DevFest

Jakarta

Zero to Viral: Build TikTok-style Recommendation

linkedin.com/in/yodiaditya

Google Developer Groups

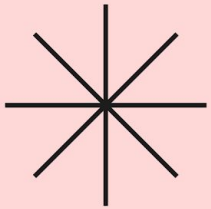
Disclaimer

These are the personal opinions of the author(s). Any assumptions, opinions stated here are theirs and not representative of or attributable to their current or any prior employer(s). Apart from publicly available information, any other information here is not claimed to refer to any company including ones the author(s) may have worked in or been associated with.

AI
@DevFest

Google
Developer
Groups

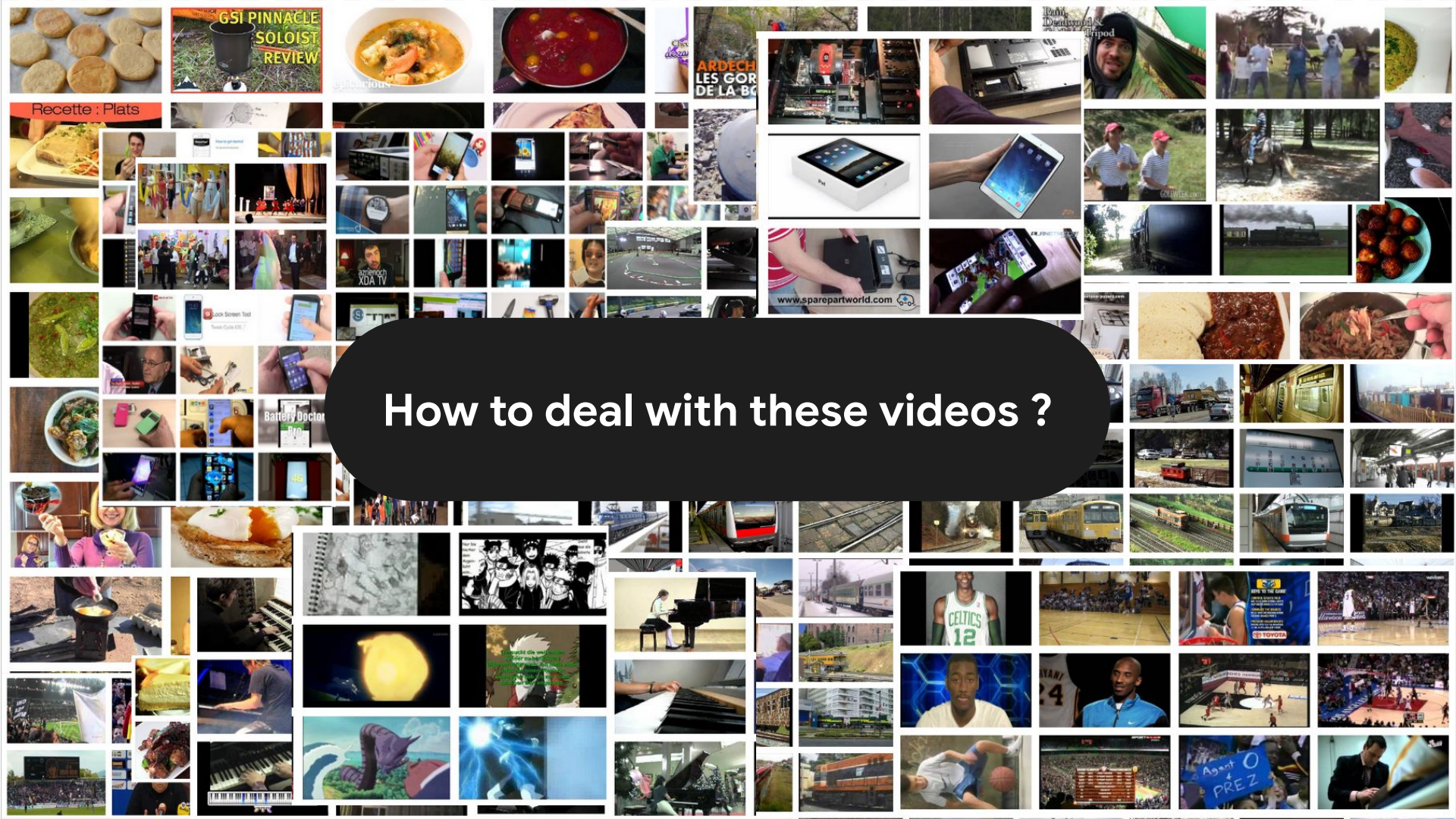
How do you build short videos recommendation when all have you just raw files with no metadata and nothing else ?



Statistics

Estimated 1 Billion views per day

34 million TikTok videos are uploaded daily


GSI PINNACLE SOLOIST REVIEW
Recette : Plats
How to deal with these videos ?

What you will get in next 17 minutes

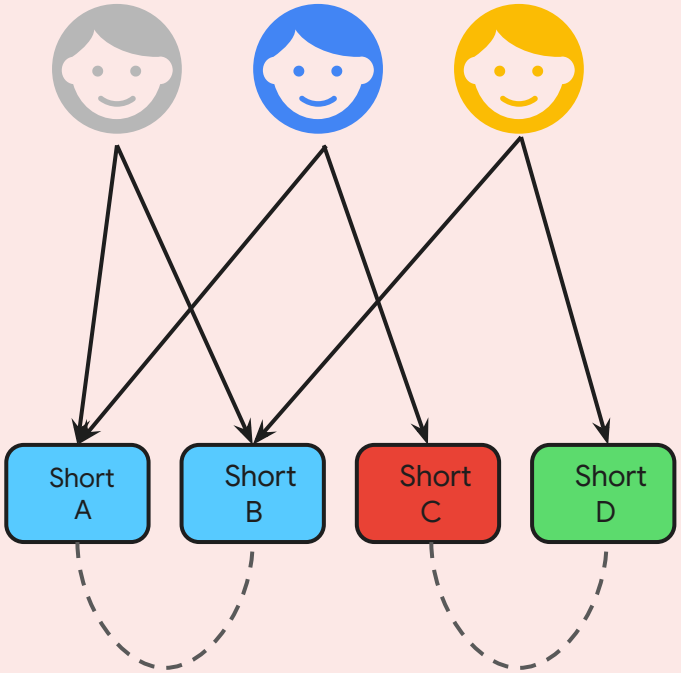
1. Intro: Recommendation System
2. Architecture: Raw Dataset to Final Results
3. Code (Yes, its code!)
4. Tips to Scale

A bit about myself

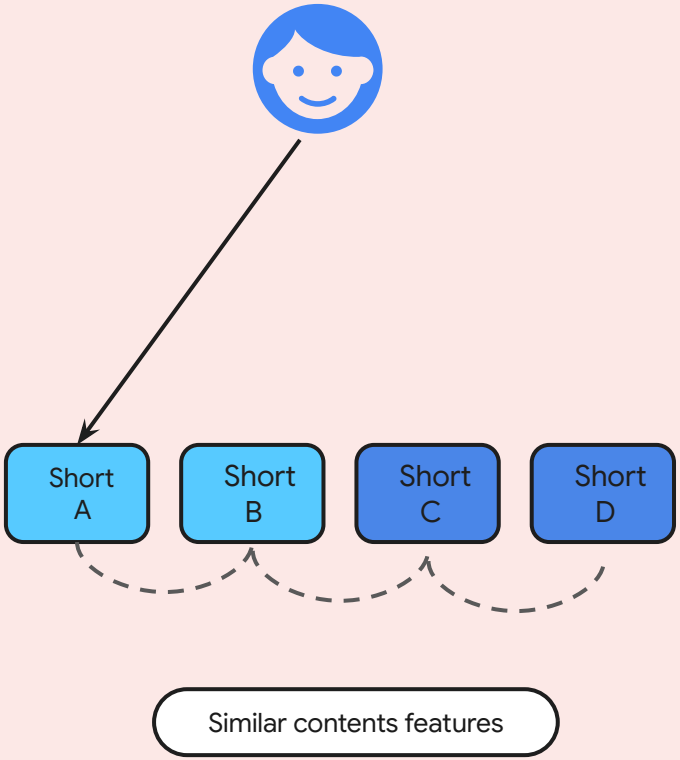
- Build short-videos recommendation at scale in production serving millions of users
- Certified Google Cloud Professional Machine Learning Engineer (PMLE).
- 15 years working experience multi-domains and loves open-source !

https://github.com/yodiaditya

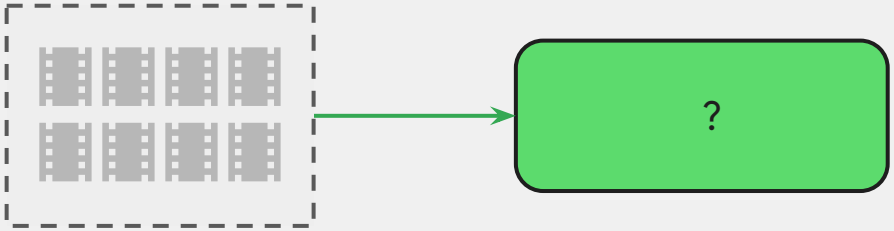
Collaborative Filtering



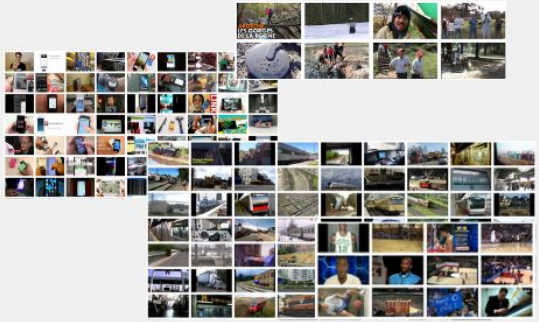
Content Based Filtering

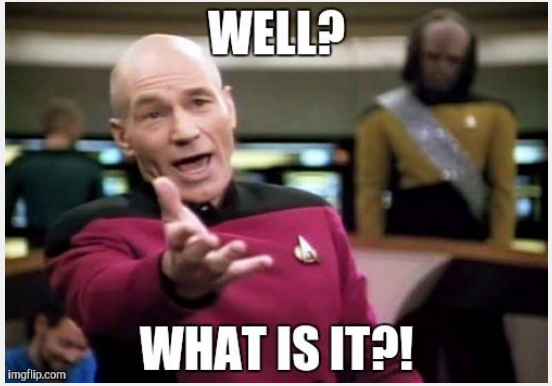


Users upload videos to server

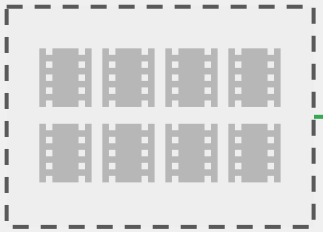


Shorts Videos
(Raw version)





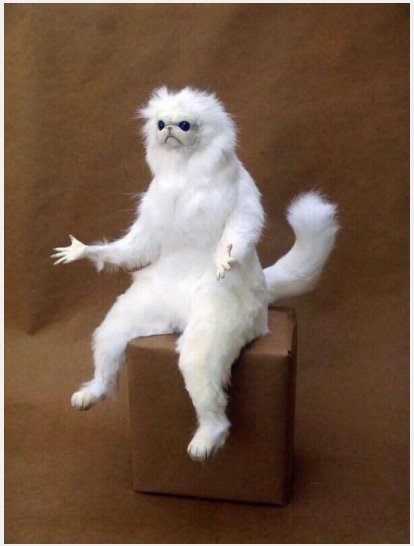
First step is to extract the information from the videos



Feature Extraction

Images

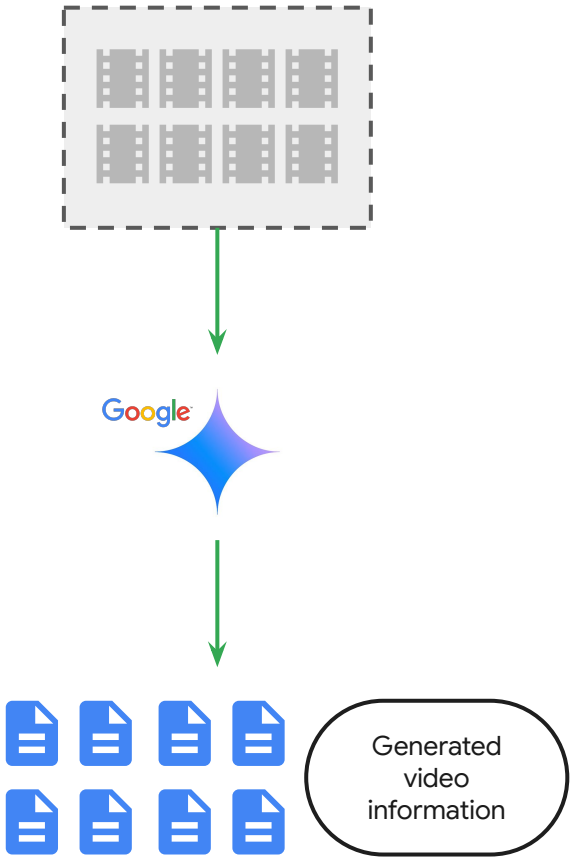
Audio



Gemini: Video Understanding

Model	Video modality details
Gemini 1.5 Flash Go to the Gemini 1.5 Flash model card	Maximum video length: • With audio: ~50 minutes • Without audio: 60 minutes Maximum videos per prompt: 10
Gemini 1.5 Pro Go to the Gemini 1.5 Pro model card	Maximum video length: • With audio: ~50 minutes • Without audio: 60 minutes Maximum videos per prompt: 10
Gemini 1.0 Pro Vision Go to the Gemini 1.0 Pro Vision model card	Maximum video length: 2 minutes The maximum videos per prompt: 1 Audio in the video is ignored.

https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/video-understanding



Next, Embedding Space

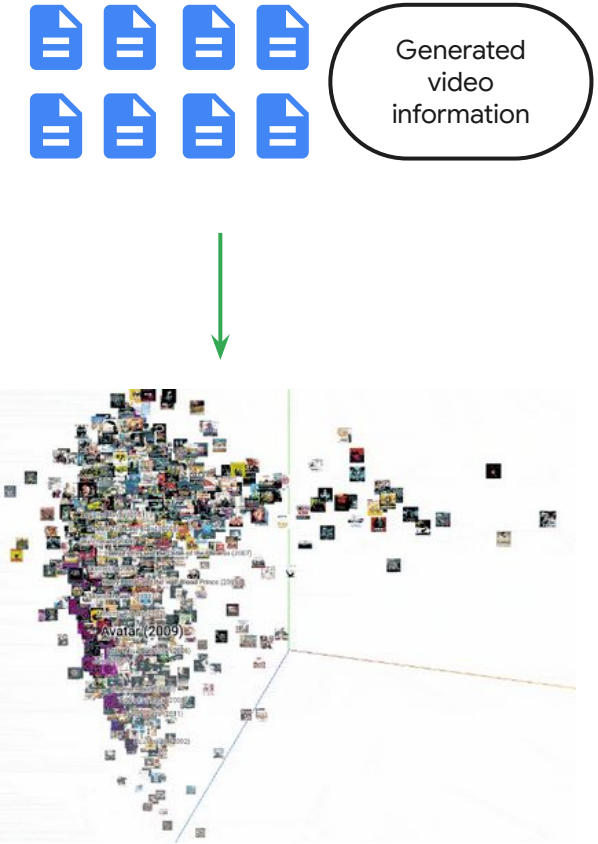
Supported models

You can get text embeddings by using the following models:

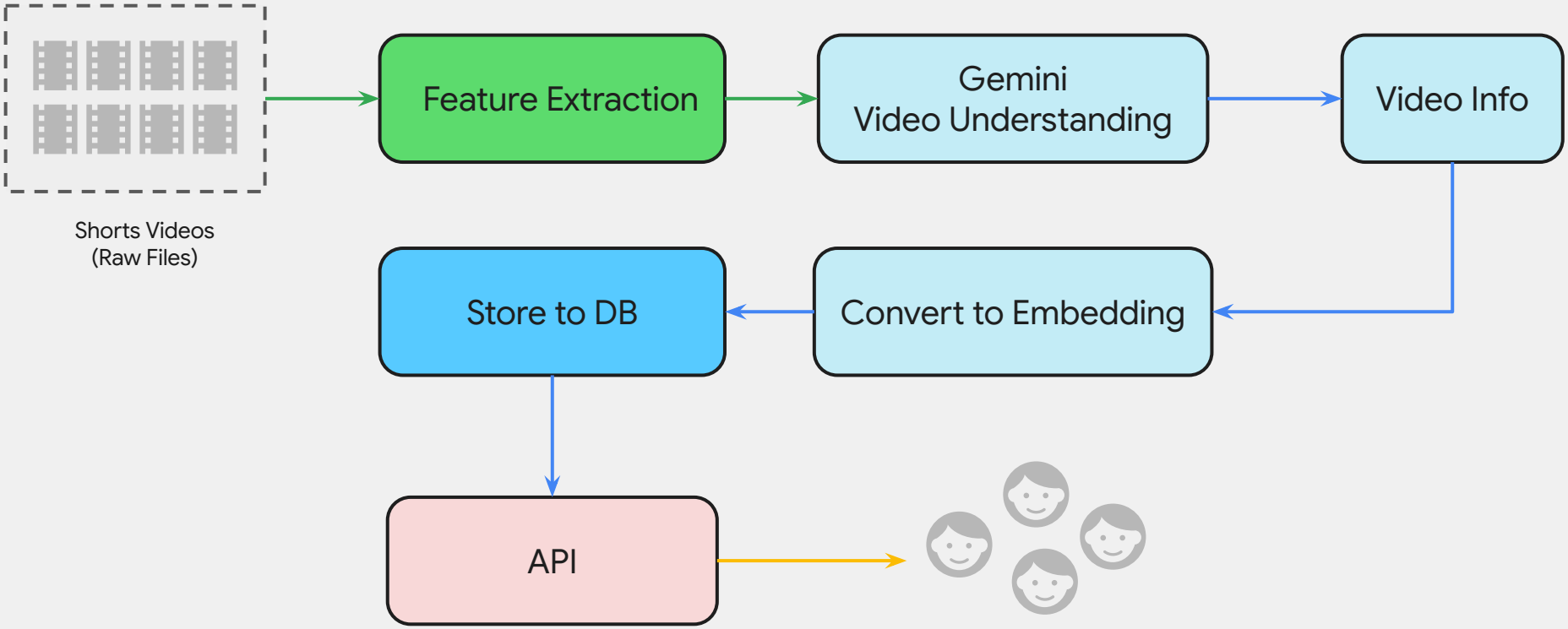
English models	Multilingual models
`textembedding-gecko@001`	`textembedding-gecko-multilingual@001`
`textembedding-gecko@003`	`text-multilingual-embedding-002`
`text-embedding-004`	
`text-embedding-005`	

If you are new to these models, we recommend that you use the latest versions. For English text, use `text-embedding-005`. For multilingual text, use `text-multilingual-embedding-002`.

https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings



Architecture Content-Based Raw Shorts Videos



Show me the code

```python
import vertexai
from vertexai.generative_models import GenerativeModel, Part

PROJECT_ID = 'YOUR-PROJECT-ID'
vertexai.init(project=PROJECT_ID, location="us-central1")
vision_model = GenerativeModel("gemini-1.5-flash-002")

# Generate text
response = vision_model.generate_content(
  [
    Part.from_uri(
      "gs://shorts-hdr-dataset/videos/sdr/SDR_Animal_45j4.mp4", mime_type="video/mp4"
    ),
    "Watch each frame in video. Do not make up any information that is not part of the video. Generate title of video, genres, short description and taggings of content ",
  ]
)
print(response.text)
```

https://media.withyoutube.com/sfv-hdr

Here's a breakdown of the video based on your request:

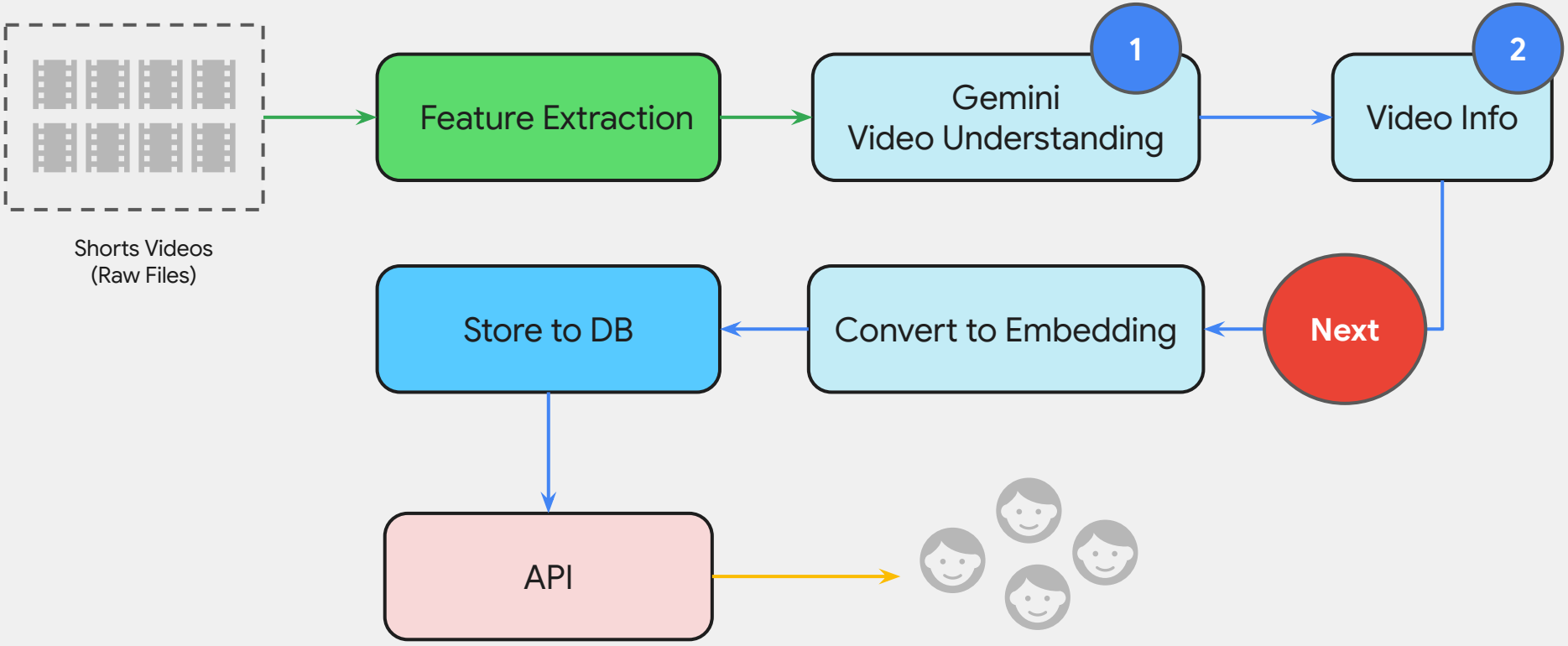
Title: Fluffy Persian Cat

Genres: Animals, Cats, Pets

Short Description: A beautiful fluffy Persian cat sits on a wooden pallet.

Taggings: Persian cat, fluffy cat, cat, kitten, pets, animals, cute, adorable, white cat, long hair cat

Architecture Content-Based Raw Shorts Videos



```
import google.generativeai as genai
import os

genai.configure(api_key=os.environ["GEMINI_API_KEY"])

text = f"""
    *Title:**  Fluffy Persian Cat **Genres:** Animals, Cats, Pets
    **Short Description:** A beautiful fluffy Persian cat sits on a wooden pallet.
    **Taggings:** Persian cat, fluffy cat, cat, kitten, pets, animals, cute, adorable, white cat, long
hair cat"""

result = genai.embed_content(
    model="models/text-embedding-004",
    content=text)

print(str(result['embedding']))
```

[-0.07675823, 0.027660733, 0.049145754, -0.045824636, 0.061620172, 0.032313757,
-0.015517017, -0.03240082, 0.015460015, 0.062071748, -0.03166332, 0.036584534,
0.024837952, 0.009332091, 0.04251558, -0.029856514, 0.05006616, 0.073106274, -0.029902106,
-0.025679907, 0.084334895, -0.017889986, 0.059542187, 0.0013102485, -0.04699988,
-0.007112105, 0.055407282, 0.018771457, 0.07350461, -0.01813061, 0.04455579, 0.064248785,
0.00448263, 0.008756887, 0.047846865, -0.025803361, -0.019773666, 0.041618045,
0.051186264, -0.07005315, -0.033514354, -0.026876828, -0.0008525868, 0.08978644,
0.027724743, 0.010746625, -0.061409652, 0.026029717, -0.0034387638, 0.035402797,
0.007589881, -0.023214854, -0.019220363, 0.02619021, -0.026539033, 0.012508128,
0.0040561967, -0.028133305, -0.016629174, 0.020951482, 0.06473735, 0.034282178,
0.051744517, 0.019634075, -0.016341783, 0.021416675, -0.05009744, -0.0076771406,
-0.036871992, 0.095764354, -0.02469845, 0.025160229, -0.046339314, 0.024582243,
-0.008852839, -0.010245625, 0.011931524, 0.037149847, -0.027476897, 0.021095295,
-0.0048002778, 0.012314561, 0.09466355, -0.028999066, -0.010545491, 0.019936696,
-0.0075119766, 0.02987802, -0.08925607, 0.0030204963, 0.06398722, -0.005665603,
0.055336952, 0.031011213, 0.0803204, -0.046087332, -0.09462911, -0.029559113, 0.017180113,
0.035609562, -0.041688204, 0.016256401, -0.018201571, -0.06340068, 0.05076339,
0.00789683, -0.04663778, 0.003865695, -0.02993782, 0.05526901, -0.059481055, -0.08222241,
-0.044425923, 0.03254365, 0.052314047, 0.077546, 0.0024010574, 0.030455906, -0.017964754,
-0.03830613, 0.042473085, 0.021389754, 0.031468235, -0.0121501945, -0.010297442,
0.012780487, -0.0086625, -0.019988453, 0.017340286, -0.009815917, 0.068223, 0.02263392,
0.02600014, 0.03895912, -0.01126567, 0.03847954, 0.035594787, -0.08233705, 0.007975588,
-0.010537024, -0.042857118, -0.037623975, -0.04778282, 0.007993469, 0.0021377627,
-0.046956647, -0.019428788, 0.068178736, -0.002273978, 0.032764167, 0.022261541,
-0.050062243, -0.0049963817, -0.0493768, 0.027171357, 0.030386483, 0.035637327,
-0.026583953, -0.05276558, -0.051282965, 0.060191035, -0.03913675, 0.009324002,
-2.8839599e-05, -0.010929866, -0.0029119896, 0.013345155, -0.013394253, 0.006282062,
-0.029159687, -0.023629908, -0.04224124, -0.045065757, -0.023713488, -0.038271822,
0.046776716, -0.024885133, -0.05454553, -0.07411608, 0.02653326, 0.01615619, -0.017124163,
-0.03665412, -0.07935555, 0.034953605, -0.073878795, -0.035953894, -0.019765098,
0.004043873, 0.024911957, -0.005691729, -0.050333932, -0.000985027, -0.036075074,
0.011225469, -0.016764432, 0.09254729, -0.072244085, -0.037920788, 0.0124474, 0.041128416,
-0.018892227, 0.02873513, 0.03394367, 0.005887189, 0.04828047, -0.015023347, -0.010894143,
0.003211685, -0.042894274, -0.110083245, 0.013853838, 0.02313884, -0.072523855,
0.024407636, 0.02207754, 0.027719164, -0.01624695, 0.040581483, -0.039446704,
0.062565975, 0.014453987, 0.021910533, -0.019675337, 0.04837492, -0.011042686, -0.02570678,
-0.0035492862, -0.012308466, -0.020631982, -0.035574436, 0.042452924, 0.022002371,
-0.012221022, -0.039625946, 0.009512582, 0.0071309633, -0.01212419, -0.014330538,
0.06173416, -0.007517297, 0.016374068, 0.08243889, 0.0066845445, 0.053010326,
0.004689015, 0.027499422, 0.05117024, -0.06105194, -0.0407715, -0.006838235, 0.007643424,
-0.028924558, 0.031687874, -0.06336678, -0.01165728, -0.003804315, 0.04765609,
-0.008533617, 0.006844215, -0.026217818, -0.049941003, -0.010676516, -0.039658975, ...

Tips to scale

Gemini: Video Understanding

Limitations

While Gemini multimodal models are powerful in many multimodal use cases, it's important to understand the limitations of the models:

- **Content moderation**: The models refuse to provide answers on videos that violate our safety policies.
- **Non-speech sound recognition**: The models that support audio might make mistakes recognizing sound that's not speech.
- **High-speed motion**: The models might make mistakes understanding high-speed motion in video due to the fixed 1 frame per second (fps) sampling rate.
- **Transcription punctuation**: *(if using Gemini 1.5 Flash)* The models might return transcriptions that don't include punctuation.

https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/video-understanding

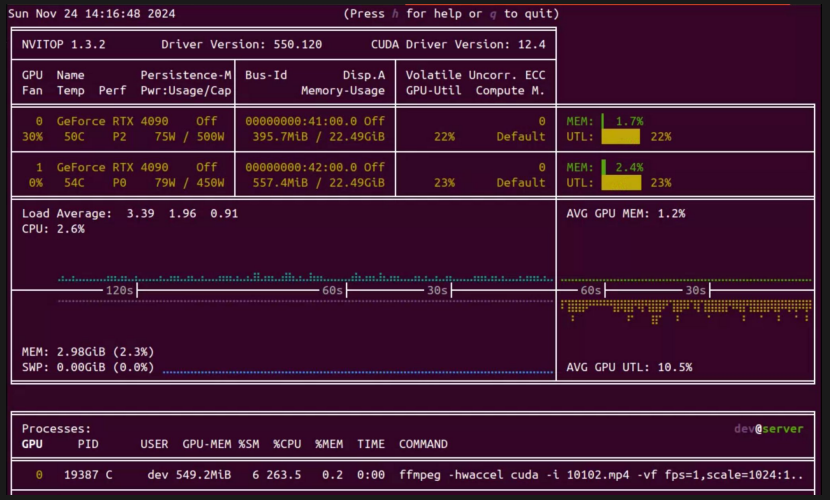
```bash
#!/bin/bash

# Create the output folder if it doesn't exist
mkdir -p small
files=(*.mp4)

for i in "${!files[@]}"; do
 file="${files[$i]}"
 filename=$(basename "$file" .mp4)
 output="small/${filename}.mp4"
 gpu=$((i % 2))

 CUDA_VISIBLE_DEVICES="$gpu" ffmpeg -hwaccel cuda -i "$file" -vf
"fps=1,scale=1024:1024:force_original_aspect_ratio=decrease,pad=1024:1024:(ow-iw)/2:(oh-ih)/2" \
  -c:v h264_nvenc -preset slow -cq 30 -c:a aac -b:a 96k -movflags +faststart "$output" &

 echo "Started processing: $file on GPU $gpu"
 # Wait for processes to complete every two files
 if ((i % 2 == 1)); then
  wait
 fi
done
```



1. Use FFMPEG-GPU NVIDIA
2. Reduce size and apply FPS 1

Process 2,6K videos: 31GB to 2.7GB (90% reduction) with FPS 1. 10x faster speed than CPU.

Use PostgreSQL and PGVector

Enable the extension (do this once in each database where you want to use it)

```
CREATE EXTENSION vector;
```

Create a vector column with 3 dimensions

```
CREATE TABLE items (id bigserial PRIMARY KEY, embedding vector(3));
```

Insert vectors

```
INSERT INTO items (embedding) VALUES ('[1,2,3]'), ('[4,5,6]');
```

Get the nearest neighbors by L2 distance

```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

Also supports inner product (`<#>`), cosine distance (`<=>`), and L1 distance (`<+>` , added in 0.7.0)

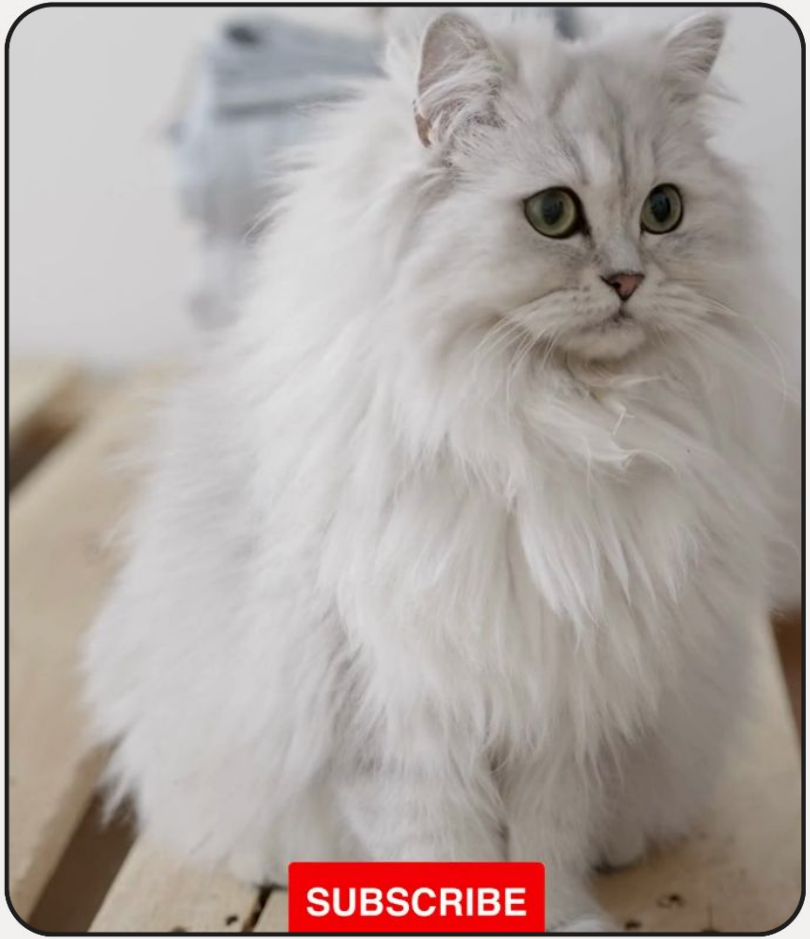
Note: `<#>` returns the negative inner product since Postgres only supports `ASC` order index scans on operators

https://github.com/pgvector/pgvector

Thank you!

Linkedin
linkedin.com/in/yodiaditya

Github
https://github.com/yodiaditya/shorts



Google Developer Groups