



AI 개발자 트랙 미니프로젝트 3차

AI 강사 Agent 구축

AI 9반 17조



목차

- 01 모델 간략 소개
- 02 고객의 요구사항
- 03 워크플로우 소개
- 04 기대효과
- 05 참고

모델 간략 소개

PPT → 강의영상 자동 제작

1단계: 스크립트 생성

생성된 스크립트

'1단계: 스크립트 생성' 버튼을 눌러주세요...

수정사항

'수정 내용을 입력해주세요'

확인(수정X)
수정

생성된 스크립트

오늘은 ViT, 즉 Vision Transformer에 대해 알아보겠습니다. 먼저, CNN, 즉 합성곱 신경망의 한계를 살펴보면, CNN은 주로 지역적인 패턴을 학습하는 데 강점을 가지고 있지만, 전역 문맥을 이해하는 데에는 약한 점이 있습니다. 수용 영역이 깊어질수록 전역 정보를 처리할 수 있는 능력이 커지긴 하지만, 여전히 유연성이 부족하다는 것이 문제입니다. 반면, Transformer는 Self-Attention 메커니즘을 통해 전역 정보를 효과적으로 학습할 수 있습니다. 이는 자연어 처리 분야에서도 혁신적인 성과를 이끌어냈죠.

이제 ViT의 핵심 개념으로 넘어가겠습니다. ViT는 이미지를 여러 개의 패치로 나누어, 이를 마치 텍스트의 단어처럼 처리합니다. 각 패치를 벡터로 변환한 후, 위치 정보를 추가하여 Transformer의 입력으로 사용하고, Self-Attention을 통해 패치 간의 전역 관계를 학습합니다. 이를 통해 CNN의 한계를 극복할 수 있게 되었습니다.

표를 보면, ViT 외에도 CLIP, BLIP, LLaVA, Qwen2.5-omni, GPT-4o와 같은 다양한 모델들이 있습니다. ViT는 전역 문맥을 학습할 수 있도록 도와주고, CLIP은 이미지와 텍스트를 공통된 임베딩 공간으로 매핑하여 검색과 분류를 지원합니다. BLIP와 LLaVA는 이미지 설명 생성과 대화형 모델로 기능을 확장하며, Qwen2.5-omni와 GPT-4o는 다양한 입력과 출력을 통합적으로 관리하는 멀티모달 처리에 강점을 가지고 있습니다.

마지막으로, 이미지를 통해 CNN과 ViT의 수용 영역과 주의 집중 방식을 비교해보면, CNN은 국소적인 정보에 집중하는 반면, ViT는 전역 정보에 대한 주의를 강조하고 있습니다. ViT는 이러한 혁신적인 접근을 통해 CNN의 한계를 극복하고 있습니다. 감사합니다.

수정사항

'수정 내용을 입력해주세요'

확인(수정X)

수정

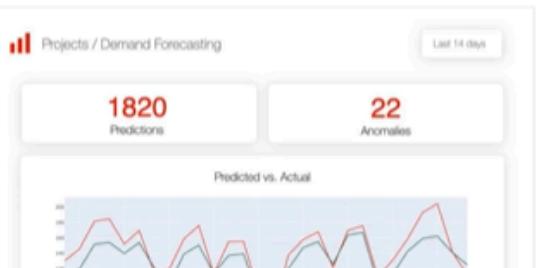
2단계: 영상 제작

최종 동영상 미리보기

모델 성능 모니터링

✓ 모델 모니터링

- 프로덕션 환경에서 ML 모델의 성능 및 동작을 지속적으로 추적, 분석, 평가하는 프로세스
- 필요성 : 모델은 배포 후에도 품질 저하, 데이터 문제, 사용 환경 변화 등의 위험에 노출됨
- 주요 리스크
 - 데이터/컨셉 드리프트
 - 데이터 품질 문제
 - 적대적 공격(예: prompt injection)
 - 연결된 앞 단계의 모델 오류 전파
- 모니터링 목표
 - 문제 조기 탐지
 - 근본 원인 분석
 - 모델 동작 이해 및 투명한 문서화



PPT 기반 자료를 자동으로 분석하여 스크립트-> 음성-> 영상으로 강의 생성

고객의 요구사항



사내교육

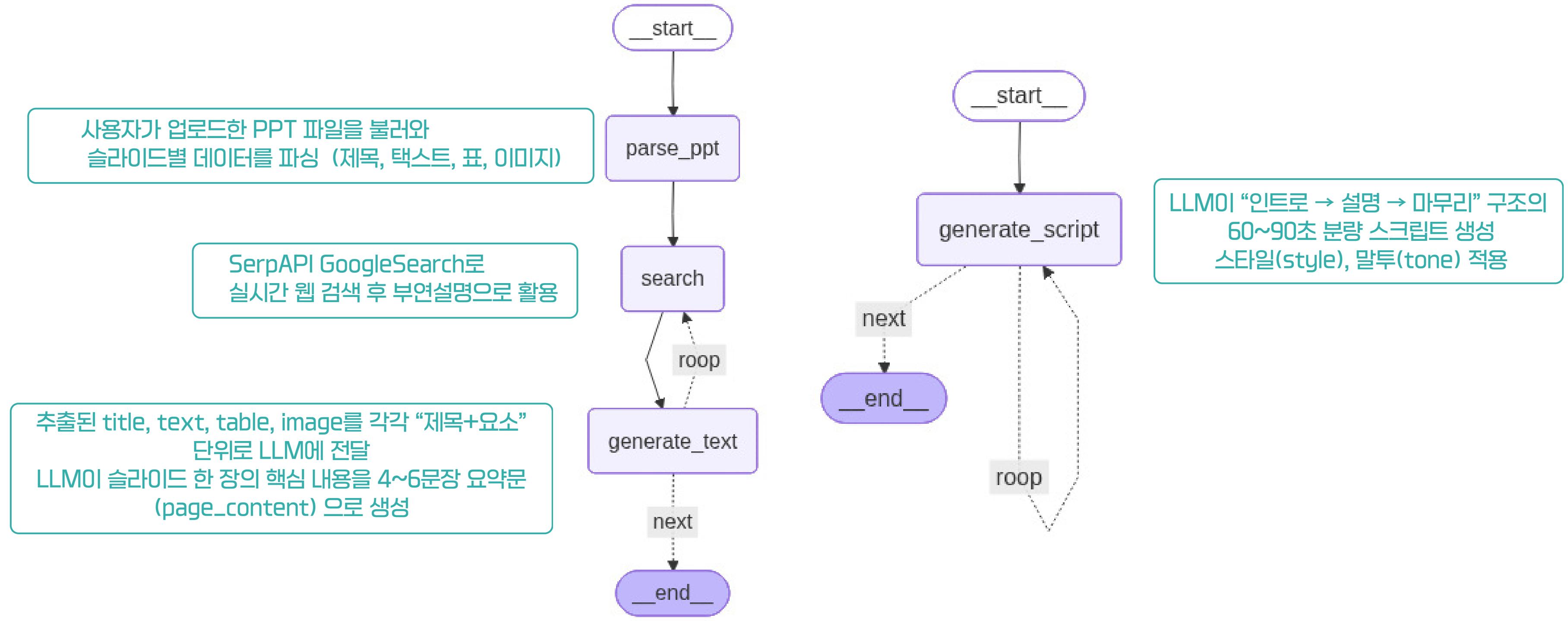
- 교육 시 강사를 충원해야하는 인력문제
- 기존 문서 자료를 영상으로 전환 활용 어려움
- 단순 문서나 슬라이드 배포로 학습 효과가 낮음



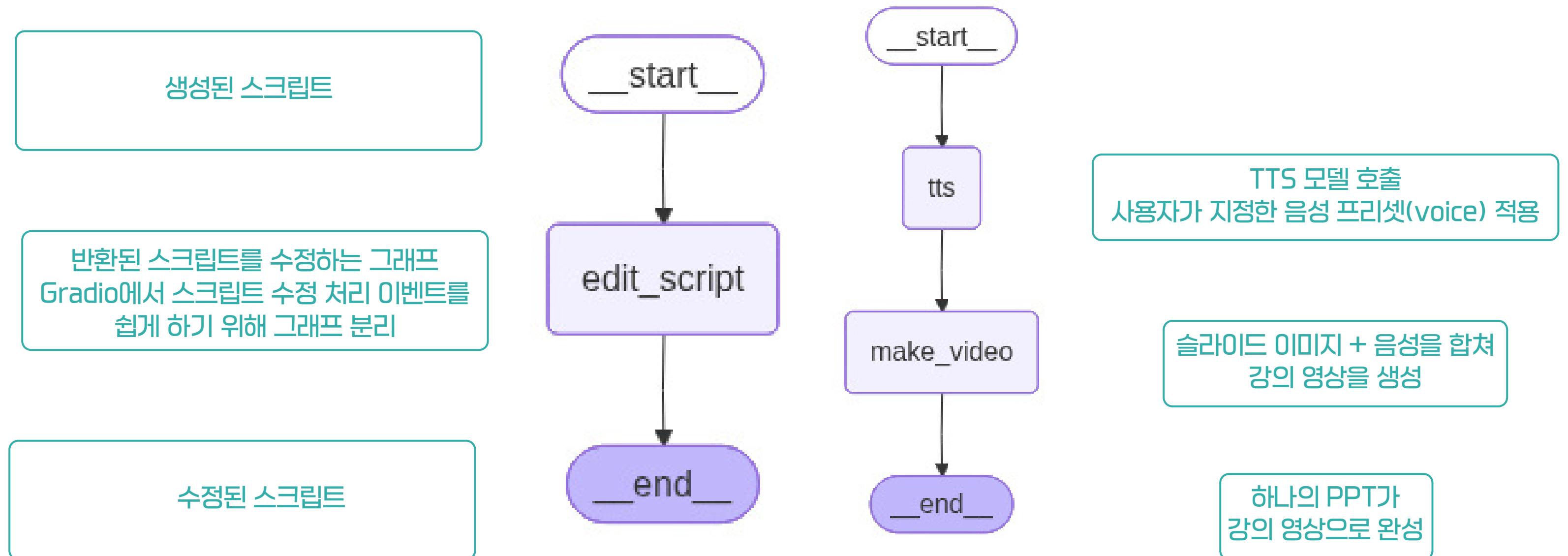
영상 제작의 어려움

- 외주 제작 시 비용 부담과 제작 기간 지연
- 영상 품질이 제작자에 따라 달라 일관성 부족
- 교육용 영상을 직접 녹화·편집 하는 비효율적 프로세스

워크플로우 소개



워크플로우 소개



워크플로우 소개 - 고도화 방안

TTS Voice

```

alloy
if voice in ["alloy", "aria", "verse", "shimmer", "coral", "sage", "tenor", "nova", "amber"]:
    for idx,scr in enumerate(script):
        # TTS 호출
        print(f"음성 생성 중... (voice: {voice})")
        resp = client.audio.speech.create(
            model=TTS_MODEL,
            voice=voice,
            input=scr
        )

        # MP3 파일 저장
        mp3_path = os.path.join(work_dir, f"narration{idx}.mp3")
        with open(mp3_path, "wb") as f:
            f.write(resp.content)

        state["audio"].append(mp3_path)

        # 오디오 길이 확인
        duration = ffprobe_duration(mp3_path)
        print(f"음성 파일 생성 완료: {mp3_path}")
        print(f"음성 길이: {duration:.2f}초")
        print("== 노드: TTS 변환 완료 ==\n")

```

✓ alloy

aria

verse

shimmer

**TTS 선택 가능 -> TypeCast TTS 모델, GPT TTS
GPT TTS의 한국어 음성 어색한 부분 개선**

**SerpAPI GoogleSearch로 실시간 웹 검색
-> 검색 결과를 배경지식으로 습득
-> 부연 설명의 근거로 활용**

생성된 스크립트 (수정 가능)

==== 슬라이드 1 ===

오늘은 모델 성능 모니터링에 대해 이야기해보겠습니다. 머신러닝 모델이 프로덕션 환경에 배포된 후, 그 성능을 지속적으로 추적하고 평가하는 과정이 바로 모델 성능 모니터링입니다. 이 과정은 모델이 배포되고 나서도 품질 저하나 데이터 문제, 환경 변화 등의 여러 위험에 노출될 수 있다는 점을 인식하고 있습니다.

주요 리스크에는 데이터 드리프트, 즉 데이터가 시간이 지남에 따라 변하는 현상, 품질 문제, 적대적 공격, 모델 오류가 발생할 수 있는 전파 등이 있습니다. 이러한 문제를 조기에 탐지하고 근본 원인을 분석하는 것이 모니터링의 중요한 목표인데요, 이를 통해 모델의 동작을 이해하고 문서화하는 과정도 포함됩니다.

슬라이드에 보이는 이미지에서는 최근 14일 동안의 예측 결과와 이상치를 보여주고 있습니다. 총 1820개의 예측이 이루어졌고, 그 중 22개의 이상치가 발견되었습니다. 그래프는 예측값과 실제값을 비교하여 직관적으로 부여하고 있습니다.

```

## SerpAPI 검색
results = serpapi_search_by_title(title)
# pprint(results['organic_results'])
# pprint(results['inline_videos'])

## 검색 결과를 AI강사의 배경지식으로 축적
# 시스템 프롬프트
sys_prompt = f'''당신은 많은 레퍼런스를 참조하여 인사이트를 도출하고, 요약 정리 우선 지금까지의 배경지식을 정리합니다. (아래 내용이 없는 경우, 생략합니다)
{background_knowledge}
'''

# 사용자 프롬프트
user_prompt = f"""다음 레퍼런스를 참조하여 인사이트 도출과 요약을 수행하고 핵심
{results}
"""

```

**생성된 스크립트 수정 가능
-> 사용자의 판단에 따라 내용 수정 가능**

AI Agent 도입 기대효과

교육적 기대효과

- ✓ 강의 준비 시간 단축
→ PPT 업로드만으로 자동 강의 생성
- ✓ 학습자 맞춤 강의 가능
→ 음성·톤·설명 깊이 조절 가능
- ✓ 피교육자 이해도 향상
→ 명확한 구조와 설명으로 이해도 향상

비즈니스/ 운영적 기대효과

- ✓ 대규모 강의 제작 가능, 시간과 인력 절약
→ 강사 투입 인력 및 제작비 절감
- ✓ 지속 가능한 교육 자산 구축
→ 한 번 만든 자료를 재활용 가능, 업데이트 시 자동 반영에 용이
- ✓ 교육 프로세스를 일괄 관리 가능
→ 강의 완주율, 재시청률 등 수치화 후 분석 가능

참고 - State 구조

```

# State 정의 및 초기화
class State(TypedDict, total=False): # total=False는 TypedDict에서 모든 키를 선택(optional)으로 취급

    # 입력/기본 - 기본초기화
    pptx_path: str      # PPT 경로 - ./ppt명 > 그래프 그리고 외부에서 선언한 뒤 그래프에 적용됨
    work_dir: str        # 작업파일 경로 - ./step1_output
    persona : Dict      # TTS 프롬프트 - USER_PROMPT
    slide_index: int     # PPT몇장 0,1,2,3 ....

    # 추출 산출물
    texts: List[List[str]]          # ppt의 텍스트 > [[["1. ~~","2. ~~"], ["1. ~~","2. ~~"]]]
    tables: List[ List[List[List[str]]]]  # 표 [[[표1의 상단],[표1의 2번째줄]], [[["표2..."]]]]
    images: List[ List[str]]         # [img1_path,img2_path....] > List[List[str]] 로?
    slide_image: List[str]          # [ppt 전체 이미지 경로1, 2....]
    slide_indices: List[int]         # 전체 슬라이드 인덱스 리스트
    slide_titles: List[str]          # 전체 슬라이드 제목 리스트
    background_knowledge: str       # serpapi 인사이트 도출 및 요약(배경지식) 축적

    # 생성 산출물
    page_content: List[str]    #현재 페이지의 요약 > List[str]로?
    script: List[str]           #현재 페이지 스크립트 > List[str]로?

    # 미디어 산출물
    audio: List[str]            # 오디오 경로 > List[str] 로?
    video_path: List[str]        #비디오 경로 > List[str]로?
    final_video : str

    # 미디어 생성 과정 산출물
    pptx_state: str              # 현재 처리 상태 ("파싱중", "텍스트생성중", "스크립트생성중", "음성변환중", "영상제작중", "완료")
    script_editing: bool          # 스크립트 편집 모드 여부

```

