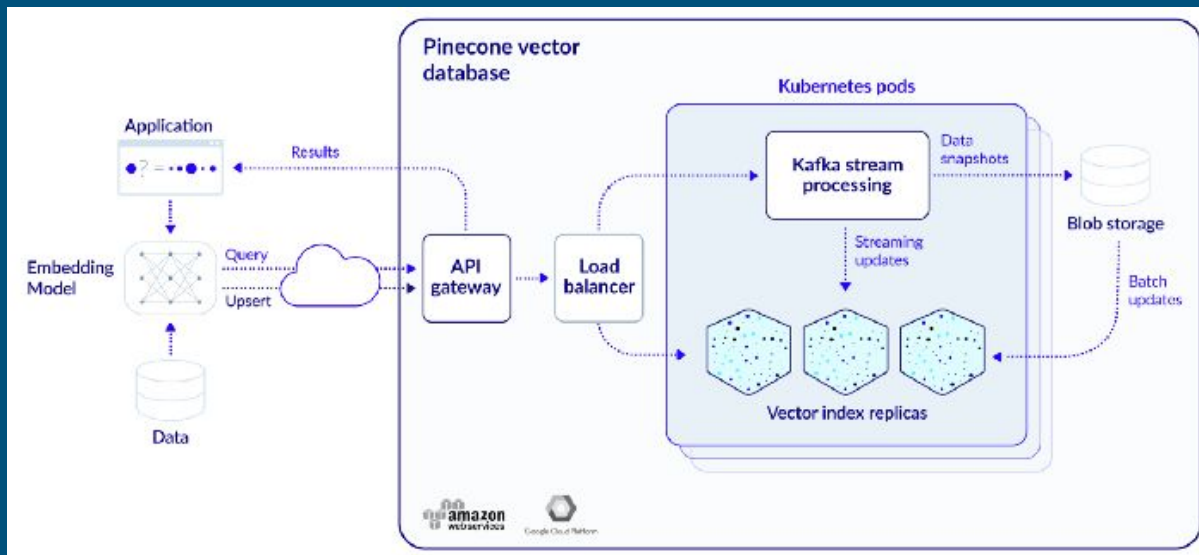
 [pinecone.io](https://pinecone.io)

 managed vector database

 close source

## Main features:

- Fully managed vector database
- Single-stage filtering capability: search for your objects (sweaters) + filter by metadata (color, size, price) in one query





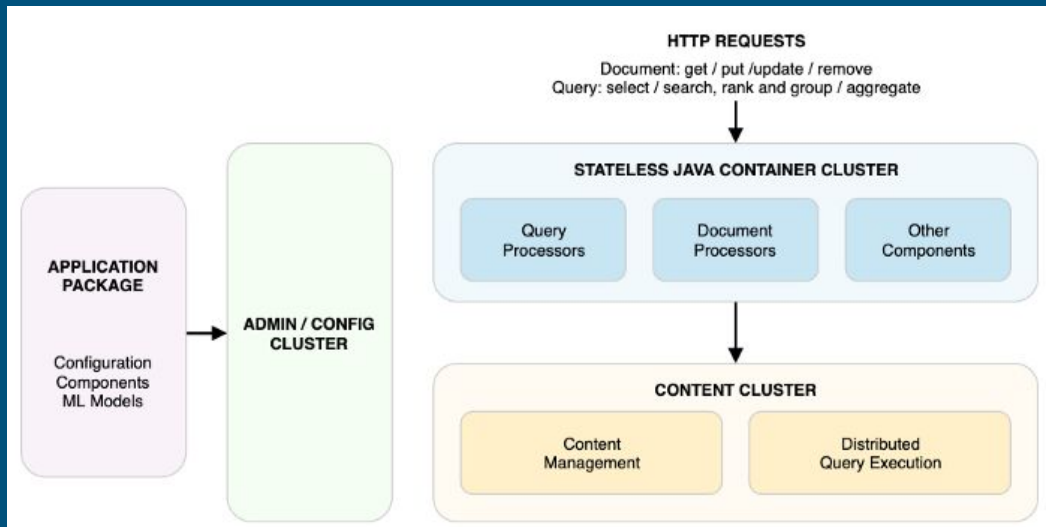
 [vespa.ai/](https://vespa.ai/)

 managed / self-hosted

 Code: [open source](#)

## Value proposition:

- low-latency computation over large data sets
- stores and indexes your data so that queries, selection and processing over the data can be performed at serving time
- customizable functionality
- deep data structures geared towards deep-learning like data science, like Tensors



# Weaviate



[semi.technology/developers/weaviate/current/](https://semi.technology/developers/weaviate/current/)



managed / self-hosted

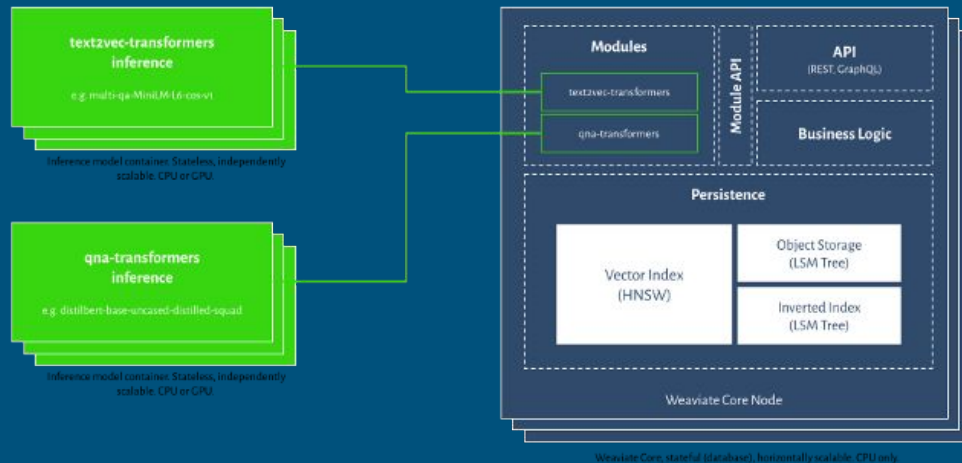


[open source](#)

Value proposition:

- Expressive query syntax
- [Graphql-like](#) interface
- combo of vector search, object storage and inverted index
- Wow-effect: Has an impressive [question answering component](#) — esp for demos

Weaviate System Level Overview (Example with two modules)



Two modules (text2vec-transformers, qna-transformers) shown as an example. Other modules include vectorization for other media types, entity recognition, spell checking and others.

Persistence in Weaviate Core shows one shard as an example. Users can create any number of indices, each index can contain any number of shards. Shards can be distributed and/or replicated across nodes in the cluster. A shard always contains object, inverted and vector storage. Vector storage is not affected by LSM organization.



# Vald

Link: [vald.vdaas.org/](https://vald.vdaas.org/)

Type: Self-hosted vector database

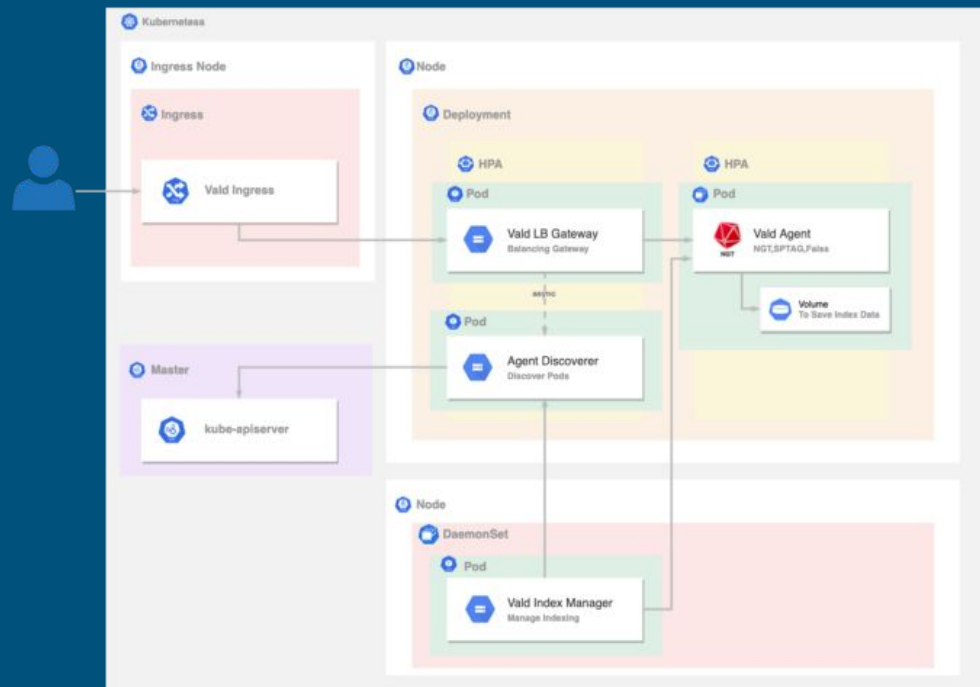
Code: [open source](#)

Value proposition:

- Billion-scale
- Cloud-native architecture
- Fastest ANN Algo: NGT
- Custom reranking / filtering algorithm plugins



## Vald Basic Architecture

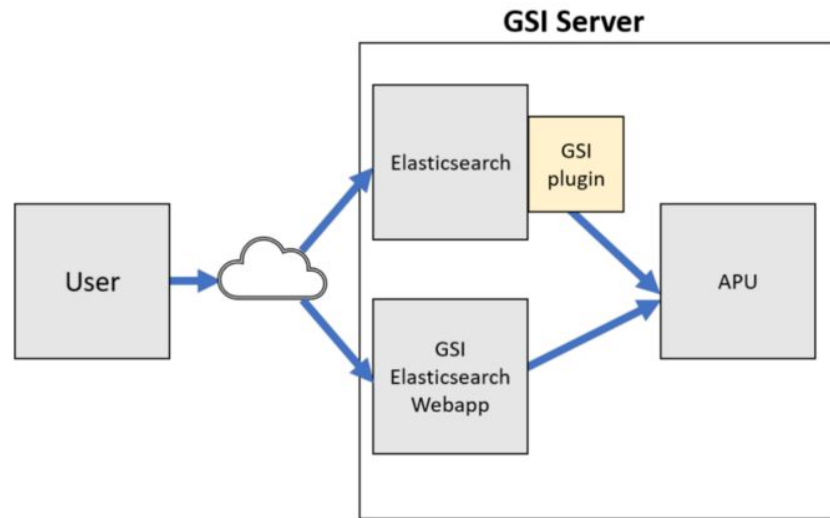


# GSI APU

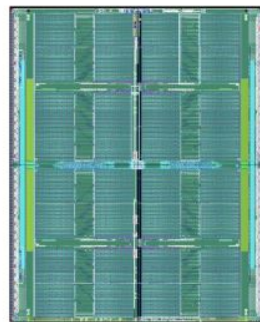
- 🌐 Link: [gsitechnology.com/APU](https://gsitechnology.com/APU)
- 💡 Type: Vector search hardware backend for your [Elasticsearch](#) / [OpenSearch](#)
- 🤖 Code: close source

## Value proposition:

- Billion-scale
- Extends your Elasticsearch / OpenSearch capabilities to similarity search
- On-prem / hosted APU board hosted cloud backend



## Gemini® APU Processor



- Internal Clock
  - 200 – 500 MHz
- Compute In Memory
  - 48 million 10T SRAM cells
  - 2 million units of prog "bit-logic"
- L1 Cache
  - 96Mb
- Algorithms
  - Similarity Search
  - Vector Processing
  - SAR BPA, Image Processing



# Qdrant

 [qdrant.tech/](https://qdrant.tech/)

 self-hosted vector database (Cloud in roadmap)

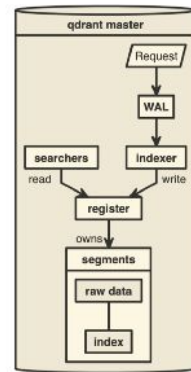
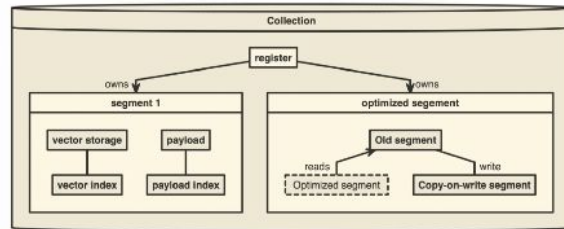
 [open source](#)

## Value proposition:

- The vector similarity engine with extended filtering support
- dynamic query planning and payload data indexing
- string matching, numerical ranges, geo-locations, and more
- [Metric Deep Learning](#)

## Qdrant Architecture

- Storage is split into **Segments**
- Segments can be re-built by the **optimizer**
- Segments are always available for search



# Outline

---

- Introduction to Multimodal LLMs
- Key use cases
- Specific applications
- The role of Vector Databases
- Emerging trends in LLMs



# Reasoning LLMs

---

- Self-replay: rStar (Microsoft Asia)
- Slower, but more accurate models, like QwQ-32B-Preview by team Qwen (Alibaba Research) - 32.5B parameters, 32000 prompt capacity:  
<https://techcrunch.com/2024/11/27/alibaba-releases-an-open-challenger-to-openais-o1-reasoning-model/>

# rStar: Self-play muTuAl Reasoning

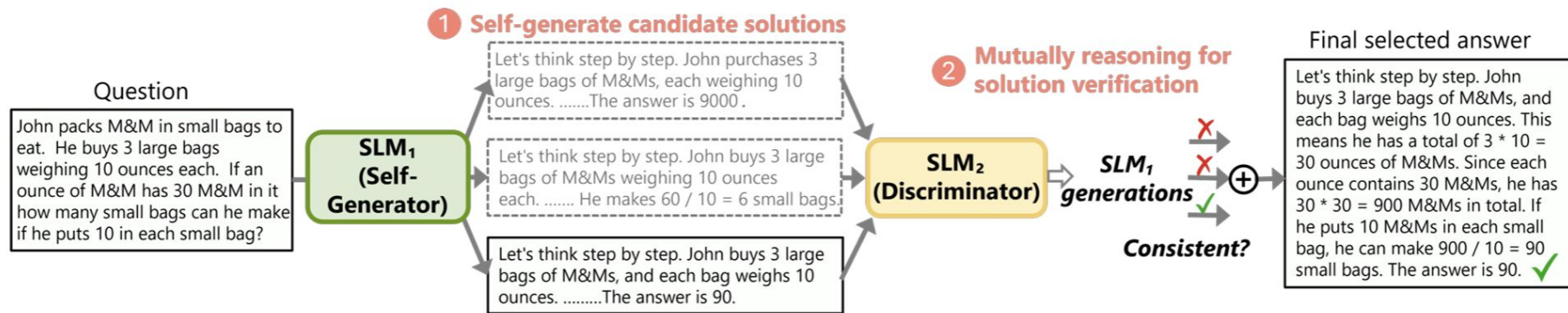


Figure 2: Our self-play mutual reasoning is a generation-discrimination process: (1) a self-generator augments the target SLM to generate candidate reasoning trajectories using MCTS; (2) the discriminator uses another SLM to provide unsupervised feedback on each trajectory based on partial hints; (3) based on this feedback, the target SLM decides a final reasoning trajectory as the solution.

# Lab!

---

1. Continue working on the RAG app, if you are still not done
2. Improve `query_tables.py` with:
  - a. Loading all sections to avoid hard-coding a section with tables
  - b. Test the capabilities of reasoning with table data: can it sum up numbers or do some other calculation?
3. Implement code in `synthetic_data.py`: generate variants of queries with misspellings and test them
4. Optional: set up, run and test:  
<https://github.com/DAMO-NLP-SG/LLM-Sentiment>

For 2 and 3 consult week-6/README.md

# Group session

---

Think about a field of life that deserves to be improved with LLMs.

- Can you come up with a solution to this?
- What are the expected benefits?
- What is the risk of model hallucination / inaccuracy / bias / misuse and what is the mitigation?
- Why does it matter? Broader impact on society or individuals

Examples of fields: Healthcare, Education, Transportation, Social Media, Customer Service

# Be prepared to present

---

- The field you chose
- The problem and your LLM-based solution
- A potential risk and how it could be mitigated

# Further study

---

- Ichigo paper: <https://arxiv.org/pdf/2410.15316>
- MSFT's GraphRAG: <https://microsoft.github.io/graphrag/>
- GraphRAG with Neo4j: demo  
<https://neo4j.com/labs/genai-ecosystem/rag-demo/>
- Book code: Building LLM applications:  
<https://github.com/PacktPublishing/Building-LLM-Powered-Applications/tree/main>
- rStar paper:
- GitHub Copilot:  
<https://github.blog/ai-and-ml/github-copilot/inside-github-working-with-the-lms-behind-github-copilot/>

# Further study

---

- ColPali demo: <https://huggingface.co/spaces/manu/ColPali-demo>
- Podcast episode with the Cursor team:  
<https://lexfridman.com/cursor-team/>