Use cases and applications of LLMs - Lecture 6

Dmitry Kan, PhD

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)

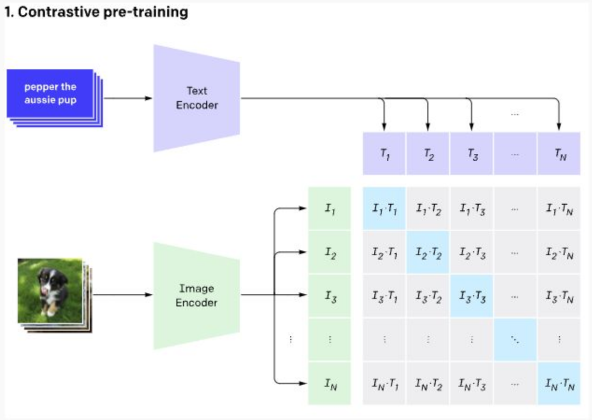
Key Architectural Elements

- **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: Radfortd 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 (decoder-based training)

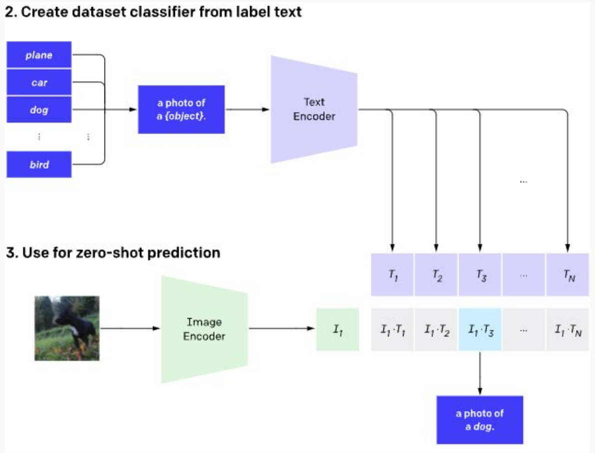
Pretraining objectives

1. **Contrastive Learning**: align embeddings of related data from different 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 question 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

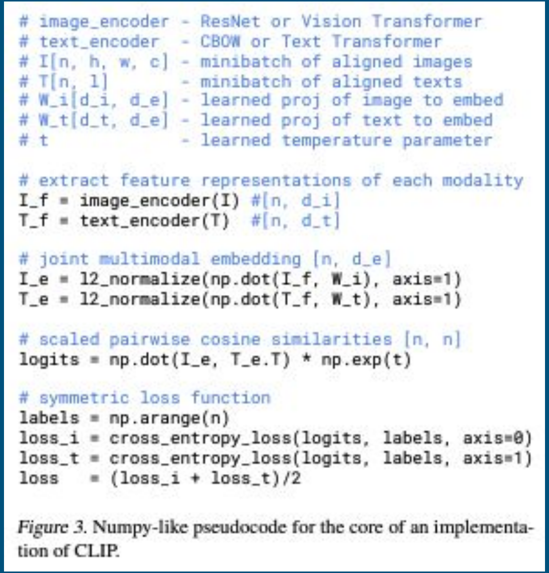

1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ... I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

Overview A


2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
3. Use for zero-shot prediction
Image Encoder
T_1 T_2 T_3 ... T_N
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

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, l]       - 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
```

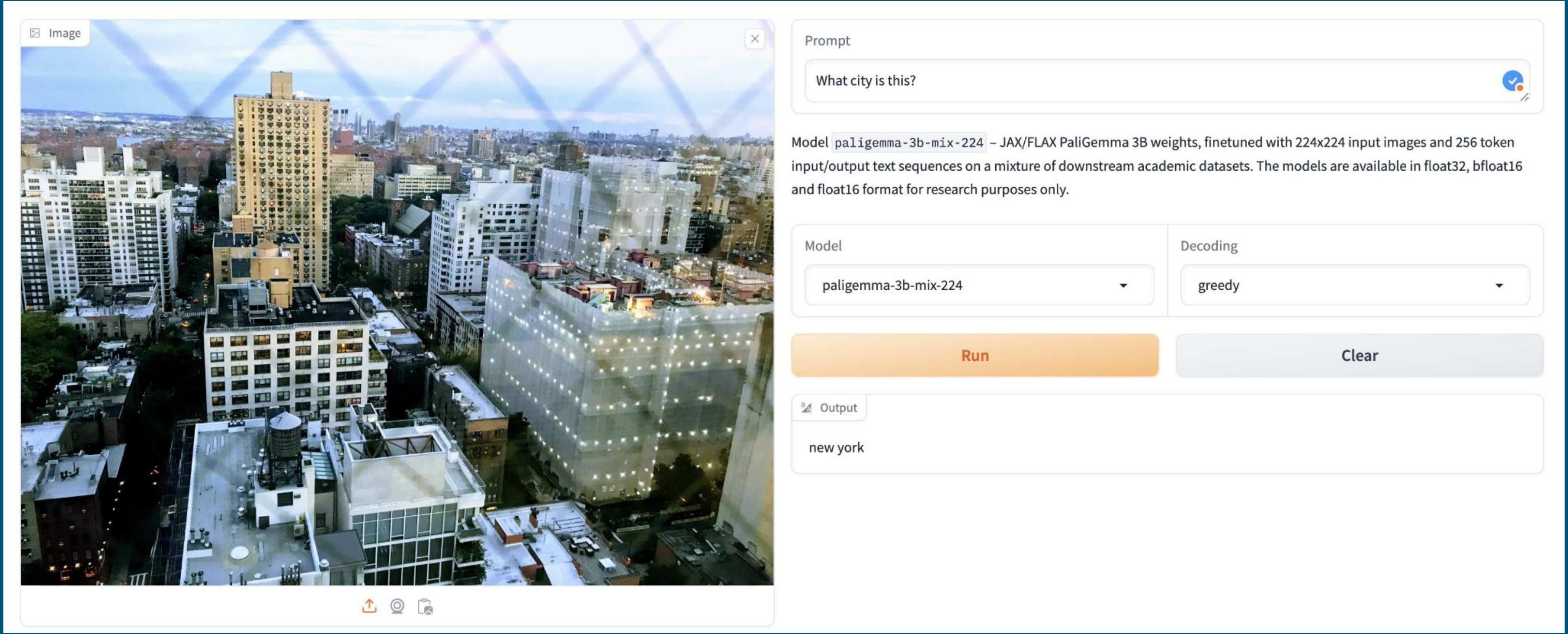
Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

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 (tested 2024, demo paused)



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

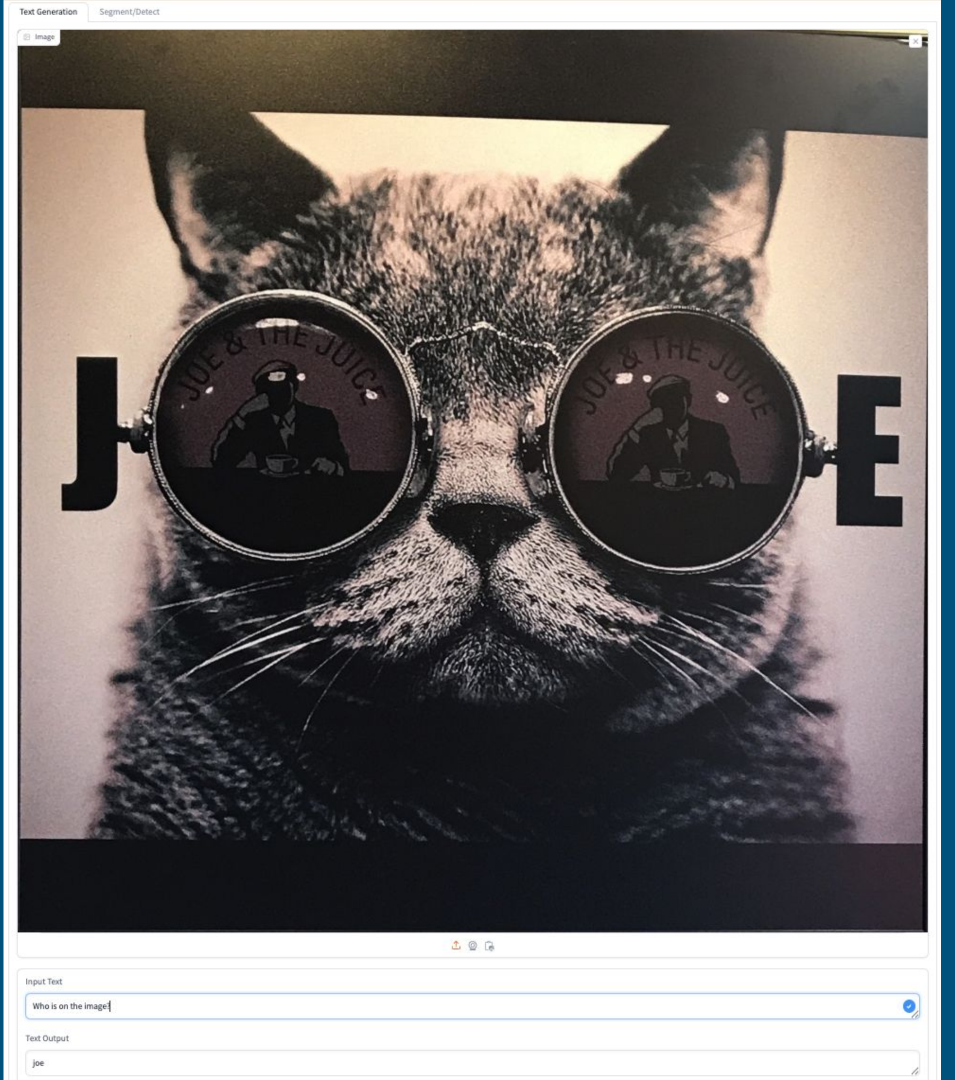
Decoding: greedy

Output

new york

PaliGemma-hf (2025)

https://huggingface.co/spaces/big-vision/paligemma-hf



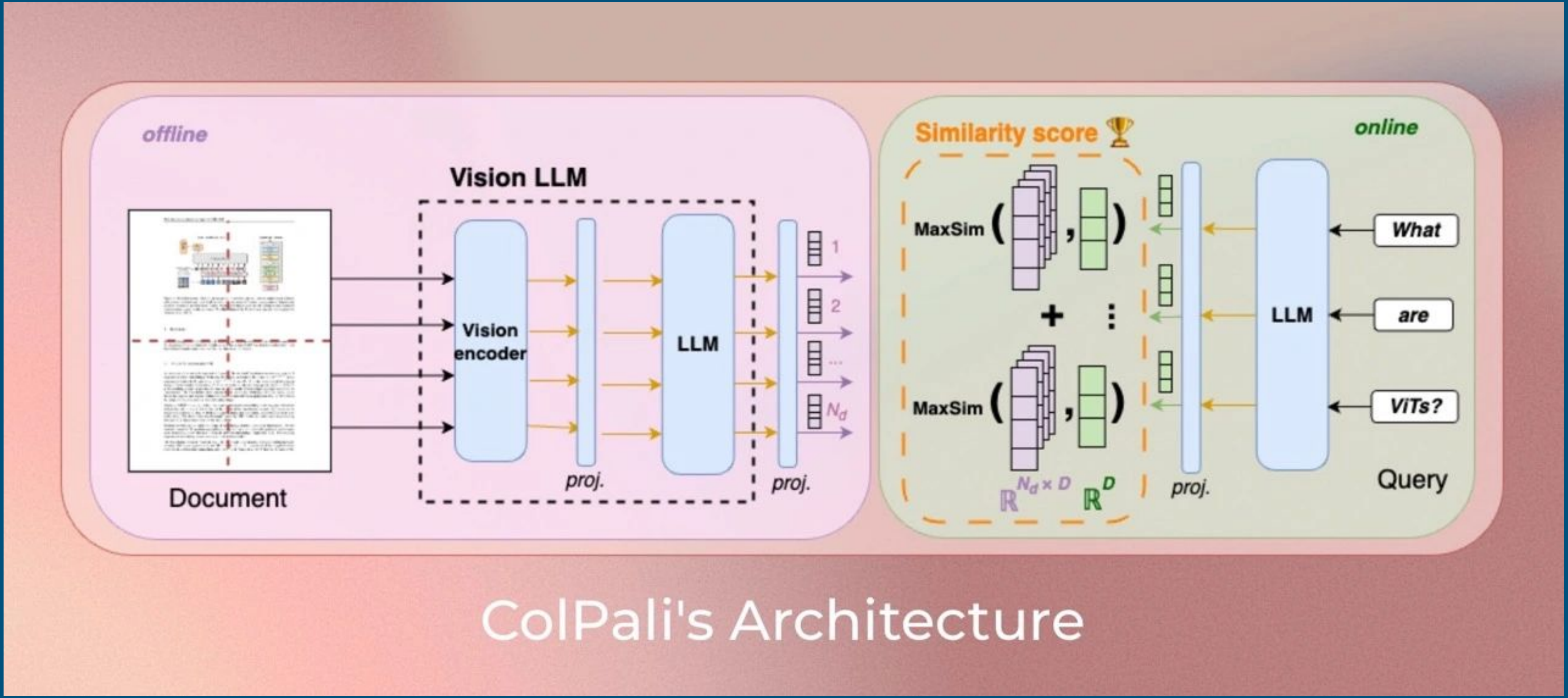
Input Text

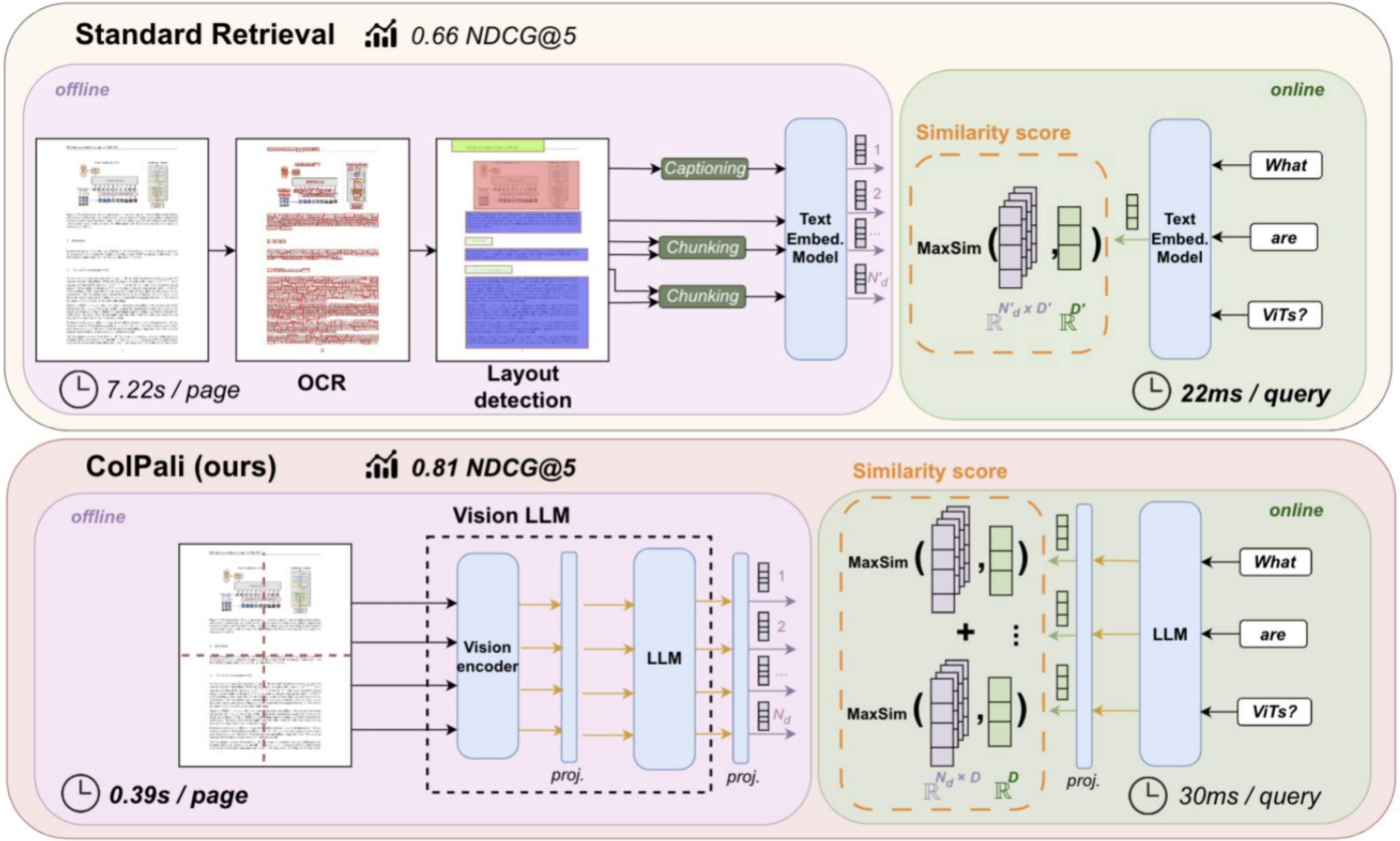
Who is on the image?

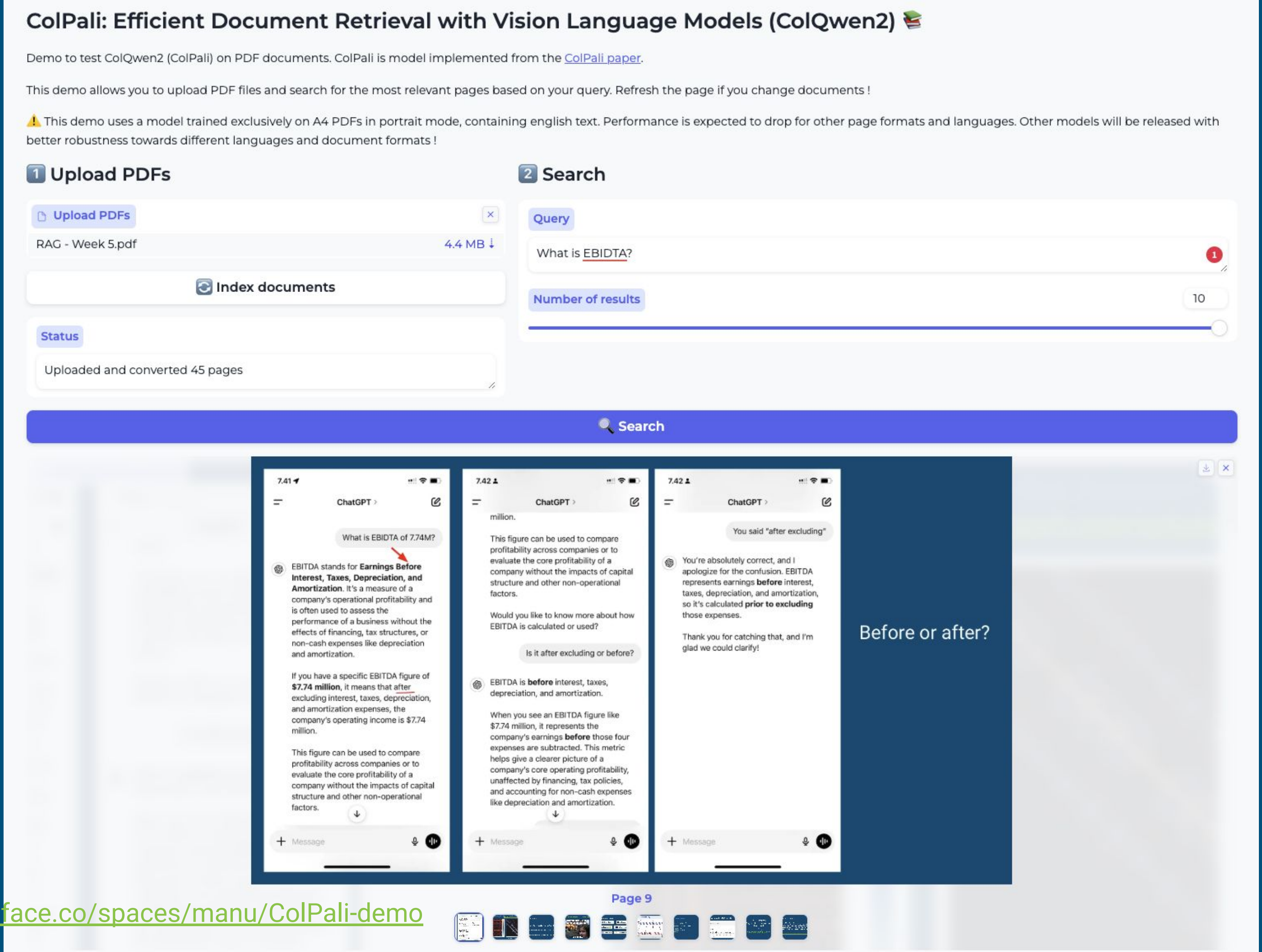
Text Output

joe

ColPali


offline
Vision LLM
Vision encoder
proj.
LLM
proj.
1
2
...
N_d
Document
Similarity score
online
MaxSim
MaxSim
+
⋮
$\mathbb{R}^{N_d \times D}$
$\mathbb{R}^{D}$
proj.
LLM
What
are
ViTs?
Query
ColPali's Architecture


Standard Retrieval
0.66 NDCG@5
offline
7.22s / page
OCR
Layout detection
Captioning
Chunking
Chunking
Text Embed. Model
1
2
...
N'_d
online
Similarity score
MaxSim
$\mathbb{R}^{N'_d \times D'}$
$\mathbb{R}^{D'}$
Text Embed. Model
What
are
ViTs?
22ms / query
ColPali (ours)
0.81 NDCG@5
offline
Vision LLM
Vision encoder
proj.
LLM
proj.
1
2
...
N_d
0.39s / page
Similarity score
MaxSim
+
MaxSim
$\mathbb{R}^{N_d \times D}$
$\mathbb{R}^{D}$
proj.
online
LLM
What
are
ViTs?
30ms / query



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 !

1️⃣ Upload PDFs

Upload PDFs

RAG - Week 5.pdf 4.4 MB

Index documents

Status

Uploaded and converted 45 pages

2️⃣ Search

Query

What is EBIDTA?

Number of results 10

🔍 Search

7:41

ChatGPT

What is EBIDTA of 7.74M?

EBITDA stands for **Earnings Before Interest, Taxes, Depreciation, and Amortization**. It's a measure of a company's operational profitability and is often used to assess the performance of a business without the effects of financing, tax structures, or non-cash expenses like depreciation and amortization.

If you have a specific EBITDA figure of **$7.74 million**, it means that after excluding interest, taxes, depreciation, and amortization expenses, the company's operating income is $7.74 million.

This figure can be used to compare profitability across companies or to evaluate the core profitability of a company without the impacts of capital structure and other non-operational factors.

Message

7:42

ChatGPT

million.

This figure can be used to compare profitability across companies or to evaluate the core profitability of a company without the impacts of capital structure and other non-operational factors.

Would you like to know more about how EBITDA is calculated or used?

Is it after excluding or before?

EBITDA is **before** interest, taxes, depreciation, and amortization.

When you see an EBITDA figure like $7.74 million, it represents the company's earnings **before** those four expenses are subtracted. This metric helps give a clearer picture of a company's core operating profitability, unaffected by financing, tax policies, and accounting for non-cash expenses like depreciation and amortization.

Message

7:42

ChatGPT

You said "after excluding"

You're absolutely correct, and I apologize for the confusion. EBITDA represents earnings **before** interest, taxes, depreciation, and amortization, so it's calculated **prior to excluding** those expenses.

Thank you for catching that, and I'm glad we could clarify!

Message

Before or after?

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. It consists of several components:

1. Pre-trained Image Encoder: Extracts features from individual video frames.
2. Position Embedding Layer: Injects temporal information into the video frames.
3. Video Q-Former: Aggregates frame-level representations and generates visual query tokens.
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.

https://arxiv.org/pdf/2306.02858

October, 2023 - Alibaba Research

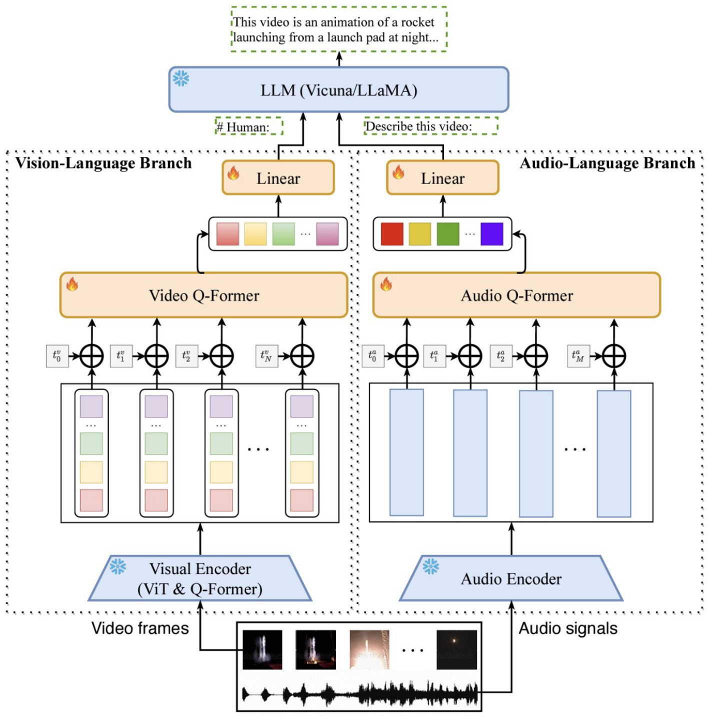
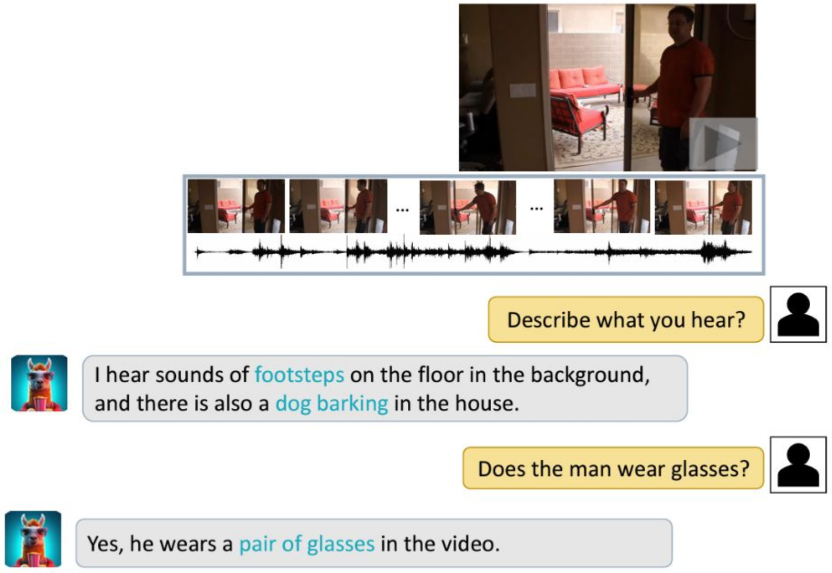


Figure 1: Overall architecture of Video-LLaMA.

Video-LLaMA: examples



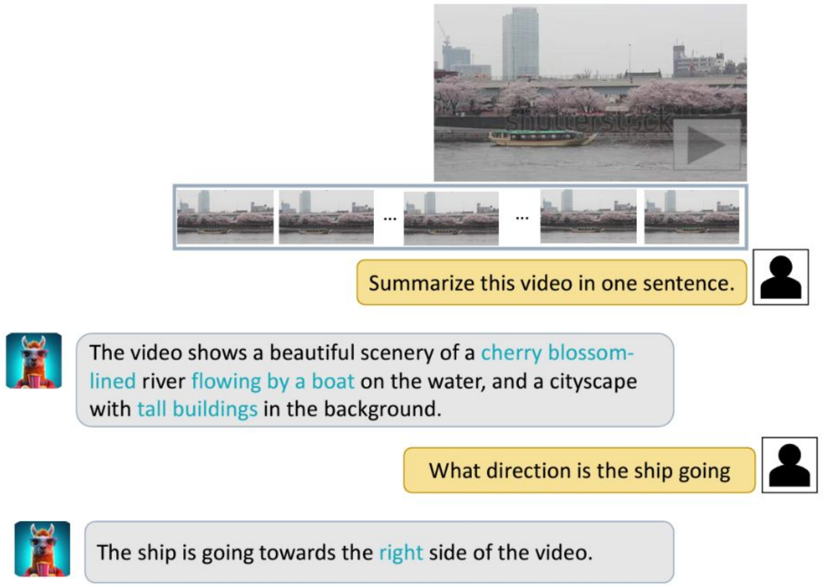
Describe what you hear?

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

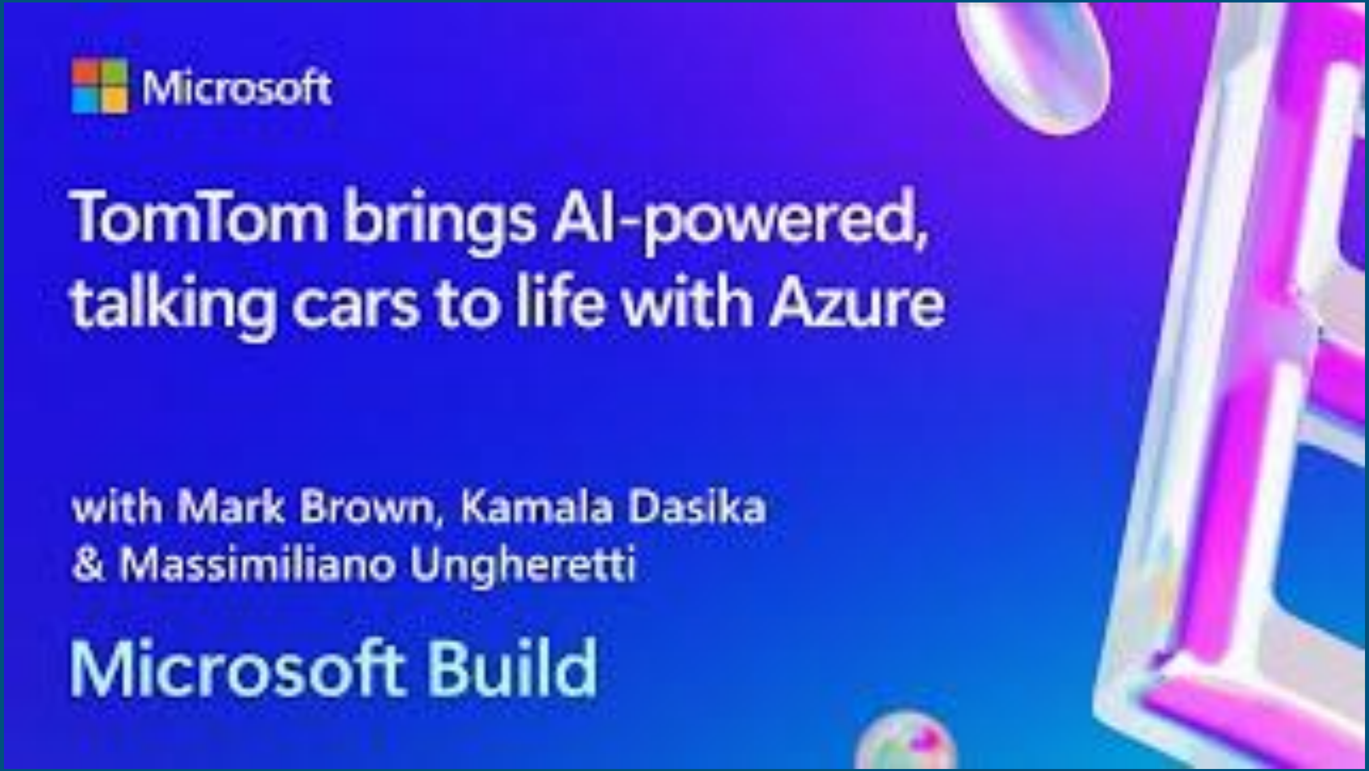
Use cases

- 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


Microsoft

TomTom brings AI-powered, talking cars to life with Azure

with Mark Brown, Kamala Dasika & Massimiliano Ungheretti

Microsoft Build

Ingredients for a conversation with your car

Brain:
GPT (LLM)

Ears: Automatic Speech Recognition (ASR)

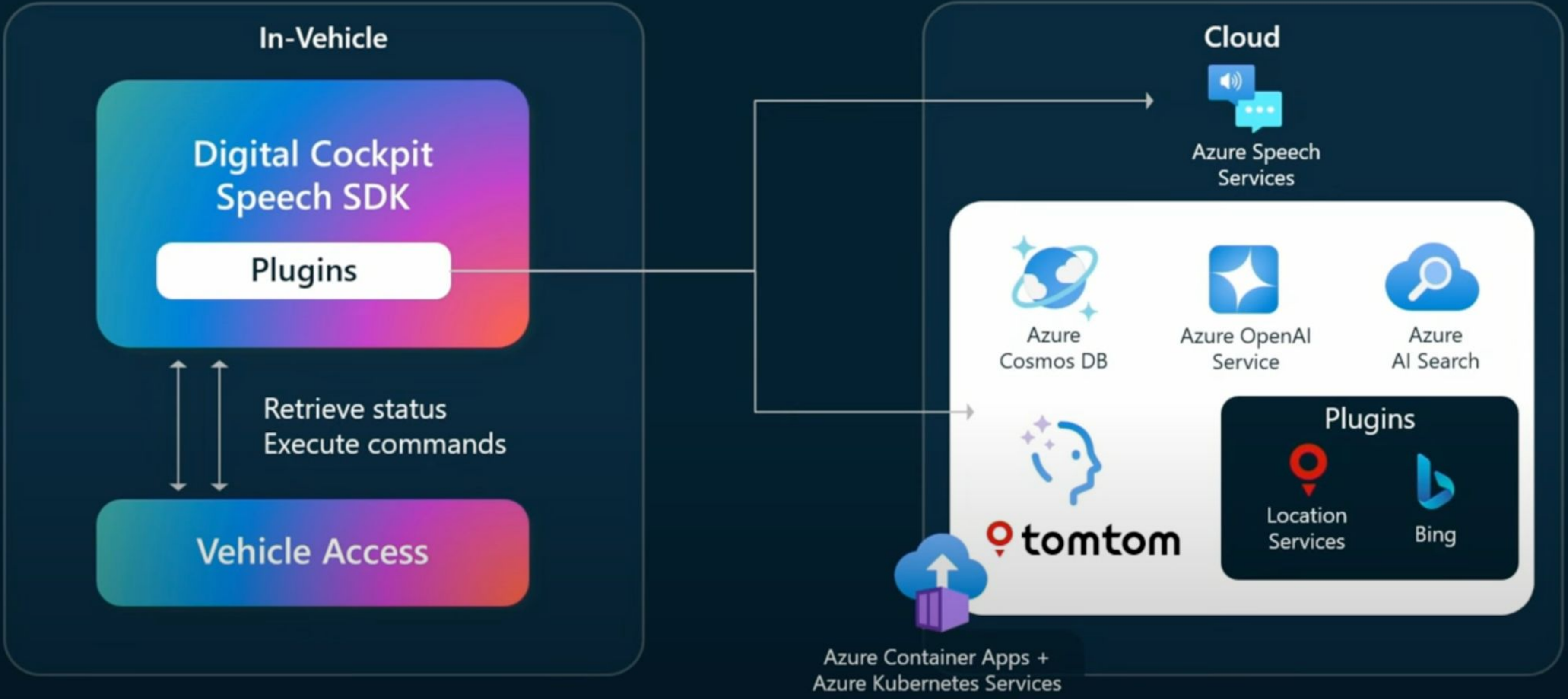
Mouth: Text-to-speech (TTS)

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

Short term memory: Conversation state

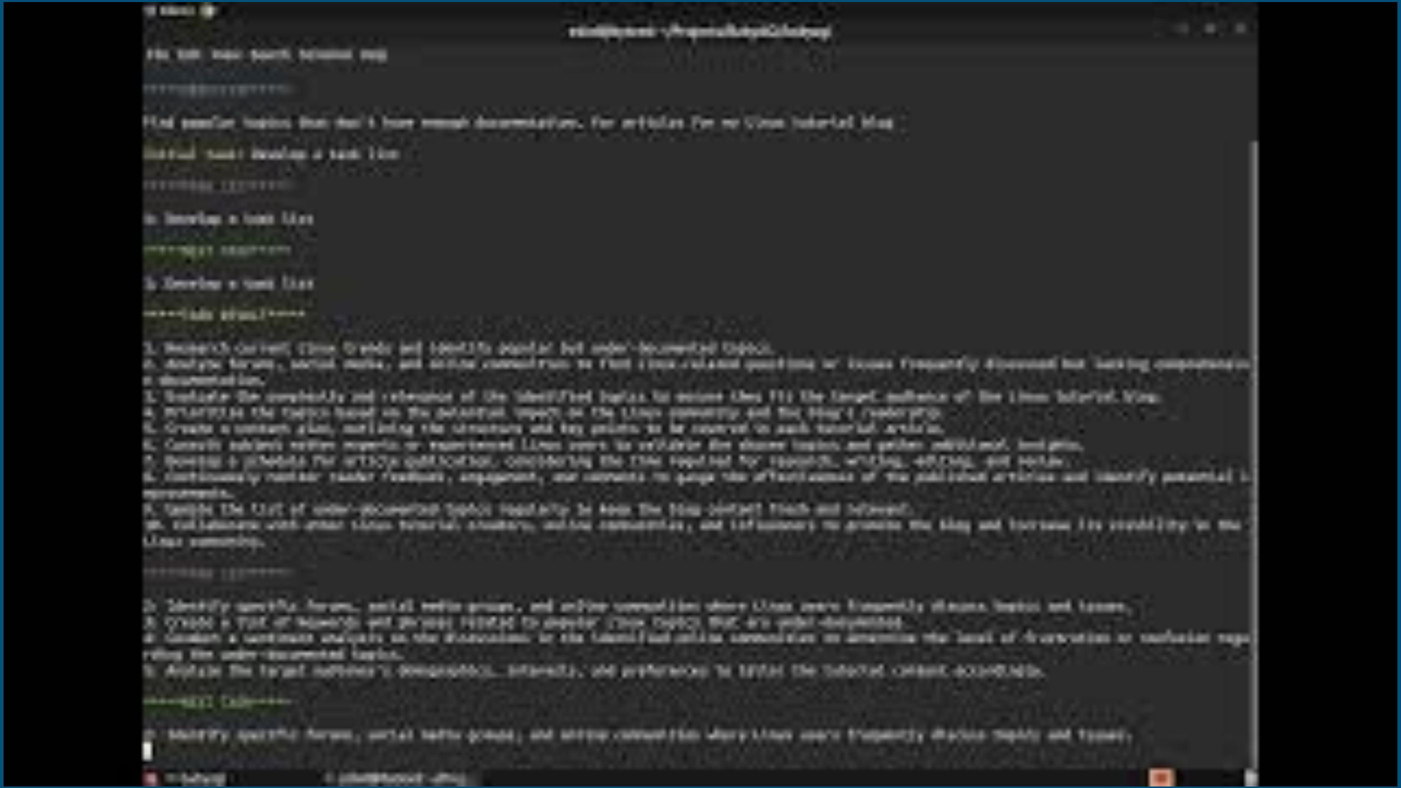
Knowledge: Semantic/Vector search & RAG

Architecture



BabyAGI

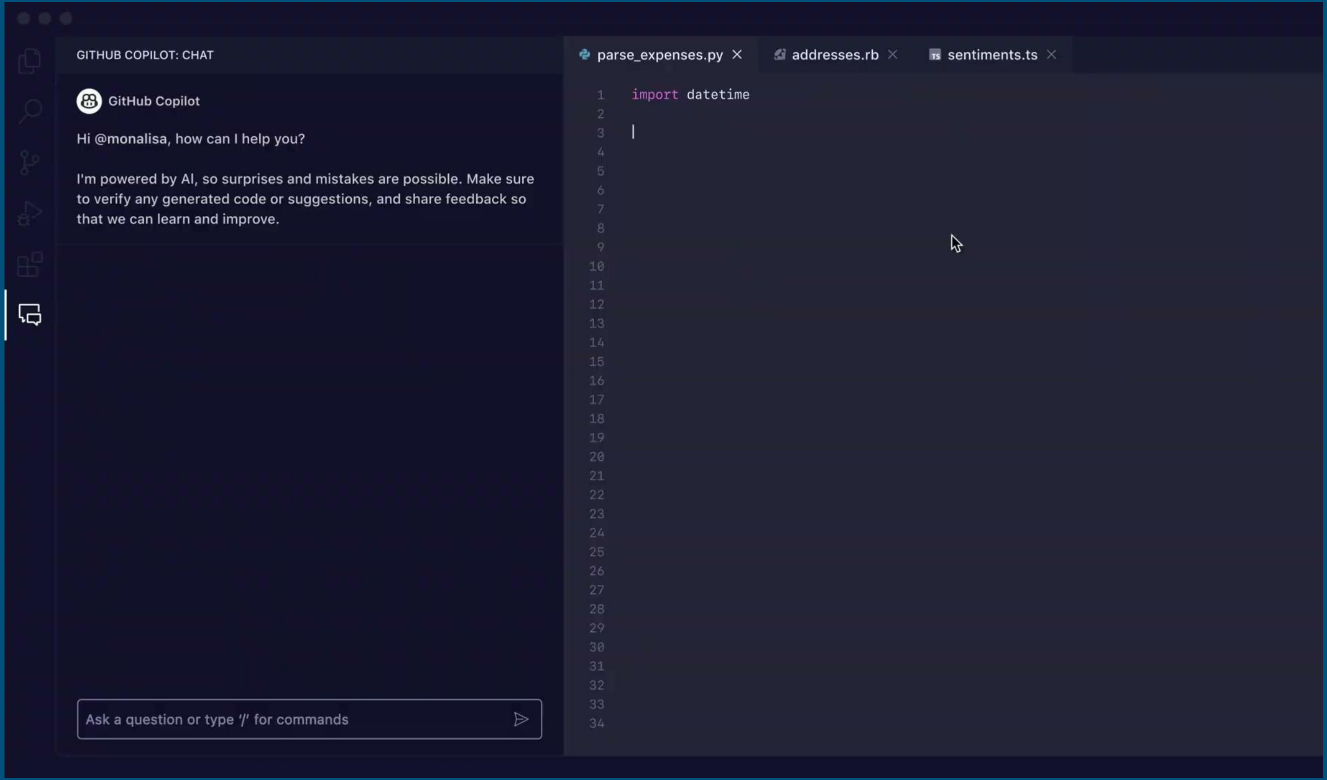
https://github.com/yoheinakajima/babyagi by Yohei Nakajima, Venture Capitalist at Untapped Capital



```
*****OBJECTIVE*****

Find popular topics that don't have enough documentation, for articles for my Linux tutorial blog

Initial task: Develop a task list

*****TASK LIST*****

1: Develop a task list

*****NEXT TASK*****

1: Develop a task list

*****TASK RESULT*****

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 complexity and relevance of the identified topics to ensure they fit the target audience of the Linux tutorial blog.
4. Prioritize the topics based on the potential impact on the Linux community and the blog's readership.
5. Create a content plan, outlining the structure and key points to be covered in each tutorial article.
6. Consult subject matter experts or experienced Linux users to validate the chosen topics and gather additional insights.
7. Develop a schedule for article publication, 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 i
mprovements.
9. Update the list of under-documented topics regularly to keep the blog content fresh and relevant.
10. Collaborate with other Linux tutorial creators, online communities, and influencers to promote the blog and increase its visibility in the
Linux community.

*****TASK LIST*****

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 popular Linux topics that are under-documented.
4: Conduct a sentiment analysis on the discussions in the identified online communities to determine the level of frustration or confusion rega
rding the under-documented topics.
5: Analyze the target audience's demographics, interests, and preferences to tailor the tutorial content accordingly.

*****NEXT TASK*****

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


GITHUB COPILOT: CHAT

GitHub Copilot

Hi @monalisa, how can I help you?

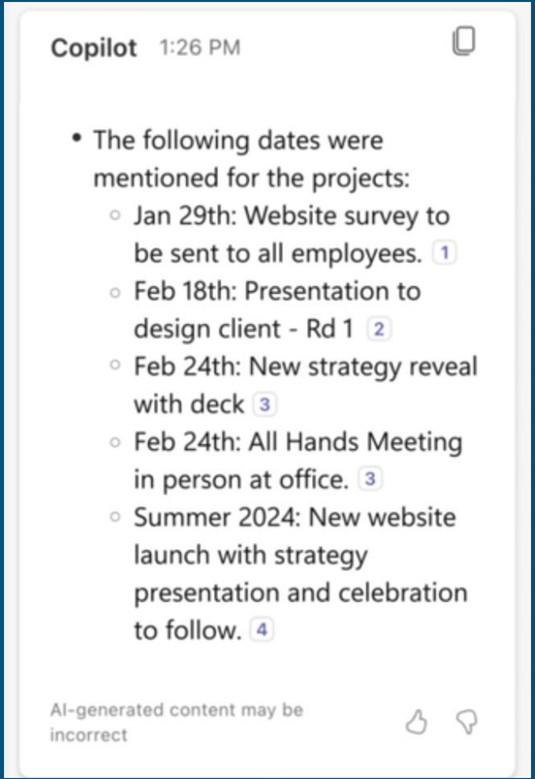
I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

Ask a question or type '/' for commands

parse_expenses.py | addresses.rb | sentiments.ts

```python
import datetime
```

Copilot for work (beyond coding)



Copilot 1:26 PM

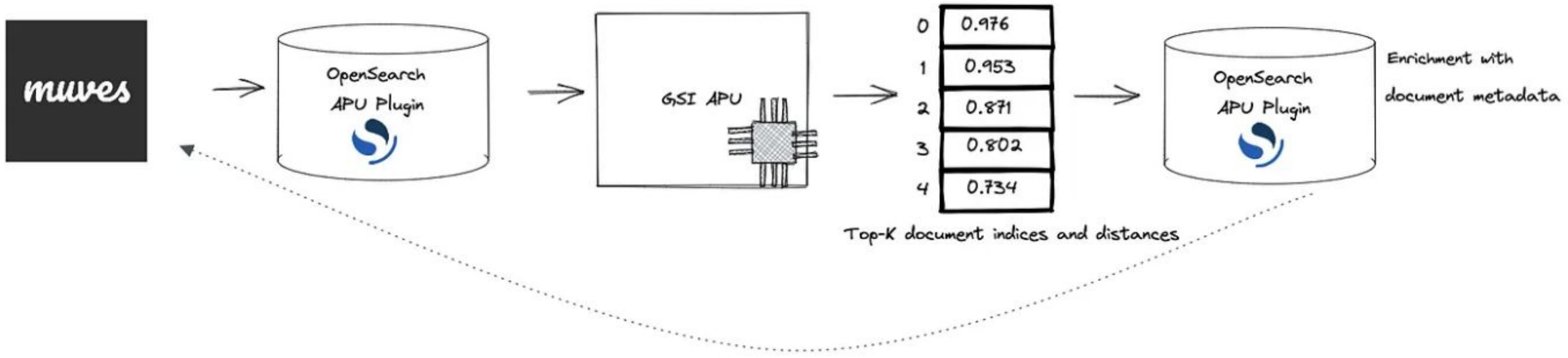
- The following dates were mentioned for the projects:
  - Jan 29th: Website survey to be sent to all employees. [1]
  - Feb 18th: Presentation to design client - Rd 1 [2]
  - Feb 24th: New strategy reveal with deck [3]
  - Feb 24th: All Hands Meeting in person at office. [3]
  - Summer 2024: New website launch with strategy presentation and celebration to follow. [4]

AI-generated content may be incorrect

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

https://copilot.cloud.microsoft/en-US/prompts?ocid=copilot_akams_copilotlab

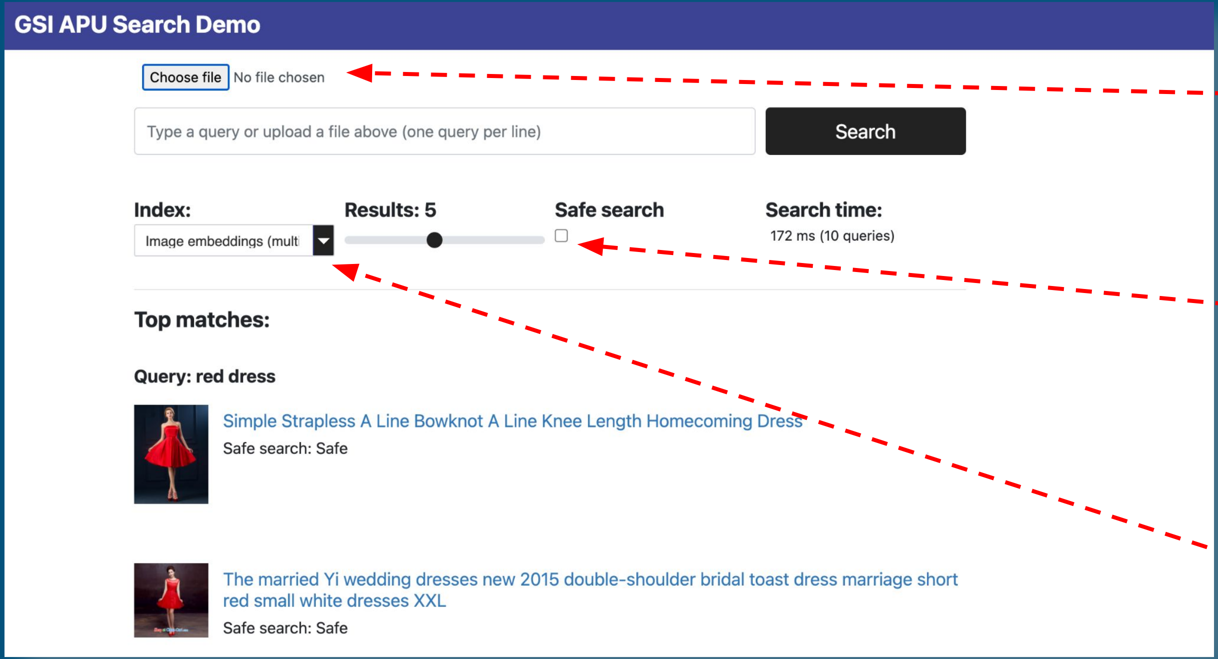
Better e-commerce with multilingual and multimodal vector search with HW acceleration



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 (multi

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

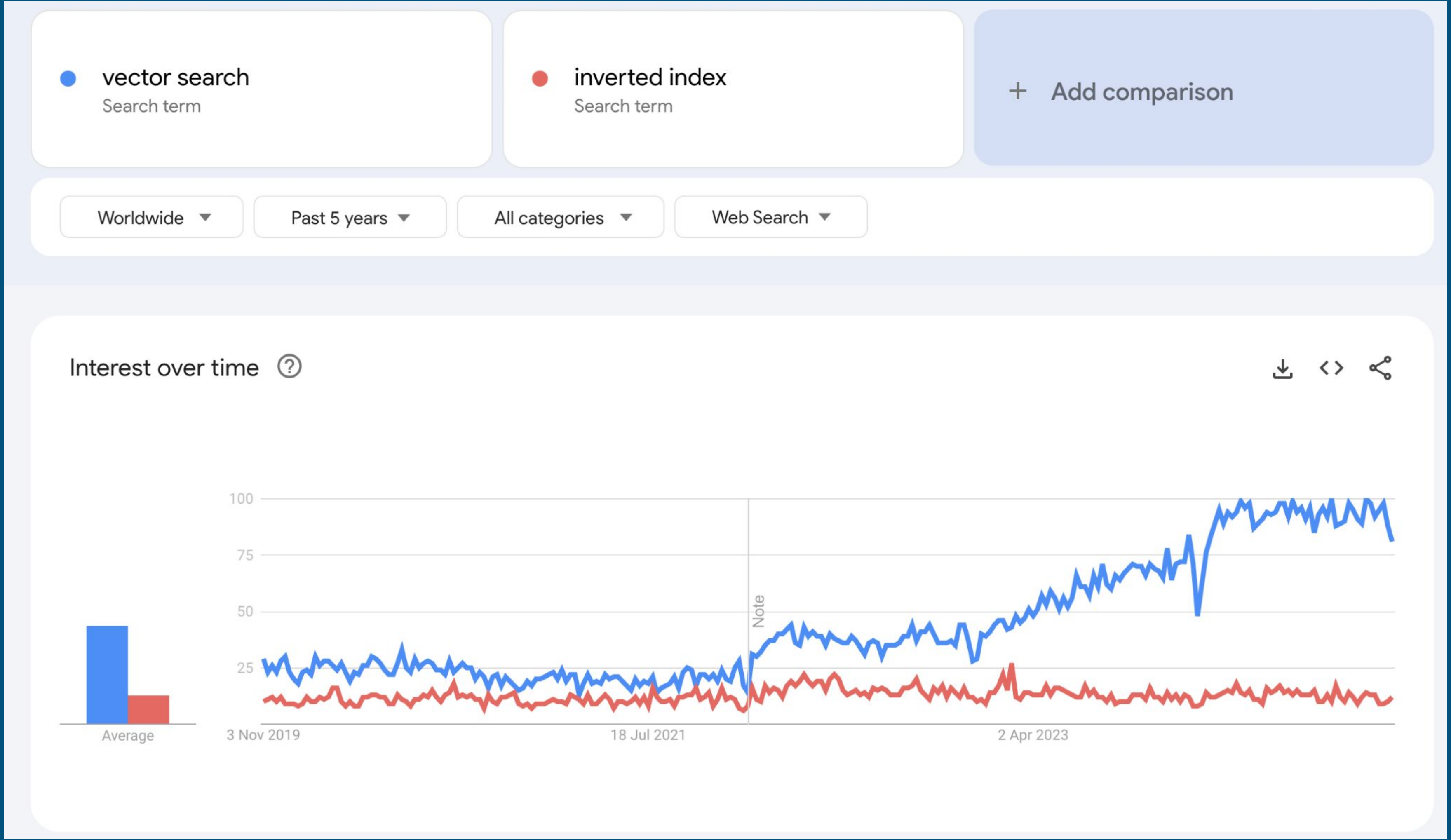
You can submit a text file with multiple queries (one query per line)

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

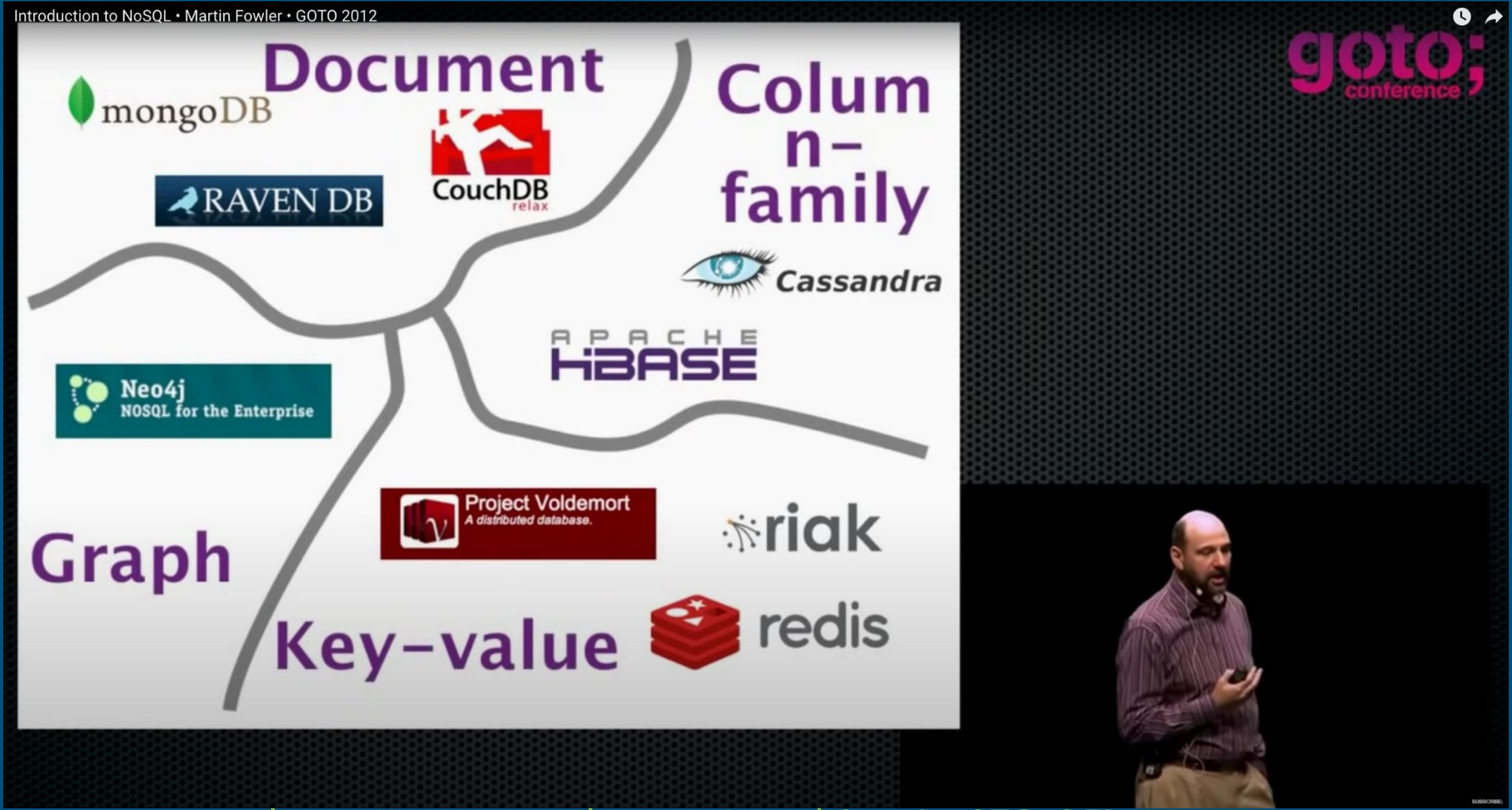
● inverted index
Search term

+ Add comparison

Worldwide ▾ | Past 5 years ▾ | All categories ▾ | Web Search ▾

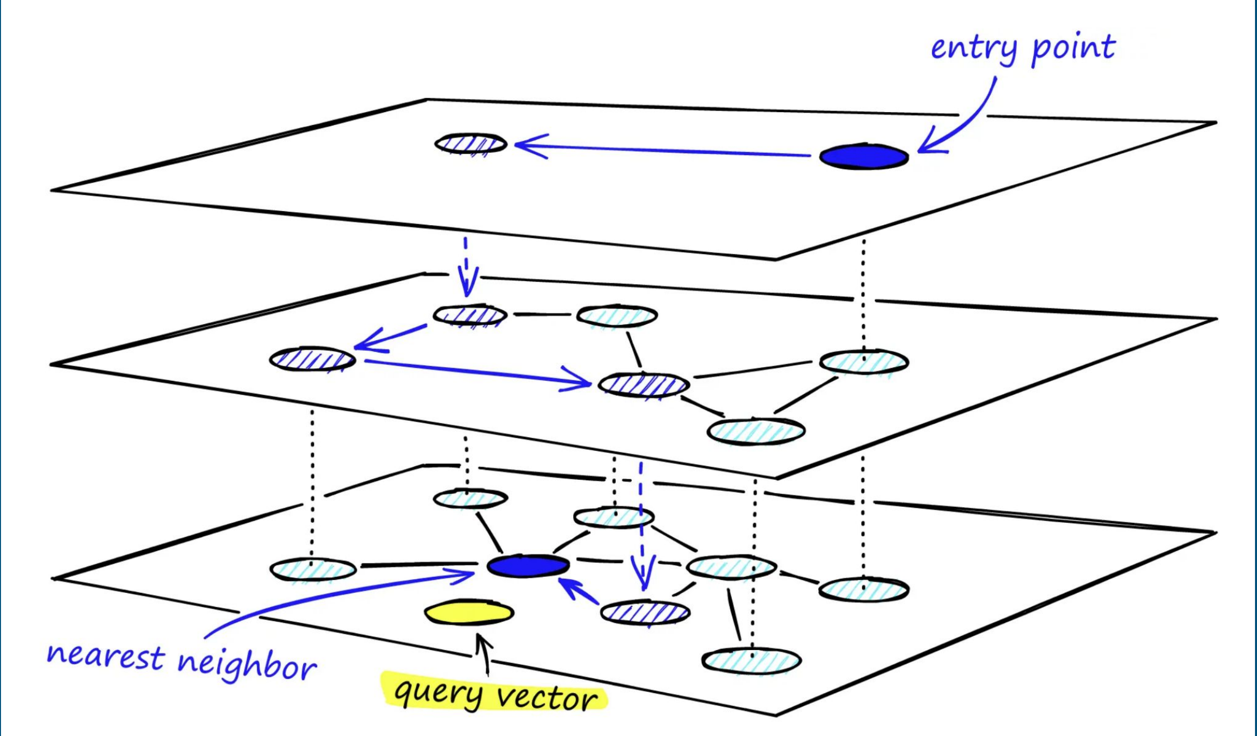
Interest over time

https://trends.google.com/trends/explore?date=today%205-y&q=vector%20search,inverted%20index&hl=en-GB


Introduction to NoSQL • Martin Fowler • GOTO 2012
Document
mongoDB
CouchDB relax
RAVEN DB
Column-family
Cassandra
APACHE HBASE
Neo4j NOSQL for the Enterprise
Graph
Project Voldemort A distributed database.
riak
redis
Key-value
goto; conference

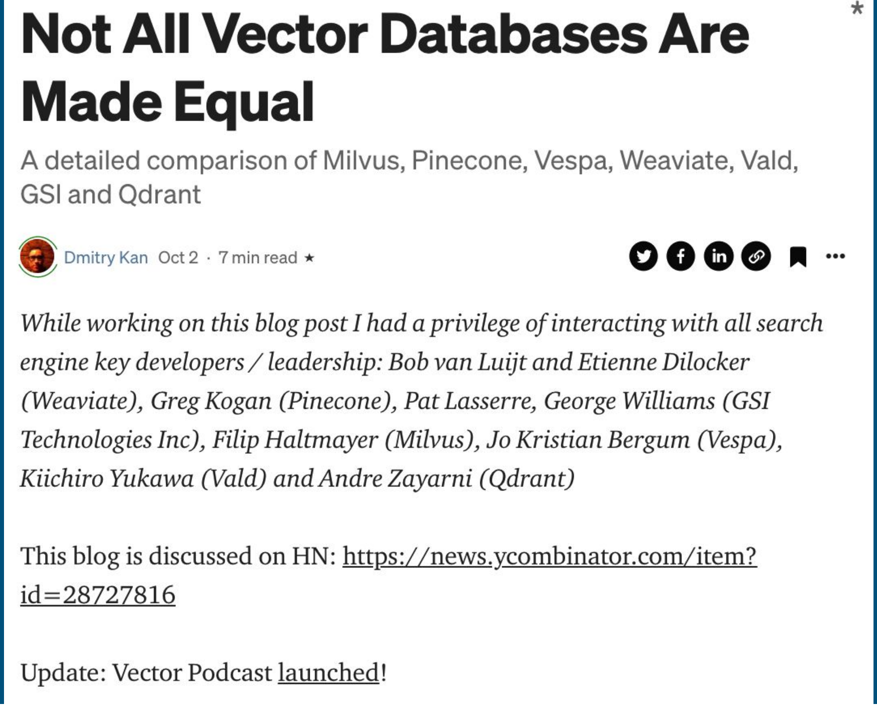
https://www.youtube.com/watch?v=qI_g07C_Q5I

Vector search algorithms: HNSW



The search process through the multi-layer structure of an HNSW graph.

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



Not All Vector Databases Are Made Equal

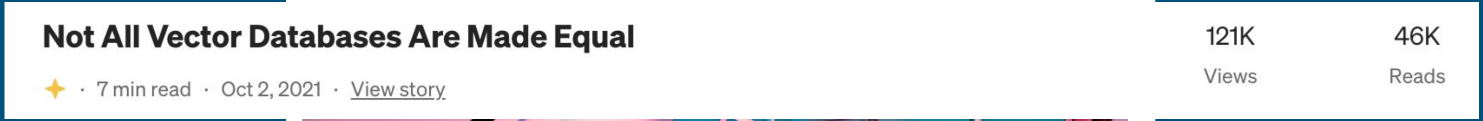
A detailed comparison of Milvus, Pinecone, Vespa, Weaviate, Vald, GSI and Qdrant

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 Dilocker (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!



Not All Vector Databases Are Made Equal

· 7 min read · Oct 2, 2021 · View story

121K	46K
Views	Reads

https://dmitry-kan.medium.com/milvus-pinecone-vespa-weaviate-vald-gsi-what-unites-these-buzz-words-and-what-makes-each-9c65a3bd0696

Smaller Vector DB players: 71% are Open Source

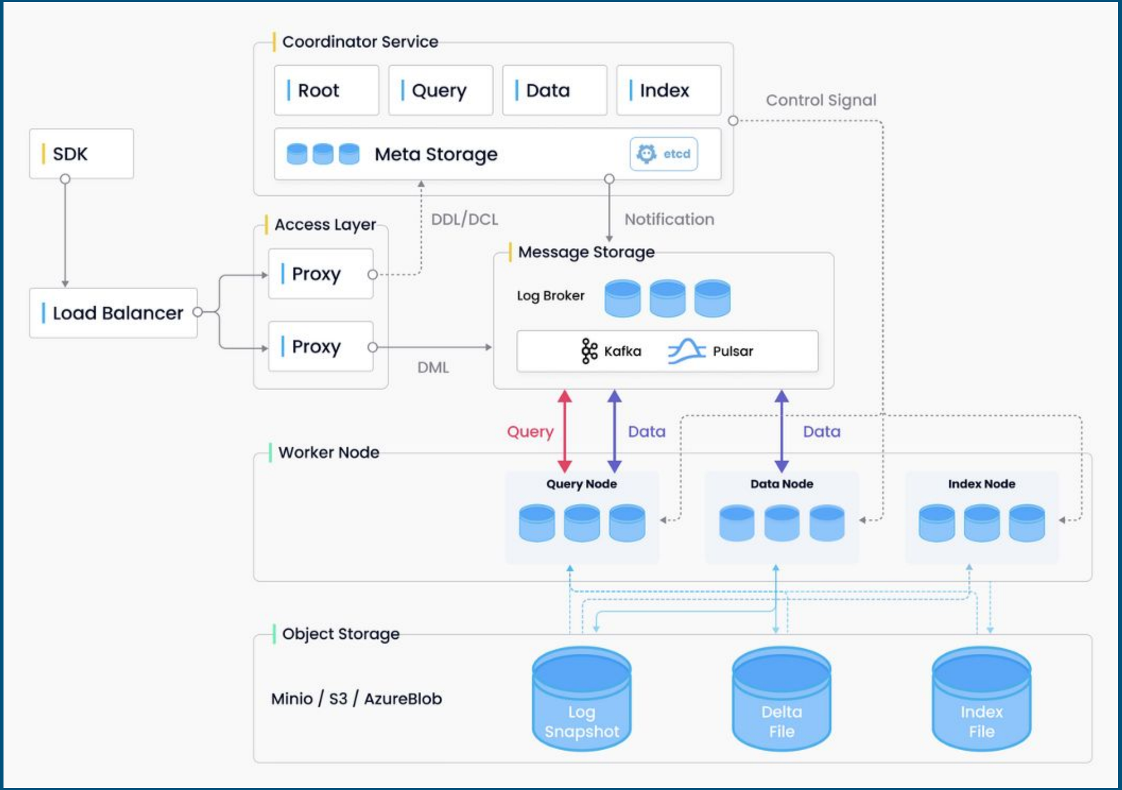
Company	Product	Cloud	Open Source: Y/N	Algorithms
Weaviate	Weaviate	Y	Y (Go)	custom HNSW
Pinecone	Pinecone	Y	N (Rust)	FAISS + own
GSI	APU chip for Elasticsearch / Opensearch	N	N	Neural hashing / Hamming distance
Qdrant	Qdrant	Y	Y (Rust)	HNSW (graph)
Yahoo!	Vespa	Y	Y (Java, C++)	HNSW (graph)
Ziliz	Milvus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

Milvus

🌍 milvus.io
💡 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



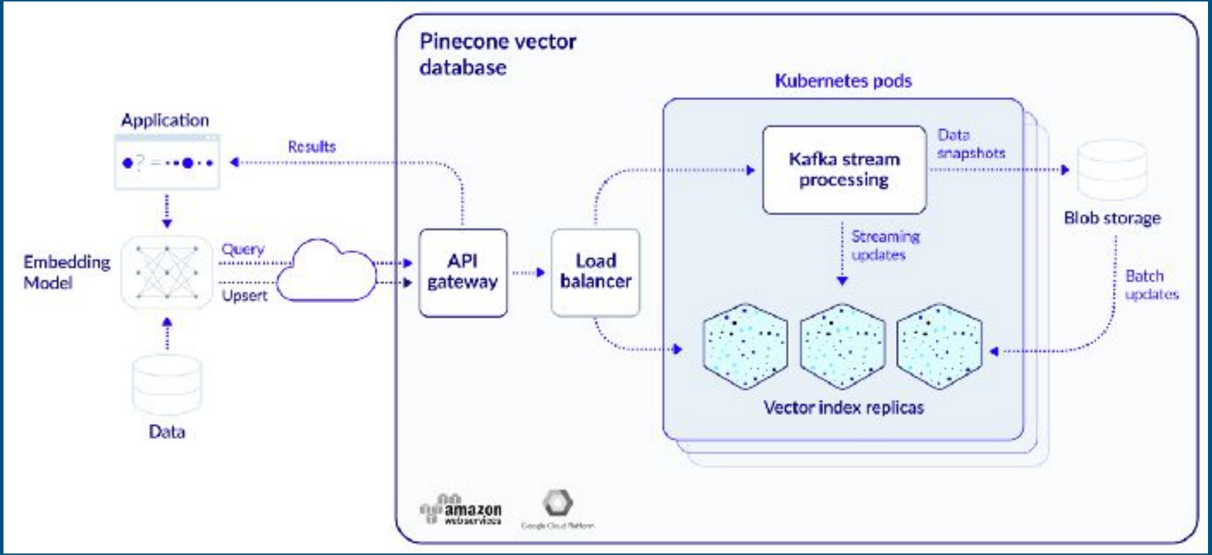
milvus

Pinecone

🌍 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



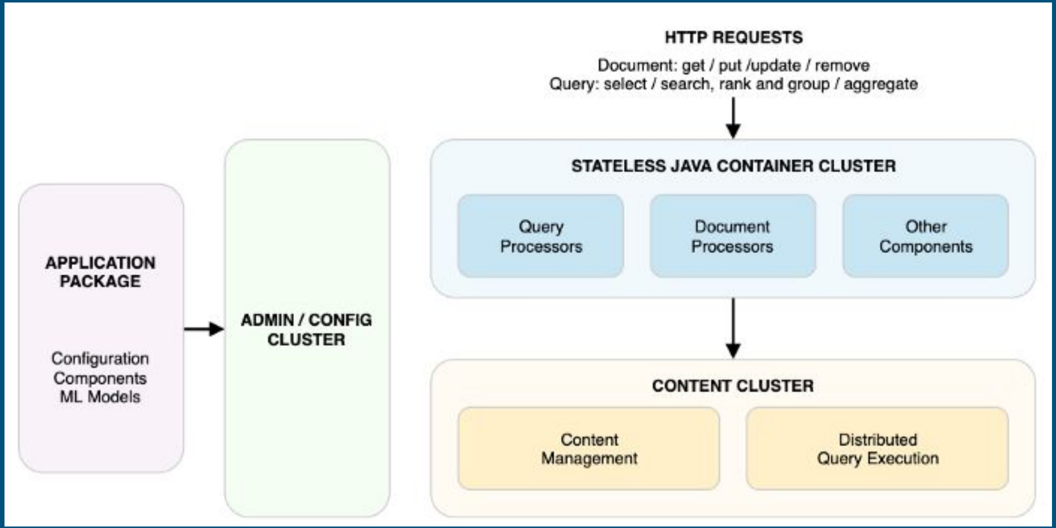
Pinecone

Vespa

🌍 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



vespa

Weaviate

🌍 semi.technology/developers/weaviate/current/

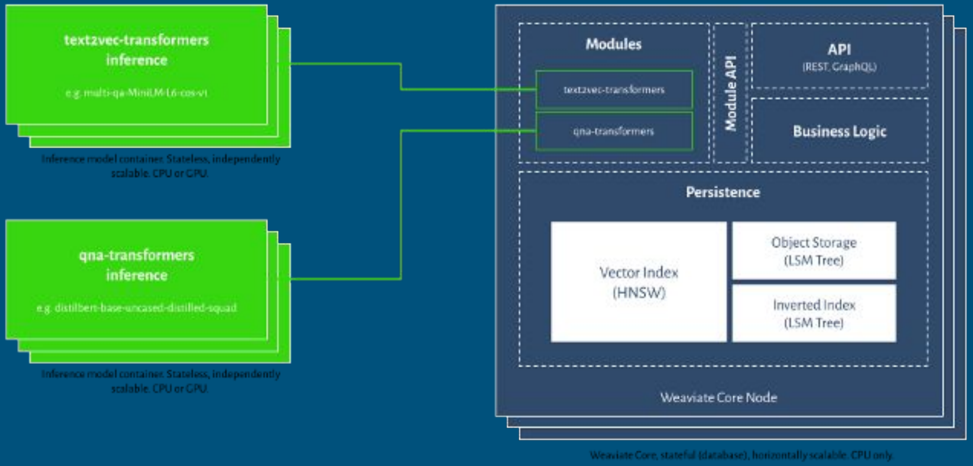
💡 managed / self-hosted

🤖 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

Weaviate System Level Overview (Example with two modules)


text2vec-transformers inference
e.g. multi-qa-MiniLM-L6-cos-v1
Inference model container. Stateless, independently scalable. CPU or GPU.
qna-transformers inference
e.g. distilbert-base-uncased-distilled-squad
Inference model container. Stateless, independently scalable. CPU or GPU.
Modules
text2vec-transformers
qna-transformers
Module API
API
(REST, GraphQL)
Business Logic
Persistence
Vector Index (HNSW)
Object Storage (LSM Tree)
Inverted Index (LSM Tree)
Weaviate Core Node
Weaviate Core, stateful (database), horizontally scalable. CPU only.

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

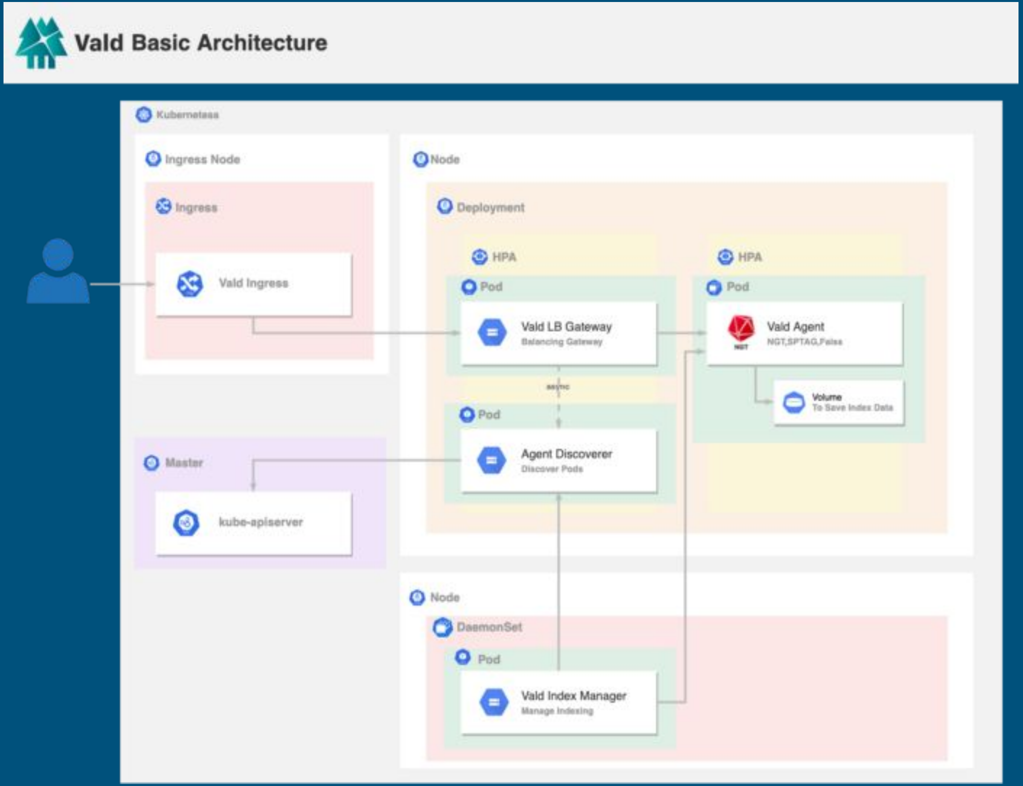
Persistence in Weaviate Core shows one shard as an example. Users can create any number of indices, each index can contain any number of shards. Shards can be distributed and/or replicated across nodes in the cluster. A shard always contains object, inverted and vector storage. Vector storage is not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

Vald

🌍 Link: 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

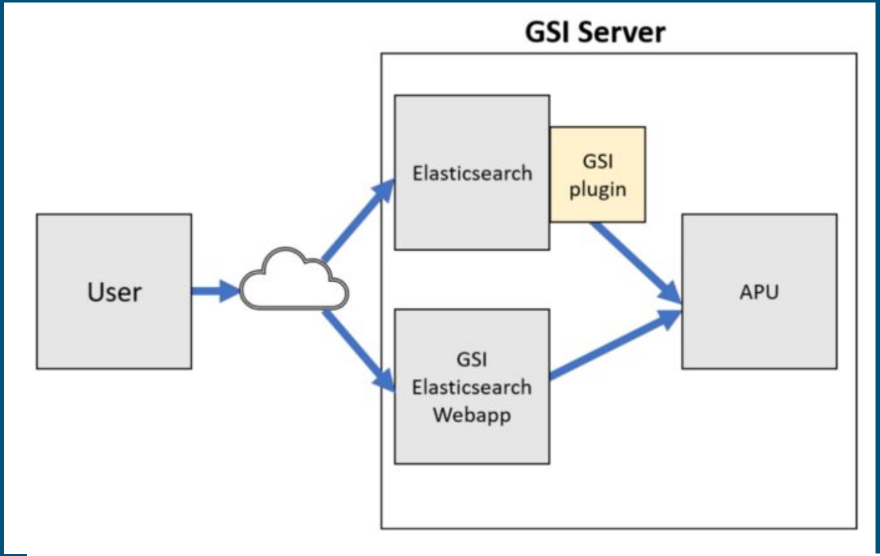


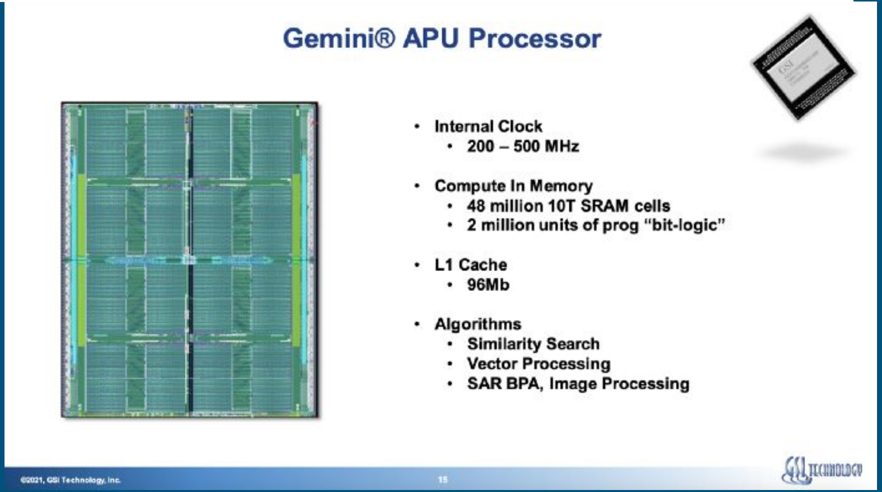
GSI APU

🌍 Link: 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

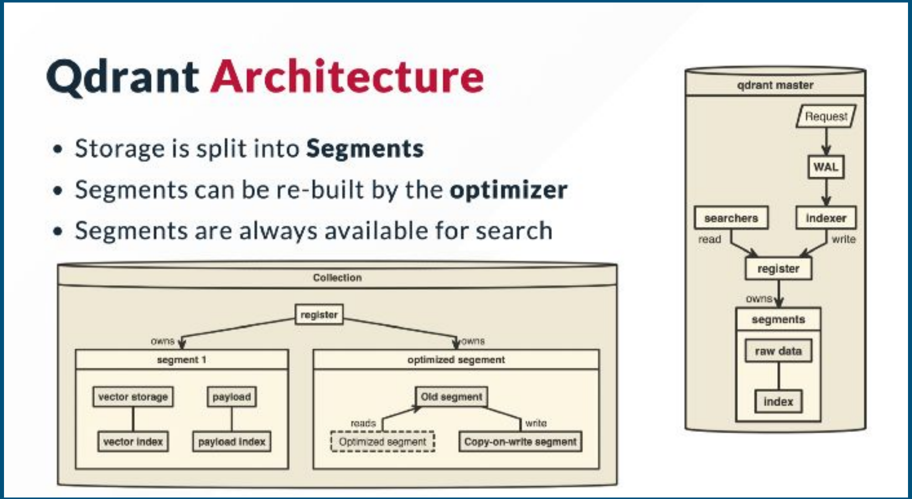
©2021, GSI Technology, Inc. 15

Qdrant

🌍 qdrant.tech/
💡 self-hosted vector database + Cloud
🤖 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

Semantic frameworks / layers: 57% Open Source

Company	Product	Open Source: Y/N	Focus
Deepset.ai	Haystack	Y	NLP, neural search
Jina.AI	Jina, Hub, Finetuner	Y	NLP, CV, ASR
Featureform	Feature store, EmbeddingHub	Y	All AI verticals
ZIR.AI	AI search platform	N	NLP
Hebbia.AI	Knowledge Base	N	NLP -> Finance
Rasa.ai	Virtual assistants	Y	NLP
Muves.io	Multilingual vector search	N	Multilingual search, multimodality

Vector Search Pyramid

user interface

Application business logic: neural / BM25, symbolic filters, RAG, ranking

Encoders: BERT, Clip, GPT3… + Mighty 25%

Neural frameworks: Haystack, Jina.AI, Vectara, Hebbia.AI, txtai … 43%

Vector Databases: Milvus, Weaviate, Pinecone, GSI, Qdrant, Vespa, Vald, Elastiknn… 71%

KNN / ANN algorithms: HNSW, PQ, IVF, LSH, Zoom, DiskANN, BuddyPQ … 100 %

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

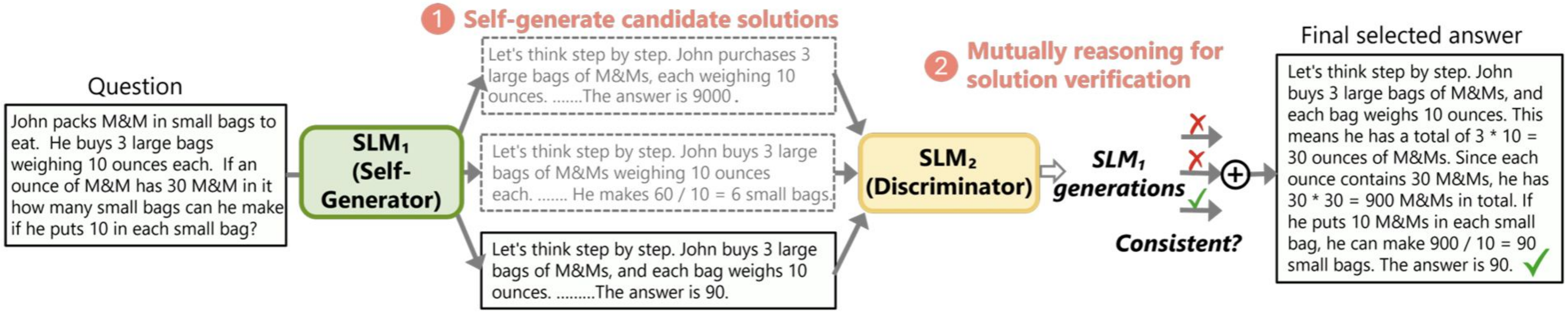

① Self-generate candidate solutions
Question
John packs M&M in small bags to eat. He buys 3 large bags weighing 10 ounces each. If an ounce of M&M has 30 M&M in it how many small bags can he make if he puts 10 in each small bag?
SLM₁ (Self-Generator)
Let's think step by step. John purchases 3 large bags of M&Ms, each weighing 10 ounces. ....The answer is 9000.
Let's think step by step. John buys 3 large bags of M&Ms weighing 10 ounces each. .... He makes 60 / 10 = 6 small bags.
Let's think step by step. John buys 3 large bags of M&Ms, and each bag weighs 10 ounces. ....The answer is 90.
② Mutually reasoning for solution verification
SLM₂ (Discriminator)
SLM₁ generations
Consistent?
Final selected answer
Let's think step by step. John buys 3 large bags of M&Ms, and each bag weighs 10 ounces. This means he has a total of 3 * 10 = 30 ounces of M&Ms. Since each ounce contains 30 M&Ms, he has 30 * 30 = 900 M&Ms in total. If he puts 10 M&Ms in each small bag, he can make 900 / 10 = 90 small bags. The answer is 90.

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.

https://arxiv.org/pdf/2408.06195v1

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

Answers (class, 2025)

1. Field: **hospitality**, hands-free speaking to a LLM agent, speech recognition. Retrieve list of guests, distribute to tables. Risks: omits guests, places to wrong tables, privacy.
2. Field: **healthcare**, in countryside areas (lack of doctors). The system helps to sort the patients based on symptoms. Risk: wrong sorting of patients.
3. Field: **vacation planner**, conversation agent to plan your trip. Risks: biases (product promotions), mistakes in planning (wrong hotel etc).
4. Field: **early education**, parenting assistant (parents lack time / knowledge). Risk: high. Content must be controlled.
5. Field: **healthcare**, affordable. Risks: wrong advice, privacy.
6. Field: **education**, language learning, course material support. Risk: LLM gives the full answer, and you learn nothing. Low-resource languages are a challenge for LLMs.

Answers (class, 2024)

1. Law-bot in Legal. Problem: expensive lawyers, legal text is hard to understand. Solution: ask the bot to represent yourself in court. Risks: judge bots. Benefit: more accessible lawyer practice
2. Legal AI Assistant. Problem: small companies cannot compete with legal departments of large companies. Risks: a) irrelevant information b) bias. Mitigation: context in RAG. Benefit: legal advice, easy access
3. Healthcare bot. Problem: time pressure for the doctor between patients. Solutions: summarize patient history with RAG. Use the context of the visit.

Further study

- Ichigo paper (mixed-modal model to handle both speech and text): 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 (reasoning LLM): https://arxiv.org/pdf/2408.06195v1 code: https://github.com/zhentingqi/rStar
- GitHub Copilot: https://github.blog/ai-and-ml/github-copilot/inside-github-working-with-the-llms-behind-github-copilot/
- Muves Papers: AI research assistant: https://papers.muves.io/

Further study

- ColPali demo: https://huggingface.co/spaces/manu/ColPali-demo
- Podcast episode with the Cursor team: https://lexfridman.com/cursor-team/
- Improved video LLM: https://arxiv.org/pdf/2311.18445v1
- Vector Podcast on YouTube: https://www.youtube.com/c/VectorPodcast
- Vector Podcast on Spotify: https://open.spotify.com/show/13JO3vhMf7nAqcpvlIgOY6
- Vector Podcast on Apple Podcasts: https://podcasts.apple.com/us/podcast/vector-podcast/id1587568733
- More use cases from Weaviate: https://weaviate.io/developers/weaviate/more-resources/example-use-cases