5장 제미나이 API로 유튜브 동영상 인식하기

제미나이 API를 사용하여 인공지능에게 유튜브 동영상을 묘사하거나 요약하라는 작업을 지시할 수 있습니다. 하지만 문제가 하나 있습니다. 현재 구글은 구글 클라우드 스토리지(Google Cloud Storage, GCS)나 File API(제미나이 인식용 임시 파일 저장소)에 올라가 있는 영상만 제미나이를 통해 인식하도록 구성해 놓았습니다. 따라서 영상을 인식하려면 ①영상 다운로드 → ② 영상 업로드 → ③ 영상 인식이라는 과정이 필요한데, 편리한 사용을 위해 이 3가지 과정을 파이프라인(Pipeline)으로 자동화하는 것이 좋습니다. 이번 절에서는 File API를 사용해 유튜브 동영상을 인식하는 방법을 다루면서, 위의 단계를 파이프라인으로 만드는 방법도 함께 알아보겠습니다.

6.1. 유튜브 동영상 인식

몇 년 전 큰 화제를 모았던 “Feel the Rhythm of Korea” 시리즈의 <서울 편> 동영상으로 실습하겠습니다.

다음 순서대로 제미나이가 유튜브 동영상을 인식하게 하겠습니다. 1) 영상을 로컬 컴퓨터로 다운로드 2) 다운로드된 영상을 File API를 통해 업로드 3) 제미나이 API로 영상 인식 4) 로컬 컴퓨터와 File API에서 영상 삭제

유튜브 동영상 다운로드

유튜브 영상을 내려받는 데 사용할 pytubefix 패키지를 설치한 후 아래 코드를 실행하여 유튜브 영상을 파일로 저장합니다.

```
from pytubefix import YouTube
url = "https://www.youtube.com/watch?v=i-E7NiyRDa0"

yt = YouTube(url)
stream = yt.streams.get_highest_resolution()

# 동영상 다운로드
file_path = stream.download(output_path="./videos")
```

File API로 유튜브 인식하기

File API

제미나이 API를 사용할 때 텍스트, 이미지, 오디오 등 멀티모달 파일을 프롬프트에 포함시키려면 파일을 저장할 수 있는 별도의 공간이 필요합니다. 기존에는 Google Cloud Storage(GCS)에 파일을 업로드한 후 제미나이 API에서 참조하는 방식을 사용했습니다. 그러나 이 방법은 구글 클라우드 플랫폼의 스토리지 서비스를 사용해야 하는 번거로움이 따랐고 Vertex AI 플랫폼을 사용해야만 했습니다. File API는 이러한 불편함을 해소하기 위해 2024년 4월 처음 도입됐습니다.

File API를 사용하면 제미나이 API와 동일한 인터페이스로 파일을 업로드, 조회, 삭제할 수 있으므로 GCS를 별도로 사용할 필요가 없습니다. 또한 File API는 제미나이 API와 통합되어 있어 업로드한 파일을 프롬프트에 쉽게 포함시킬 수 있습니다. 이를 통해 개발자는 멀티모달 입력을 보다 간편하게 처리할 수 있습니다.

File API의 또 다른 장점은 무료로 제공된다는 점입니다. 제미나이 API를 사용할 수 있는 모든 지역에서 추가 비용 없이 File API를 사용할 수 있으므로 비용 부담을 줄일 수 있습니다. 다만 API 키를 통해 업로드된 파일에 접근할 수 있으므로 키 관리에 주의가 필요합니다.

동영상 업로드하기

앞서 말한 것처럼 별도의 패키지가 아닌, 모델 패키지인 generativeai의 upload_file 메서드를 통해 파일을 업로드합니다.

```
import google.generativeai as genai

model = genai.GenerativeModel(model_name="gemini-1.5-flash")
genai.configure(api_key="구글API  Key")

uploaded_file = genai.upload_file(path=file_path)
print("uploaded_file.uri:", uploaded_file.uri)
```

동영상 인식하기

제미나이 API는 멀티모달을 추구하기 때문에 텍스트 생성, 이미지 인식, 영상 인식 사용 방법이 거의 같습니다. contents 파라미터에 프롬프트와, File API로 업로드한 파일 경로를 리스트로 할당하기만 하면 됩니다.

```
iimport IPython.display  # 동영상 플레이어 출력을 위해 추가

prompt = """
유튜브를 보고 아래에 답하세요.
- 영상에 등장하는 춤을 추는 인물은 몇 명인가요?
- 각각의 인물에 대한 특징을 짧게 기술하세요.
"""
contents = [prompt, uploaded_file]
responses = model.generate_content(contents, stream=True, request_options={"timeout": 60*2})

IPython.display.display(IPython.display.Video(file_path, width=800 ,embed=True))
for response in responses:
```

```
print(response.text.strip(), end="")
```

실행 결과:

```
<IPython.core.display.Video object>
영상에 등장하는 춤추는 인물은 8명입니다.- 빨간색 옷을 입고 모자를 쓴 인물: 가장 눈에 띄는 인물이며, 춤을 주도적으로 이끌어 나갑니다.
- 검은색 옷을 입고모자를 쓴 인물: 빨간색 옷을 입은 인물과 함께 춤을 이끌어 나가는 역할을 합니다.
- 흰색 옷을 입고 모자를 쓴 인물: 밝은 옷차림으로 춤을 즐기는 모습을 보여줍니다.
- 파란색옷을 입고 모자를 쓴 인물: 빨간색 옷을 입은 인물과 함께 춤을 이끌어 나가는 역할을 합니다.
- 초록색 옷을 입고모자를 쓴 인물: 빨간색 옷을 입은 인물과 함께 춤을 이끌어 나가는 역할을 합니다.
- 핑크색 옷을 입고 모자를 쓴 인물: 빨간색 옷을 입은 인물과함께 춤을 이끌어 나가는 역할을 합니다.
- 검은색 옷을 입고 모자를 쓴 인물: 춤을 즐기는 모습을 보여줍니다.
- 흰색 옷을 입고 모자를 쓴 인물:춤을 즐기는 모습을 보여줍니다.
```

동영상 삭제하기

유튜브는 스트리밍 플랫폼이므로 영상을 다운로드하는 것을 원칙적으로 허용하지 않습니다. 영상을 다운로드하여 사용하려면 프리미엄 서비스를 이용해야 약관에 위배되지 않습니다. 물론 이 경우도 개인적인 용도로만 사용해야 합니다. 만일 다운로드한 동영상을 공개하거나 영리적인 목적으로 사용하면 저작권법에 위반될 수 있으니 이 점은 각별히 유의해야 합니다. 따라서 클라우드 스토리지는 물론 로컬의 동영상에 대해서도 인식이 완료되면 즉시 삭제하는 것이 좋습니다.

```
import os

if os.path.exists(file_path):
    os.remove(file_path)

uploaded_file.delete()
```

File API를 통해 업로드한 파일은 2일이 지나면 자동 삭제됩니다.

5.2. 유튜브 인식 파이프라인 만들기(feat. 소라 Sora)

각각 나누어 작업했던 유튜브 다운로드, 클라우드 스토리지 업로드, 동영상 인식, 동영상 삭제를 한 번에 처리하도록 구현하겠습니다.

1) 앞서 구현했던 유튜브 다운로드 코드를 함수로 만듭니다.

```
8  model = genai.GenerativeModel(model_name="gemini-1.5-flash")
9
10 url = "https://www.youtube.com/watch?v=7qQTyBW4uhI"
11
12 def download_youtube(url):
13     yt = YouTube(url) # YouTube 객체 생성
14     stream = yt.streams.get_highest_resolution() # 가장 높은 해상도의 스트림 선택
15     # 현재 디렉터리에 동영상 다운로드
16     file_path = stream.download(output_path="./videos")
17     print("Download complete!")
18     return file_path
```

2) 로컬과 File API로 생성한 파일을 삭제하는 함수를 만듭니다.

```python
def delete_file(file_path, uploaded_file):
    if os.path.exists(file_path):
        os.remove(file_path)
        uploaded_file.delete()
```

3) 영상 인식 메인 함수를 만듭니다. 영상 업로드 시간을 확보하기 위해 sleep(5) 를 중간에 실행하는 코드를 추가했습니다.

```python
import time
import IPython.display  # 동영상 플레이어 출력을 위해 추가

def recog_video(prompt, url, model):
    global file_path, uploaded_file # 이게있어야 delete_file 함수에서 변수를 사용가능
    file_path = download_youtube(url)
    uploaded_file = genai.upload_file(path=file_path)
    contents = [prompt, uploaded_file]
    time.sleep(5)
    responses = model.generate_content(contents, stream=True, request_options={"timeout": 60*2})

    IPython.display.display(IPython.display.Video(file_path, width=800 ,embed=True))
    for response in responses:
        print(response.text.strip(), end="")

```

4) 지금까지 작성한 함수들을 호출하면서 동영상 인식을 수행합니다. 인식 내용은 오픈AI 소라(Sora)의 '스타일리시한 도쿄 여성' 영상을 보고, 그 내용을 묘사한 후 감상평을 말하도록 했습니다.

```python
prompt = """
- 주인공과 영상의 배경을 소설처럼 디테일하게 묘사하세요.
- 주인공이 어느 도시에 있는지 말하고, 왜 그렇게 생각하는지 설명하세요.
- 만일 이 영상이 생성형 AI가 만들었다면, 어떤 부분이 가장 놀랍나요?
"""

recog_video(prompt, url, model)
delete_file(file_path, uploaded_file)
```

다음은 실행 결과입니다.

```
<IPython.core.display.Video object>
밤늦은 도쿄 거리. 비가 그친 후 습한 공기가도시의 불빛에 젖어 있었다. 네온사인과광고판의 빛은 마치 거리에 그림을 그리는 듯했다.

여자는 검은 가죽 재킷을멋지게 걸치고, 붉은 긴 원피스를 입고 있었다. 검은 부츠와 검은 핸드백은 그녀의 시크한 스타일을 더욱 돋보이게 했다. 선글라스를 낀 채, 그녀는  붉은 립스틱으로 강렬한 인상을 남겼다.도쿄라는 것을 직감할 수 있었던 것은, 거리의 분위기 때문이었다.

길거리에는 사람들이 분주히 오가고 있었다. 밤늦은 시간이었지만도쿄는 여전히 활기가 넘쳤다.

그녀는 젖은 거리를 걸으며 빛의 반사를 즐기고 있었다. 마치 거울처럼 빛나는 웅덩이 속에는 네온사인과 건물들의 모습이 뒤섞여몽환적인 분위기를 자아냈다.

이 영상은 생성형 AI가 만들었다면, 빛의 표현과 분위기가 놀라웠다. 특히 빗물에 반사된 네온사인의 빛이 실제처럼 자연스럽게 표현되어 마치 실제 도쿄 거리를 걷고 있는 듯한 착각을 불러일으켰다.

마치 영화 속 한 장면처럼, 그녀의 모습은 밤늦은 도쿄 거리의 화려함과 어둠 속에묻힌 고독함을 동시에 보여주는 듯했다.Traceback (most recent call last):
```