

SETUP INIZIALE

```
@echo off
setlocal
echo Avvio installazione...

:: Aggiorna pip
python -m pip install --upgrade pip

:: CREA cartella temporanea
set "TMPDIR=%TEMP%\ffmpeg_download"
mkdir "%TMPDIR%" 2>nul

:: SCARICA e decompime FFmpeg (build release essenziale)
echo Download di FFmpeg...
powershell -Command "Invoke-WebRequest -Uri
https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-essentials.z
ip -OutFile '%TMPDIR%\ffmpeg.zip'"

echo Estrazione di FFmpeg...
powershell -Command "Expand-Archive -Path
'%TMPDIR%\ffmpeg.zip' -DestinationPath '%USERPROFILE%\ffmpeg'"

:: Trova la sotto-cartella estratta
for /D %%D in ("%USERPROFILE%\ffmpeg\ffmpeg-*.") do (
    set "FFMPEG_EXTRACTED=%%D\bin"
)

:: Aggiunge FFmpeg al PATH
if exist "%FFMPEG_EXTRACTED%\ffmpeg.exe" (
    echo Aggiunta di FFmpeg al PATH: %FFMPEG_EXTRACTED%
    setx PATH "%PATH%;%FFMPEG_EXTRACTED%"
) else (
    echo ERRORE: Impossibile trovare ffmpeg.exe dopo
l'estrazione.
)

:: Installa Git
winget install --id Git.Git -e --source winget

:: Installa torch (CPU)
pip install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cpu
```

```
:: Installa whisper  
pip install git+https://github.com/openai/whisper.git
```

```
:: Installa yt-dlp  
pip install -U "yt-dlp[default]"
```

```
:: Installa requests  
pip install requests
```

```
echo Installazione completata. Riavvia il terminale per  
applicare le modifiche al PATH.  
pause
```

SCRIPT PYTHON

```
import subprocess
import whisper
import os
import requests
import json
from datetime import datetime

def summarize_with_ollama(text, model="phi3:3.8b"):
    """Riassume il testo usando Ollama"""
    url = "http://localhost:11434/api/generate"

    prompt = f"""Riassumi questa trascrizione in modo chiaro e
strutturato.
Organizza il contenuto in paragrafi, con titoli quando
appropriato.
Mantieni i punti chiave e le informazioni importanti.

Testo da riassumere:
{text}

Riassunto strutturato:"""

    payload = {
        "model": model,
        "prompt": prompt,
        "stream": False,
        "options": {
            "temperature": 0.3,
            "top_p": 0.9
        }
    }

    try:
        response = requests.post(url, json=payload,
timeout=120)
        if response.status_code == 200:
            return response.json()['response']
        else:
            print(f"Errore Ollama: {response.status_code}")
            return None
    except Exception as e:
        print(f"Errore connessione Ollama: {e}")
```

```

        return None

def chunk_text(text, chunk_size=3000):
    """Divide il testo in chunk per modelli con limite di
    contesto"""
    words = text.split()
    chunks = []
    current_chunk = []
    current_length = 0

    for word in words:
        if current_length + len(word) > chunk_size:
            chunks.append(' '.join(current_chunk))
            current_chunk = [word]
            current_length = len(word)
        else:
            current_chunk.append(word)
            current_length += len(word) + 1

    if current_chunk:
        chunks.append(' '.join(current_chunk))

    return chunks

# Input URL
youtube_url = input("URL YouTube: ")

# Nome file con timestamp
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
audio_file = f"audio_{timestamp}.wav"
transcript_file = f"trascrizione_{timestamp}.txt"
summary_file = f"riassunto_{timestamp}.txt"

# Scarica audio
print(f"Scaricando audio...")
subprocess.run([
    "yt-dlp",
    "--extract-audio",
    "--audio-format", "wav",
    "--output", f"audio_{timestamp}%(ext)s",
    youtube_url
])

# Trascrivi

```

```

print("Trascrivendo...")
model = whisper.load_model("tiny")
result = model.transcribe(audio_file)

# Salva trascrizione originale
with open(transcript_file, "w", encoding="utf-8") as f:
    f.write(result["text"])

print(f"✅ Trascrizione salvata: {transcript_file}")

# Riassumi con LLM
print("Generando riassunto...")
transcript_text = result["text"]

# Se il testo è molto lungo, dividilo in chunk
if len(transcript_text) > 3000:
    chunks = chunk_text(transcript_text)
    summaries = []

    for i, chunk in enumerate(chunks):
        print(f"Elaborando chunk {i+1}/{len(chunks)}...")
        summary = summarize_with_ollama(chunk)
        if summary:
            summaries.append(summary)

    # Riassumi i riassunti se ci sono multipli chunk
    if len(summaries) > 1:
        combined_summary = "\n\n".join(summaries)
        final_summary = summarize_with_ollama(
            combined_summary,
            model="phi3:3.8b"
        )
    else:
        final_summary = summaries[0] if summaries else "Errore
nella generazione del riassunto"
    else:
        final_summary = summarize_with_ollama(transcript_text)

# Salva riassunto
if final_summary:
    with open(summary_file, "w", encoding="utf-8") as f:
        f.write(f"RIASSUNTO VIDEO YOUTUBE\n")
        f.write(f"URL: {youtube_url}\n")

```

```
f.write(f>Data: {datetime.now().strftime('%d/%m/%Y
%H:%M')} \n")
f.write("="*50 + "\n\n")
f.write(final_summary)

print(f"✅ Riassunto salvato: {summary_file}")
else:
    print("❌ Errore nella generazione del riassunto")

# Pulizia file audio (opzionale)
cleanup = input("Eliminare file audio? (s/n): ")
if cleanup.lower() == 's':
    os.remove(audio_file)
    print("🗑️ File audio eliminato")

print("\n🎉 Processo completato!")
```