CDW Documentation

Video Keyword Transcription

Video Keyword Transcription

Azure Video + Text Analytics Demo Summary

Service Deployment

1. **Azure Video Indexer (Trial or Full Account)**
   - Created or accessed via videoindexer.ai or the Azure Portal.
   - Required: Account ID and API Key (from API portal or Azure resource).
   - Purpose: Automatically analyze and extract audio, facial recognition, and transcription from videos.
2. **Azure AI Language (Text Analytics)**
   - Deployed via the Azure Portal as a **Language** resource.
   - Required: Endpoint URL and API Key.
   - Purpose: Perform NLP tasks like key phrase extraction, summarization, sentiment analysis.

Script Functionality

The Python script:

- Uploads a video to Azure Video Indexer.
- Waits for indexing to complete.
- Downloads the video transcript.
- Feeds the transcript into Azure Text Analytics to extract **key phrases** (optionally, you could also extract sentiment, entities, or summarization).

⚠ Limitations Encountered and Corrected

Issue	Cause	Solution
`InvalidDocument` errors	Text Analytics API rejected input over 5120 text elements	Implemented intelligent chunking
Word-based chunking failed	Sentence structure and word length varied too much	Switched to **character-count-based chunking**
Sentence-based chunking still failed	Uneven sentence lengths	Character-bound chunking (~4,000 chars max) ensured compliance

The final implementation used `split_text_by_characters()` to ensure every document sent to the API was well under the limit.

Expected Output

- A list of **unique key phrases** extracted from the entire transcript.
- Optionally, these phrases could be:
  - Displayed in a report or dashboard
  - Stored in a database
  - Used for tagging or indexing content
  - Summarized in bullet points for meeting minutes

Key Knowledge Gained

Topic	Insight
Azure AI Integration	How to combine Video Indexer and Text Analytics for multimodal processing
API Authentication	Using API keys and endpoint tokens securely
Real-world NLP limits	Document size limitations and handling token counts gracefully
Practical AI Ops	Building automation around Azure AI services for repeatable workflows
Transcript Optimization	Learning to preprocess text for better reliability and API compliance

Final Script

azure_video_highlight_demo.py

```
#!/Users/don.dehamer/.local/pipx/venvs/azure-ai-textanalytics/bin/python3.13

import time
import requests
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential
import re

# --- CONFIGURATION ---

# Azure Video Indexer
LOCATION = "trial"  # use your region if not trial
ACCOUNT_ID = "ACCOUNTID" # Azure Video Indexer Account ID
VIDEO_INDEXER_API_KEY = "APIKEY"
VIDEO_PATH = "/Path/to/video/video.mp4"  # Local path to your video
VIDEO_NAME = "Nametoshowonupload"

# Azure Text Analytics
TEXT_ANALYTICS_KEY = "APIKEY"
TEXT_ANALYTICS_ENDPOINT = "https://<RESOURCE
NAME>.cognitiveservices.azure.com/"

# --- STEP 1: Get Access Token from Video Indexer ---

def get_access_token():
```

```
    url =
f"https://api.videoindexer.ai/Auth/{LOCATION}/Accounts/{ACCOUNT_ID}/AccessTo
ken?allowEdit=true"
    headers = {"Ocp-Apim-Subscription-Key": VIDEO_INDEXER_API_KEY}
    response = requests.get(url, headers=headers)
    response.raise_for_status()
    return response.text.strip('"')

# --- STEP 2: Upload Video File to Indexer ---

def upload_video(token):
    with open(VIDEO_PATH, 'rb') as video_file:
        files = {'file': (VIDEO_NAME, video_file, 'video/mp4')}
        url =
f"https://api.videoindexer.ai/{LOCATION}/Accounts/{ACCOUNT_ID}/Videos?name={
VIDEO_NAME}&accessToken={token}"
        response = requests.post(url, files=files)
        response.raise_for_status()
        return response.json()['id']

# --- STEP 3: Poll Until Processing is Done ---

def wait_for_processing(token, video_id):
    url =
f"https://api.videoindexer.ai/{LOCATION}/Accounts/{ACCOUNT_ID}/Videos/{video
_id}/Index?accessToken={token}"
    while True:
        response = requests.get(url)
        response.raise_for_status()
        state = response.json().get('state')
        if state == 'Processed':
            return
        print(f"Processing... current state: {state}")
        time.sleep(10)

# --- STEP 4: Download Transcript ---

def download_transcript(token, video_id):
    url =
f"https://api.videoindexer.ai/{LOCATION}/Accounts/{ACCOUNT_ID}/Videos/{video
_id}/Captions?format=ttml&accessToken={token}"
    response = requests.get(url)
    response.raise_for_status()
    return response.text

# --- STEP 5: Extract Key Phrases using Text Analytics ---

def split_text_by_characters(text, max_chars=4000):
    chunks = []
    while len(text) > max_chars:
        # Find the last sentence end before max_chars
```

```
        end = text.rfind('.', 0, max_chars)
        if end == -1:
            end = max_chars  # fallback to hard cut
        chunk = text[:end + 1].strip()
        chunks.append(chunk)
        text = text[end + 1:].strip()
    if text:
        chunks.append(text)
    return chunks

def extract_key_phrases(text):
    client = TextAnalyticsClient(endpoint=TEXT_ANALYTICS_ENDPOINT,
credential=AzureKeyCredential(TEXT_ANALYTICS_KEY))
    chunks = split_text_by_characters(text)
    all_phrases = []

    for i, chunk in enumerate(chunks):
        try:
            response = client.extract_key_phrases([chunk])
            if not response[0].is_error:
                all_phrases.extend(response[0].key_phrases)
            else:
                print(f"Chunk {i} failed: {response[0].error}")
        except Exception as e:
            print(f"Chunk {i} raised exception: {e}")

    return list(set(all_phrases))


# --- MAIN EXECUTION FLOW ---

if __name__ == "__main__":
    token = get_access_token()
    video_id = upload_video(token)
    wait_for_processing(token, video_id)
    transcript = download_transcript(token, video_id)
    key_phrases = extract_key_phrases(transcript)

    print("\n--- Key Phrases Extracted ---")
    for phrase in key_phrases:
        print("-", phrase)
```

AI Knowledge