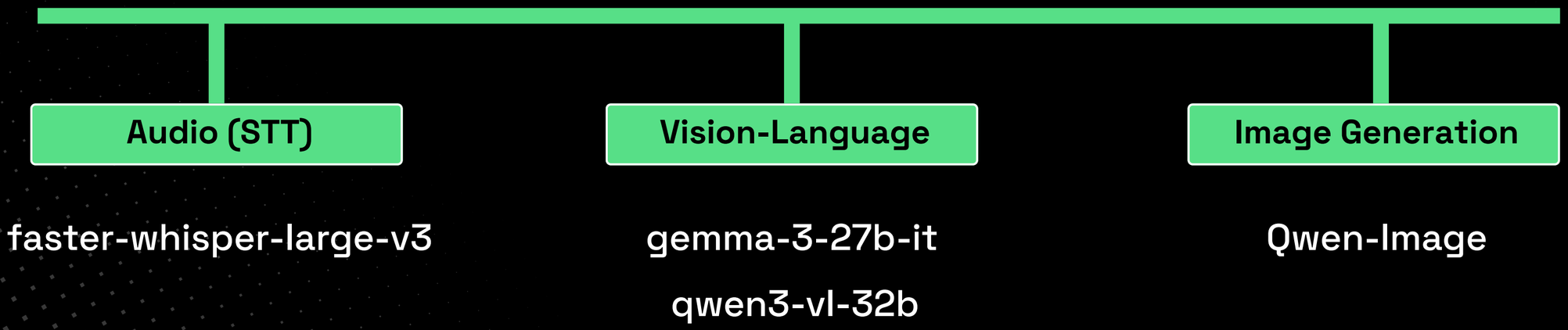
Oltre il testo:

Sviluppare app AI multimodali con regolo.ai

Eugenio Petullà

Nell’ambito dell’Intelligenza Artificiale, il termine **“multimodale”** si riferisce alla capacità di un modello di interpretare ed elaborare informazioni provenienti da diversi tipi di dati, come testo, immagini, audio e video.



Interagire con modelli STT

01 **Chunking (Best Practice)**

Per prevenire allucinazioni o loop di trascrizione, è raccomandato l’invio di segmenti audio inferiori ai **2-3 minuti**.

02 **Timeout Management**

Il limite massimo di elaborazione suggerisce di non superare mai i **5 minuti per singolo file.**

03 **Librerie Standard**

Integrabile istantaneamente con il **client ufficiale di regolo o quello di openai.**

```python
import openai
from pathlib import Path

# OpenAI client configuration
openai.api_key = "YOUR_REGOLO_KEY"
openai.base_url = "https://api.regolo.ai/v1/"

# Audio file to transcribe
AUDIO_FILE = "/path/to/your/audio"
OUTPUT_FILE = "/path/to/output/transcription.txt"

# Transcribe the file
with open(AUDIO_FILE, "rb") as audio_file:
    transcript = openai.audio.transcriptions.create(
        model="faster-whisper-large-v3",
        file=audio_file,
        language="en",
        response_format="text"
    )

# Save the transcription
output_path = Path(OUTPUT_FILE)
output_path.parent.mkdir(parents=True, exist_ok=True)

with open(output_path, "w", encoding="utf-8") as f:
    f.write(transcript)

print(f"Transcription saved to: {OUTPUT_FILE}")
```

Pro-tip per l’integrazione: > L’uso del formato OGG garantisce il miglior rapporto qualità/peso, riducendo i tempi di upload e la latenza complessiva della pipeline.

Interagire con modelli Vision-Language

01 **Struttura Messaggio**

L’input è una lista di oggetti nel campo content, **esattamente come le chat completions**, alternando type: ”text” e type: ”image_url”.

02 **Modelli Verticali**

Alcuni di questi modelli sono addestrati a riconoscere pattern diversi e possono essere utili in ambiti diversi (OCR, Medicina, etc.)

03 **Latenza (Trade-off)**

L’uso di URL Base64 riduce i tempi di risoluzione DNS esterna ma aumenta la dimensione del payload.

```python
import requests

url = "https://api.regolo.ai/v1/chat/completions"
payload = {
    "model": "qwen3-vl-32b",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "Describe this image in detail."
                },
                {
                    "type": "image_url",
                    "image_url": {
                        "url": "https://regolo.ai/images/good-cat.jpg",
                        "format": "image/jpeg"
                    }
                }
            ]
        }
    ]
}
headers = {
    "Content-Type": "application/json",
    "Authorization": "Bearer YOUR_REGOLO_KEY"
}

response = requests.post(url, json=payload, headers=headers)
print(response.json())
```

Pro-tip: > È fondamentale poter impostare nelle chiamate una finestra ampia (almeno 4096 token) per garantire spazio sufficiente ad analisi dettagliate.

Interagire con modelli di Image Generation

01 **Latenza**

La velocità di generazione è influenzata direttamente dalla dimensione selezionata e dal numero di varianti richieste.

02 **Risoluzione**

È fondamentale capire su quali risoluzioni è stato addestrato il modello per evitare artefatti e garantirne la qualità dell'output.

03 **Data Handling**

L'API restituisce i dati nel campo b64_json. È necessario **decodificare la stringa Base64** per ricostruire il file binario (PNG/JPG).

```python
import requests
import json
from PIL import Image
import io
import base64

url = 'https://api.regolo.ai/v1/images/generations'
headers = {
    'Authorization': 'Bearer YOUR_REGOLO_KEY',
    'Content-Type': 'application/json'
}

data = {
    "prompt": "A white cat resting in Rome",
    "n": 2,
    "model": "Qwen-Image",
    "size": "1024x1024"
}

response = requests.post(url, headers=headers, data=json.dumps(data))

if response.status_code == 200:
    response_data = response.json()

    for index, item in enumerate(response_data['data']):
        b64_image = item['b64_json']
        image_data = base64.b64decode(b64_image)

        image_stream = io.BytesIO(image_data)
        image = Image.open(image_stream)

        # Save the Image
        output_path = f"generated_image_{index + 1}.png"
        image.save(output_path)
        print(f"Image saved to: {output_path}")
else:
    print("Failed to generate images:", response.status_code, response.text)
```

Pro-tip: > Per output ad altissima risoluzione, la best practice suggerisce di generare a 1024x1024 e utilizzare un upscaler dedicato post-generazione (Real-ESRGAN).

Esistono tre modi ”banali” ma efficaci per gestire un input multimodale.

Sequential Chain

Smart Routing

Parallel Processing

LangChain

CrewAI

Sequential Chain (Il ”Sandwich”): È il flusso più semplice. **Audio → Testo → Sintesi.** L’informazione fluisce in una sola direzione.

Smart Routing: Un LLM ”regista” riceve l’input e decide quale strumento attivare. Se l’utente invia un’immagine, attiva la Vision; se invia un audio, attiva lo Speech-to-Text.

Parallel Processing: Il più veloce. Mentre il modello trascrive l’audio, un altro modello analizza contemporaneamente i metadati o il sentiment, unendo i risultati alla fine.

regolo.ai/labs

linkedin.com/company/regolo-ai