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)
```

Telemetry

To connect telemetry from this script and the Azure Cognitive Services it uses (Video Indexer and Text Analytics) to **Azure Application Insights**, you would need to **instrument the Python script** to:

1. Send custom telemetry (e.g., video processing status, API latency, errors)
2. Optionally, use **dependency tracking** to capture outbound API calls

Step-by-Step: Connect Script Telemetry to Application Insights

1. Create an Application Insights Resource

1. Go to Azure Portal
2. Click **Create a resource**
3. Search for **Application Insights**
4. Choose:
    - **Resource group**
    - **Region**
    - **Name**
    - **Application Type**: General
5. Click **Create**
6. After it's created, go to the resource and copy the **Instrumentation Key** or **Connection String**

2. Install Application Insights SDK for Python

```
bashCopyEditpip install opencensus-ext-azure opencensus-ext-requests
opencensus
```

3. Add Instrumentation to Your Script

At the top of your script, import and configure the telemetry client:

```
pythonCopyEditfrom opencensus.ext.azure.log_exporter import AzureLogHandler
from opencensus.ext.azure.trace_exporter import AzureExporter
from opencensus.trace.samplers import ProbabilitySampler
from opencensus.trace.tracer import Tracer
import logging

# Replace with your Application Insights connection string
APP_INSIGHTS_CONNECTION_STRING = "InstrumentationKey=<your-key>"

# Set up logging
logger = logging.getLogger(__name__)
logger.addHandler(AzureLogHandler(connection_string=APP_INSIGHTS_CONNECTION_
STRING))
logger.setLevel(logging.INFO)
```

```
# Set up tracing (optional)
tracer = Tracer(
exporter=AzureExporter(connection_string=APP_INSIGHTS_CONNECTION_STRING),
    sampler=ProbabilitySampler(1.0),
)
```

---

4. Log Custom Events, Metrics, and Errors

Throughout your script, add telemetry like this:

```
pythonCopyEditlogger.info("Uploading video to Azure Video Indexer...")

# On success
logger.info("Video uploaded successfully. Video ID: %s", video_id)

# On API error
logger.error("Chunk %d failed: %s", i, response[0].error)

# Add custom dimensions
logger.info("Processing completed", extra={
    "custom_dimensions": {
        "video_id": video_id,
        "transcript_length": len(transcript),
        "total_key_phrases": len(all_phrases)
    }
})
```

---

5. Monitor in Application Insights

After running the script:

- Go to **Application Insights > Logs (Analytics)** and run queries like:

```
kustoCopyEdittraces | where customDimensions.video_id contains "xyz"
```

Or view:

- **Failures** (to track errors)
- **Performance** (request/response durations if traced)
- **Custom Events & Metrics**

Bonus: Dependency Tracking

If you want to **automatically track outbound HTTP requests** (e.g., calls to Video Indexer or Text Analytics APIs):

```
pythonCopyEditimport requests
from opencensus.ext.requests import trace

trace.trace_integration()
```

This will auto-record dependency duration and failures to App Insights.

Summary

Element	Purpose
`AzureLogHandler`	Sends custom logs to App Insights
`Tracer + AzureExporter`	Sends operation spans and dependency telemetry
`trace_integration()`	Automatically tracks outgoing HTTP requests
Custom `logger.info()`	Manually report app behavior, events, errors

Combined code in one script

```
Fun Fact:  Python 3.13 broke compatibility with Analytics so you have to run an older version for this to work.
```

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

import time
import requests
import re
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential
from opencensus.ext.azure.log_exporter import AzureLogHandler
from opencensus.ext.azure.trace_exporter import AzureExporter
from opencensus.trace.samplers import ProbabilitySampler
from opencensus.trace.tracer import Tracer
import logging

# --- CONFIGURATION ---

# Azure Video Indexer
LOCATION = "trial"  # or your region
ACCOUNT_ID = "your_account_id"
VIDEO_INDEXER_API_KEY = "your_video_indexer_api_key"
VIDEO_PATH = "sample_video.mp4"
VIDEO_NAME = "demo_video_ai"
```

```
# Azure Text Analytics
TEXT_ANALYTICS_KEY = "your_text_analytics_key"
TEXT_ANALYTICS_ENDPOINT =
"https://your-textanalytics-resource.cognitiveservices.azure.com/"

# Application Insights
APP_INSIGHTS_CONNECTION_STRING =
"InstrumentationKey=your_instrumentation_key"

# --- Logging and Telemetry Setup ---
logger = logging.getLogger(__name__)
logger.addHandler(AzureLogHandler(connection_string=APP_INSIGHTS_CONNECTION_
STRING))
logger.setLevel(logging.INFO)

tracer = Tracer(
exporter=AzureExporter(connection_string=APP_INSIGHTS_CONNECTION_STRING),
    sampler=ProbabilitySampler(1.0),
)

# --- Utility Functions ---

def split_text_by_characters(text, max_chars=4000):
    chunks = []
    while len(text) > max_chars:
        end = text.rfind('.', 0, max_chars)
        if end == -1:
            end = max_chars
        chunk = text[:end + 1].strip()
        chunks.append(chunk)
        text = text[end + 1:].strip()
    if text:
        chunks.append(text)
    return chunks

# --- Azure Video Indexer Functions ---

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('"')

def upload_video(token):
    logger.info("Uploading video to Azure Video Indexer...")
    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()
        video_id = response.json()['id']
        logger.info("Video uploaded successfully.",
extra={"custom_dimensions": {"video_id": video_id}})
        return video_id

def wait_for_processing(token, video_id):
    logger.info("Waiting for video indexing to complete...")
    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')
        logger.info(f"Indexing state: {state}")
        if state == 'Processed':
            return
        time.sleep(10)

def download_transcript(token, video_id):
    logger.info("Downloading transcript from Azure Video Indexer...")
    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

# --- Azure Text Analytics Function ---

def extract_key_phrases(text, video_id):
    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)
                logger.info(f"Processed chunk {i + 1} successfully.",
extra={"custom_dimensions": {"chunk_length": len(chunk)}})
            else:
                logger.error(f"Chunk {i + 1} failed: {response[0].error}",
extra={"custom_dimensions": {"chunk": i + 1}})
        except Exception as e:
```

```
            logger.exception(f"Chunk {i + 1} raised exception: {str(e)}",
extra={"custom_dimensions": {"chunk": i + 1}})

    logger.info("All chunks processed.", extra={"custom_dimensions":
{"video_id": video_id, "total_key_phrases": len(all_phrases)}})
    return list(set(all_phrases))

# --- MAIN EXECUTION FLOW ---

if __name__ == "__main__":
    try:
        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, video_id)

        print("\n--- Key Phrases Extracted ---")
        for phrase in key_phrases:
            print("-", phrase)
    except Exception as e:
        logger.exception("Unhandled exception occurred during processing.")
```

AI Knowledge