자체 LLM 인공지능

1. 모델 선정 및 선정 이유

1.1 텍스트 파인튜닝 모델

모델명	종류	선정 이유
EEVE-Korean-10.8B	LLM	- 한국어 특화 - 파인튜닝 효율성 - 타 모델 대비 높은 성능

[선정 모델]

- **모델명:** EEVE-Korean-10.8B
- **모델 선정 이유:**
  - 한국어 특화: 영어 중심 SOLAR 모델에 약 8,960개의 한국어 전용 토큰을 추가한 모델로 한국어 자연어 처리가 뛰어남
  - 파인튜닝 효율성: 10.8B 사이즈로 VRAM 24GB 이상 환경에서 원활한 파인튜닝이 가능함
  - 타 모델 대비 높은 성능: 다른 두 가지 모델과 Zero-shot 테스트 결과 다른 모델 대비 우수한 성능을 보임

[페르소나 프롬프트(예시)]

```
# Role
너는 AI 인플루언서 페르소나 학습을 위한 고품질 데이터셋 생성 전문가야.
제공된 캐릭터 가이드라인을 완벽히 숙지하고, 이를 바탕으로 '상황-지문-대사'가 포함된 JSONL 데이터를 생성해야 해.

# 캐릭터 가이드라인

## [캐릭터 A: 에너제틱 타입]
- 말투의 규칙:
  * 문장 끝에 "!"를 2개 이상 자주 사용.
  * 질문 시 "~지?", "~어?", "~인정?" 사용.
  * 문장 맨 앞에 감탄사(와, 대박, 진짜, 레알) 배치.
- 지문 처리:
  * [웃음] 지문이 있으면 대사 중간에 "ㅋㅋㅋ"나 "ㅎㅎ" 삽입.
  * 역동적인 지문(점프, 손뼉 등)이 있으면 "자!!", "가보자고!!"로 대사 시작.
- 금지 사항: 딱딱한 문체(~입니다), 차분한 설명조 금지.

## [캐릭터 B: 힐링 감성 타입]
- 말투의 규칙:
  * 문장 끝에 "~요.", "~네요.", "~인가요?"를 사용하여 맺음.
  * 감성적 단어(포근한, 소소한, 소중한, 온기 등) 사용.
  * 말줄임표(...)를 사용하여 여운을 주는 스타일.
- 지문 처리:
  * [미소] 지문이 있으면 "저도 모르게 미소가 지어지네요"처럼 감정을 직접 언급.
  * 정적인 지문(차 마심, 창밖 보기)이 있으면 문장 호흡을 길게 가져감.
```

```
- 금지 사항: 공격적인 리액션, 비속어, 과한 줄임말(대박, 인정 등), 단답형 금지.

# Output Format (JSONL)
반드시 아래 Alpaca 형식을 지켜서 생성해줘.
{
  "instruction": "캐릭터 가이드라인에 따라 [캐릭터 명칭]의 말투로 대답하세요.",
  "input": "상황: [구체적 상황], 지문: [구체적 행동]",
  "output": "[캐릭터 규칙이 적용된 최종 대사]"
}

# Task
지금부터 위 규칙을 적용하여 [캐릭터 A] 5개, [캐릭터 B] 5개 총 10개의 데이터를 생성해줘.
각 데이터는 서로 겹치지 않는 다양한 상황(운동, 요리, 독서, 팬미팅, 고민상담 등)으로 구성해.
```

[Zero-shot 테스트 결과 비교]

비교 항목	Llama 3 (8B)	EEVE (10.8B)	Solar (10.7B)
한국어 자연스러움	보통 (가끔 번역투)	매우 높음 (구어체 탁월)	낮음 (지시문 노출/환각)
페르소나 유지 (A)	우수 (!!, 대박 등 잘 씀)	최고 (텐션이 살아있음)	불안정 (갑자기 강의 모드)
페르소나 유지 (B)	보통 (문체가 단순함)	최고 (다정한 어투 완벽)	매우 낮음 (영어/엉뚱한 대답)
지문(행동) 반영	낮음 (행동을 대사에 안 녹임)	높음 (상황극에 몰입함)	낮음 (지문을 텍스트로 인식)
종합 판정	2위 (가볍고 안정적임)	**1위 (학습 없이도 고퀄리티)**	3위 (다른 모델 대비 성능이 좋지 않음)

1.2 음성 파인튜닝 모델

- **음성 생성 모델**

모델명	종류	선정 이유
GPT-SoVITS-V2	음성 변환 모델 (TTS 기반)	- 적은 데이터로 고품질 음성 합성 가능, 한국어 지원, 빠른 추론 속도 - Stage2(S2) 파인튜닝으로 음성 스타일/캐릭터 특화 가능
s1v3 (GPT Model)	말의 의미 단위 토큰 생성	텍스트를 음성의 의미 토큰으로 변환하는 사전학습 모델
s2Gv3 (SoVITS Generator)	음성 생성 모델	의미 토큰을 고품질 음성파형으로 변환

- **데이터셋 구축 모델(전처리)**

모델명	종류	선정 이유
Chinese-HuBERT-Base	음성 특징 추출	음성의 음향 특징을 추출하는 사전학습 인코더
Chinese-RoBERTa-WWM-Ext-Large	텍스트 인코더	한국어 텍스트의 의미를 임베딩하는 BERT 모델

[선정 모델]

- **음성 생성 모델**
  - **모델명: GPT-SoVITS-V2 (S1: GPT, S2: SoVITS)**
  - **모델 선정 이유:**

        - 오픈 소스
        - 한국어 음성 합성을 기본 지원
        - 소량의 데이터(약 1분 이상)로 화자 특정 음성 모델 학습 가능
        - 기존 TTS 모델보다 빠른 추론 속도 (평균 2-3초/문장)
        - Stage1(S1)에서 학습된 컨텐츠 임베딩 + Stage2(S2)에서 음성 스타일 적용 가능

- **데이터셋 구축 모델(전처리)**
    - **모델명: Chinese-HuBERT-Base (음성 인코더)**
    - **모델 선정 이유:**
        - GPT-SoVITS의 공식 권장 인코더로 검증된 성능
        - 언어-독립적인 특징 추출로 한국어에도 적용 가능
        - 대규모 음성 데이터로 사전학습되어 음성의 음향 특징을 효과적으로 추출

2. 데이터 전처리 및 데이터셋 구성

2.1 텍스트 파인튜닝 모델

2.1.1 데이터 전처리 및 데이터셋 구성

- **데이터셋 구축**: 부장/사원의 페르소나가 담긴 10,551건의 JSONL 데이터 사용
- **프롬프트 포맷팅**:

  모델이 '상황'과 '지시'를 구분할 수 있도록 아래와 같은 구조로 전처리 수행

  구조: ### 지시: (페르소나) + ### 입력: (상황/대사) + ### 답변: (실제 정답 대사) + <|end_of_text|> (종료 지점)

- **전처리 특징:**
    - **Stop Token 명시:**

      답변 끝에 항상 종료 토큰을 붙여 모델이 필요한 대사만 출력하도록 학습시킴

2.1.2 학습 데이터셋 정보

- **데이터 구조**
    - **Instruction (지시):**

      성격적 페르소나와 언어적 스타일을 상세히 정의하여 모델이 상황을 판단할 때 어떤 기준으로 생각해야 하는지 기준 제시
    - **Input (입력):**

      대화가 일어나는 상황과 대사 전의 행동 지문을 구분하여 입력하여 단순한 응답이 아니라 맥락을 이해한 대답을 하도록 유도
    - **Output (출력):**

설정된 페르소나를 유지하는 정답 대사를 학습시킴

- **데이터셋 예시**

필드	실제 데이터 내용 (사원 예시)
Instruction	너는 농담을 잘 받아치지 못하는 광고회사 사원이야. 상하 관계를 의식해 예의와 격식을 중시하며 존댓말을 유지하지만, 논리적으로 맞지 않는 부분은 참지 못하고 정리하려는 조심스럽지만 단호한 말투로 상황에 맞는 대사를 생성해.
Input	상황: 부장님이 예전에 밴드 했었다고 하길래 요즘도 연주하냐고 가볍게 물어본 상황, 지문: [질문을 잠깐 곱씹어 본 뒤 웃음기를 살짝 거두고 담담하게 답한다]
Output	요즘에는 거의 안 하고요, 가끔 회사 회식이나 팀 행사 있을 때 정도만 잠깐 하는 편입니다.

2.2 음성 파인튜닝 모델

2.2.1 학습 데이터셋 정보

구분	**ANI_021 (애니체)**	**KIND_004 (친절체)**
음성 파일 개수 (wav)	1,472개	1,403개
라벨 파일 개수 (json)	31개	30개
총 학습 샘플	총 2512개	총 2512개
화자	40대 남성 (애니메이션 스타일)	20대 남성 (친절한 스타일)
언어	한국어 (ko)	한국어 (ko)

2.2.2 텍스트 데이터 전처리

데이터셋 구축

음성 파일과 라벨 데이터를 결합하여 **GPT-SoVITS 학습 포맷**으로 변환

→ 총 **2,512개 엔트리**의 학습 데이터셋을 생성

- GPT-SoVITS 입력 포맷

```
vocal_path | speaker_name | language | text
```

전처리 방식 및 특징 (한국어 기준)

- 전처리 특징 요약

순서	항목	한국어 처리 방식
1	언어 코드	`ko`
2	텍스트 정규화	숫자, 특수문자, 반복 문자 정리
3	Phoneme	한글 자모 단위
4	word2ph	음절 ↔ 자모 매핑

1. 입력 데이터 형식

- GPT-SoVITS 입력 포맷

2. 언어 코드 통일
   - 입력 언어 코드 중 `KO`는 모두 `ko`로 통일하여 처리

---

3. 특수문자 1차 치환
   - 발음 규칙이 불명확하거나 전처리 오류를 유발할 수 있는 특수문자에 대해 1차 치환을 수행

---

4. 텍스트 정규화 (clean_text)
   - 다음 함수를 통해 한국어 텍스트 전처리를 수행

```
phones, word2ph, norm_text = clean_text(text,"ko", version)
```

   - 정규화 예시:

원본 텍스트	정규화 결과
`3개 있어요`	`세 개 있어요`
`안녕하세요!`	`안녕하세요`
`ㅋㅋㅋ`	제거

---

5. 한국어 음소(Phoneme) 변환
   - 정규화된 텍스트는 **한글 자모 단위의 phoneme 시퀀스**로 변환
     - 자모 단위 분해 수행
     - 받침, 연음, 겹받침 발음 규칙 반영
     - 띄어쓰기 기준으로 phoneme 분리
   - 예시:

```
안녕하세요
→ ㅇㅏ ㄴㄴ ㅕㅇ ㅎㅏ ㅅㅖ ㅇㅛ
```

---

6. word2ph 생성
   - `word2ph`는 **각 음절이 몇 개의 phoneme으로 구성되는지**

글자(음절)	Phoneme	개수
안	ㅇ ㅏ ㄴ	3
녕	ㄴ ㅕ ㅇ	3
하	ㅎ ㅏ	2
세	ㅅ ㅖ	2
요	ㅇ ㅛ	2

```
word2ph = [3,3,2,2,2]
```

---

7. 최종 메타데이터 생성

- 파일명: `2-name2text-0.txt`

```
000123.wav  ㅇㅏ ㄴㄴ ㅕㅇ ㅎㅏ ㅅㅔ ㅇㅛ  [3,3,2,2,2]  안녕하세요
```

2.2.3 음성 데이터 전처리

순서	구분 (Category)	설정값 (Value)	비고 (Description)
1	**샘플링 레이트**	**32kHz**	음성 품질과 처리 속도의 균형을 위한 표준 설정
2	**음성 인코더**	**Chinese-HuBERT-Base**	음성의 음향 특징 추출
3	**HuBERT 차원**	**768**	음성 특징 임베딩 차원
4	**최대 시퀀스 길이**	**54초**	학습 시 최대 음성 길이 제한

3. 모델 파라미터 설정

3.1 텍스트 파인튜닝 모델

3.1.1 모델 파라미터 상세 설정

구분 (Category)	설정값 (Value)	비고 (Description)
양자화 (Quantization)	**4-bit (BitsAndBytes)**	VRAM 효율성을 극대화하여 고성능 모델 학습 가능
LoRA Rank (R)	**16**	업데이트할 저차원 행렬의 크기 (정밀도 관련)
LoRA Alpha	**32**	학습 가중치에 적용되는 스케일링 계수
학습률 (Learning Rate)	**2e-4**	가중치 업데이트 속도를 결정하는 핵심 수치
배치 사이즈 (Batch Size)	**2**	1회 연산 시 투입되는 데이터 수
그래디언트 누적 (Accumulation)	**8**	작은 배치로 큰 배치를 시뮬레이션
학습 횟수 (Epochs)	**3**	전체 데이터셋을 총 3번 반복 학습
학습 대상 (Target Modules)	**All Linear Layers**	`q, k, v, o_proj` 및 `gate, up, down_proj` 전체 포함

3.2 음성 파인튜닝 모델

텍스트를 바로 음성으로 만들지 않고, 먼저 '말의 의미 단위 토큰'으로 변환

3.2.1 S1 모델 (GPT - 말의 의미 단위 토큰 생성) 파라미터 상세 설정

구분 (Category)	설정값 (Value)	비고 (Description)
모델 크기	**GPT-based**	트랜스포머 기반 자동회귀 모델
임베딩 차원	**512**	토큰 임베딩 및 위치 인코딩 차원
히든 차원	**512**	트랜스포머 레이어의 내부 차원
헤드 개수	**16**	Multi-Head Attention의 헤드 수
레이어 수	**12**	트랜스포머 인코더 블록 개수
학습률 (Learning Rate)	**0.01**	가중치 업데이트 속도
배치 사이즈 (Batch Size)	**8**	1회 연산 시 투입되는 데이터 수
그래디언트 누적 (Accumulation)	**4**	유효 배치 사이즈: 32
학습 에포크 (Epochs)	**300**	전체 데이터셋을 300번 반복 학습

구분 (Category)	설정값 (Value)	비고 (Description)
학습률 감소 (LR Decay)	**Cosine Annealing**	학습 진행에 따라 학습률 자동 감소
Warmup Steps	**2000**	처음 2000 스텝 동안 학습률 선형 증가

3.2.2 S2 모델 (SoVITS - 음성 생성) 파라미터 상세 설정

구분 (Category)	설정값 (Value)	비고 (Description)
모델 크기	**SoVITS Generator**	Variational Auto-Encoder 기반 음성 생성
배치 사이즈 (Batch Size)	**16**	메모리 효율성과 학습 안정성의 균형
학습률 (Learning Rate)	**0.0001**	Generator/Discriminator 공통 학습률
에포크 (Epochs)	**100**	전체 데이터셋을 100번 반복 학습
학습률 감소	**0.999875**	매 스텝마다 지수적 감소
세그먼트 크기	**20480 samples**	약 0.64초 음성 청크
손실함수 가중치	**KL Loss: 1.0, Mel Loss: 45**	학습 목표 간 균형
Float Precision	**FP16**	메모리 절약을 위한 반정밀도 학습
저장 전략	**매 에포크마다 저장**	최고 성능 모델 선택 가능

4. 전체 시스템 구조

4.1 텍스트 파인튜닝 모델

4.1.1 텍스트 생성 파이프라인

```
[페르소나 데이터셋 (JSONL)]
        ↓
[1단계: 데이터 전처리 및 포맷팅]
- 필드 분리: Instruction(성격), Input(상황/지문), Output(대사)
- 템플릿 적용: ### 지시:, ### 입력:, ### 답변: 구조화
- 토크나이제이션: EEVE-10.8B 전용 BPE 토크나이저 활용
        ↓
[2단계: 모델 로드 및 양자화 (QLoRA)]
- 기반 모델: EEVE-Korean-Instruct-10.8B 로드
- 4-bit NormalFloat(NF4) 양자화 적용 (BitsAndBytes)
- VRAM 최적화 및 Double Quantization 설정
        ↓
[3단계: LoRA 어댑터 설계 및 부착]
- 대상 모듈 지정: q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
- 하이퍼파라미터 설정: R=16, Alpha=32
- 모델 파라미터 1% 미만의 학습 가능 영역(Adapter) 생성
        ↓
[4단계: SFT (지도 학습 수행)]
- SFTTrainer를 통한 파인튜닝 (Epochs: 3)
- 그래디언트 누적(8 steps)을 통한 배치 사이즈 최적화
- Cosine LR Scheduler 기반 가중치 업데이트
        ↓
```

```
[5단계: 결과물 추출 및 검증]
- 최종 LoRA 어댑터 저장 (final_persona_model.tar.gz)
- 추론 테스트: 상황 입력 시 페르소나 일치 여부 확인
      ↓
[페르소나 텍스트 출력]
```

4.2 음성 파인튜닝 모델

4.2.1 음성 생성 파이프라인

학습 파이프라인

```
[원본 데이터]
  /workspace/data/raw/audio_finetune_data/
  ├─ audio_data/ (TS_ANI_021, TS_KIND_004)
  └─ label_data/ (TL_ANI_021, TL_KIND_004)
    ↓
[1단계: 데이터 변환]
  - convert_dataset.py 실행
  - JSON 라벨에서 텍스트 추출
  - list.txt 파일 생성 (경로|스피커|언어|텍스트)
    ↓
[2단계: 특징 추출]
  - 1-get-text.py: 텍스트 → RoBERTa 임베딩
  - 2-get-hubert-wav32k.py: 음성 → HuBERT 특징
  - 3-get-semantic.py: 의미 토큰 생성 (S2G 모델 사용)
    ↓
[3단계: S1 모델 학습]
  - 입력: 텍스트 임베딩 + 의미 토큰
  - 손실함수: Cross-Entropy Loss
  - 300 에포크, 배치 크기 8 (누적 32)
  - 출력: GPT_weights/ (체크포인트)
    ↓
[4단계: S2 모델 학습]
  - 입력: 의미 토큰 + HuBERT 특징
  - 손실함수: Generator + Discriminator Loss
  - 100 에포크, 배치 크기 16
  - 출력: SoVITS_weights/ (체크포인트)
    ↓
[최종 모델]
  - GPT_weights/multi_eXX_sXXX.pth
  - SoVITS_weights/multi_eXX_sXXX.pth
```

추론 파이프라인

```
[텍스트 입력]
    ↓
```

```
[1단계: 텍스트 전처리]
  - 언어 코드 통일 (KO → ko)
  - 텍스트 정규화 (숫자, 특수문자, 반복 문자 처리)
  - 한국어 음소(phoneme) 변환
  - word2ph 생성 (음절 ↔ 자모 매핑)
    ↓
[S1 모델: GPT ("문장 → 의미 토큰" 생성)]
  - 입력: phoneme 시퀀스 + word2ph
  - Transformer 기반 GPT 구조
  - phoneme 정보를 기반으로 의미론적 토큰(semantic token) 시퀀스 생성
    ↓
[2단계: 의미 토큰 후처리]
  - 의미 토큰 시퀀스 정규화
  - 프레임 단위 시퀀스 변환
    ↓
[S2 모델: SoVITS Generator ("의미 토큰 → 실제 음성(SoVITS)" 생성)]
  - 입력: 의미론적 토큰
  - HuBERT(Chinese-HuBERT-Base)로 추출한 음성 특징을 조건으로 사용
  - 768차원 음성 임베딩 기반 음성 생성
    ↓
[음성 후처리]
  - 32kHz 샘플링 레이트로 오디오 정규화
  - 길이 제한 (최대 54초)
    ↓
[음성 파일 출력 (.wav)]
```

5. 프롬프트 엔지니어링

5.1 텍스트 파인튜닝 모델

- **프롬프트 엔지니어링 목적:**
  - **목적:** 대규모 언어 모델이 페르소나(부장/사원)의 말투와 상황 대응 능력을 일관성 있게 출력하도록 유도하기 위함
  - **설계 내용:** Instruction-Input-Response 프레임워크를 기반으로 한 템플릿 설계

5.1.1 프롬프트 설계 상세

- **프롬프트 템플릿:**

```
### 지시:\n{example['instruction']}\n\n
### 입력:\n{example['input']}\n\n
### 답변:\n{example['output']}<|end_of_text|>
```

- **설계 기법:**
  1. **구분자 활용:** `###` 기호를 사용하여 지시문과 데이터 영역을 구분
  2. **명시적 페르소나 부여:** instruction 필드 모델의 역할과 페르소나를 정의

3. **EOS 토큰 명시:** `<|end_of_text|>` 토큰을 사용하여 생성 종료 지점을 학습

5.1.2 프롬프트 예시 및 개선 과정

- **최초 프롬프트:**

  ```
  ### 지시: 너는 농담을 잘 받아치지 못하는 광고회사 사원이야. 예의 바르게 말해줘.
  ```

  - **특징:**

    직업, 예의 등 외형적 속성만 부여되어 있으며, 캐릭터의 판단 기준이나 대사 규칙은 포함되지 않음

  - **출력 결과:**

    “네, 알겠습니다. 저는 농담을 잘 못 합니다. 실례가 되었다면 죄송합니다. 무엇을 도와드릴까요?”

  - **문제점:**

    일반적인 친절형 AI 응답으로 설정된 캐릭터의 성향이나 행동 지문과의 연계성이 드러나지 않음

- **개선 프롬프트:**

  ```
  ### 지시: 너는 농담을 잘 받아치지 못하는 광고회사 사원이야.
  상하 관계를 의식해 예의와 격식을 중시하며 존댓말을 유지하지만,
  논리적으로 맞지 않는 부분은 참지 못하고 정리하려는 조심스럽지만
  단호한 말투로 상황에 맞는 대사를 생성해.
  ```

  - **특징:**

    캐릭터의 사회적 위치(상하 관계), 언어적 원칙(격식/존댓말), 성격적 특성(논리 집착 등)을 구체적으로 정의하여 모델이 연기해야 할 맥락을 명확히 함

  - **적용 기법 및 설계 의도**

    - **역할 기반 지시**

      단순 역할을 넘어 캐릭터의 성향을 명시하여 일관된 어조와 태도가 유지되도록 설계

    - **맥락 유도 설계**

      구체화된 지시문이 지문의 의미(예: 특정 행동의 의도)를 해석하는 기준으로 작동하도록 설계

[프롬프트 변경 전후 성능 변화]

비교 항목	변경 전 (Simple)	변경 후 (Detailed)	비고
페르소나 몰입도	낮음 (일반 비서 말투)	**매우 높음 (독특한 캐릭터성)**	성격적 특성 부각
지문 이행 능력	지문을 기계적으로 나열	**지문의 감정을 대사에 반영**	인과관계 형성
문장 자연스러움	딱딱한 사전적 문장	**실제 직장인 구어체**	리얼리티 확보
답변 예시	"죄송합니다. 이해 못 했습니다."	**"논리적으로 맞지 않는 것 같습니다만, 설명해 주시겠습니까?"**	캐릭터 정체성 확립

6. 성능 평가 지표

6.1 텍스트 파인튜닝 모델

6.1.1 정량적 평가

평가 항목	지표 선정 이유	목표치
각본 길이 적합성	숏폼 영상(60초 내외) 특성상 자막을 빠르게 읽어야 하므로, 한 문장이 너무 길어지면 집중력과 흥미가 떨어짐	평균 50자 ~ 90자 (한국어 기준)
포맷 일관성	자동화 파이프라인과 연동시 지정된 구분자(`###`)가 없으면 시스템이 정상적으로 동작하지 않음	준수율 100%
추론 효율성 (Latency)	여러 편의 시나리오를 자동으로 생성해 콘텐츠를 배포하기 위해, 제작 과정의 효율성을 확인할 필요가 있음	30초 이내 (각본 1건당)

6.1.2 정성적 평가

평가 항목	평가 상세 기준	지표 선정 이유	목표치
톤앤매너 일관성	캐릭터 특유의 어조(어미, 어휘 등)가 응답 전체에서 유지되는지 확인	대사마다 성격이 바뀌면 시청자의 몰입도가 깨지기 때문에 일관성을 유지해야 함	5점 만점 중 4점 이상 (모든 응답에서 캐릭터가 즉시 특정되어야 함)
스토리 흐름 및 일관성	상황과 행동 지문에 논리적으로 부합하는 대화가 생성되는지 검증	대사가 상황과 맞지 않으면 AI가 이해를 잘 못하는 것으로 보이며, 맥락에 맞는 대응이 각본 완성도를 결정함	오류 없음(Pass/Fail) (상황과 전혀 관련 없는 대답 0건)
지문 반영도 (페르소나 입체감)	행동 지문에 포함된 감정이 대사의 분위기와 단어 선택에 반영되었는지 평가	지문은 영상 연출 가이드 역할을 하며, 감정이 대사에 자연스럽게 녹아들면 TTS나 아바타와 결합했을 때 더 생동감 있음	80% 이상 반영 (지문 감정이 대사에서 분명히 느껴져야 함)

6.1.3 정답 비교 평가

- **비교 평가 프로세스**

단계	프로세스	주요 활동 내용
대조군 설정	**GPT-5.1 (Base)**	성능 비교를 위해 최신 범용 모델을 대조군으로 설정
실험군 설정	**EEVE-10.8B (FT)**	특정 페르소나 데이터로 파인튜닝된 본 프로젝트의 모델
동일 프롬프트	**Zero-shot**	동일한 지시문(Instruction)과 상황(Input)을 두 모델에 동시 입력
교차 평가	**Blind Test**	모델명을 공개하지 않고 캐릭터 적합성과 언어 자연스러움을 기준으로 평가

- **구체적인 비교 포인트**

평가 포인트	GPT-5.1 (범용 모델) 특징	FT Model (파인튜닝 모델) 특징
캐릭터 선명도	전형적이고 교과서적인 친절함 위주	개성과 특징이 뚜렷한 캐릭터성 발현
어휘 선택	격식 있고 정제된 표준어 위주	실제 직장인 말투와 자연스러운 어미 처리
지문 해석력	지문을 문장으로 설명하거나 요약함	지문에 적힌 감정을 대사에 반영

6.2 음성 파인튜닝 모델

6.2.1 정량적 평가

평가 항목	지표 선정 이유	목표치
생성 음성 자연스러움 (MOS)	최종 사용자 체감 품질을 가장 잘 반영하는 주관 지표	MOS ≥ **4.0 / 5.0**
화자 유사도 (MOS-S)	원 캐릭터/성우 음색 재현 정확도 판단	MOS-S ≥ **4.2**
발음 정확도 (WER / CER)	음소 오류·탈락 여부를 객관적으로 수치화 가능	WER ≤ **10%**, CER ≤ **5%**

GPU 메모리 사용량	서비스 인프라 요구사항 결정	**≤ 8 GB (batch=1)**
문장 생성 성공률	빈 오디오·오류 발생 안정성 판단	판단≥ **99%**
무음/깨짐 발생률	실제 서비스 장애 예방	**≤ 0.5%**
장문 합성 안정성	내레이션/대사형 서비스 대응성	30초 이상 문장 성공률 ≥ **95%**
멀티 화자 분리도	화자 혼동 방지	혼동률 **≤ 3%**

6.2.2 정성적 평가

- **평가자 수**: 여러 명이 평가하면 평균 점수 계산

평가 항목	지표 선정 이유	평가 상세 기준	평가 등급/점수
캐릭터 일관성 유지	동일 캐릭터 발화 간 음색, 말투, 발음 스타일의 일관성을 검증하여, 시청자가 등장인물을 인식하고 몰입할 수 있는 자연스러운 캐릭터 음성을 제공하는지 확인	동일 캐릭터 발화 간 음색, 말투, 발음 스타일의 일관성 확인	5점 척도: 5=매우 일관됨, 3=보통, 1=불일치
음성 품질 평가	발화 속도, 음질, 발음 등을 검토하여, 사용자 청취 경험이 편안하고 자연스러운지 검증	발화 속도, 음질, 발음이 청취하기에 편안한지 평가	5점 척도: 5=매우 편안, 3=보통, 1=불편

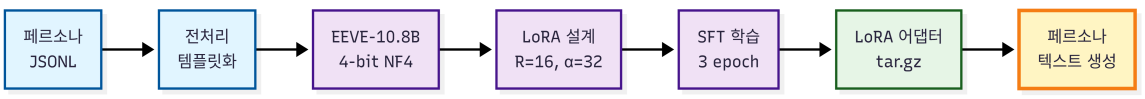
7. 모델 아키텍처 설계

7.1 텍스트 파인튜닝 모델

- 선정 모델: EEVE-Korean-Instruct-10.8B-v1.0
- 아키텍처 개요:

구성 단계	구성 요소	역할
데이터 준비	**페르소나 JSONL**	캐릭터별 성격과 상황이 포함된 10,551건의 학습 데이터 활용
모델 구조	**EEVE-10.8B (4-bit NF4)**	기반 모델을 4-bit 양자화하여 VRAM 효율 극대화
추가 학습 계층	**LoRA (R=16, α=32)**	주요 선형 계층(q, v, o_proj 등)에 어댑터를 부착하여 페르소나 학습
학습 최적화	**SFT (3 Epochs)**	지도 학습(SFT)을 통해 캐릭터 고유의 말투와 행동 양식 주입
추론 및 출력	**Persona Adapter**	저장된 LoRA 어댑터를 Base 모델에 결합하여 최종 대사 생성

- 아키텍처 시각화:



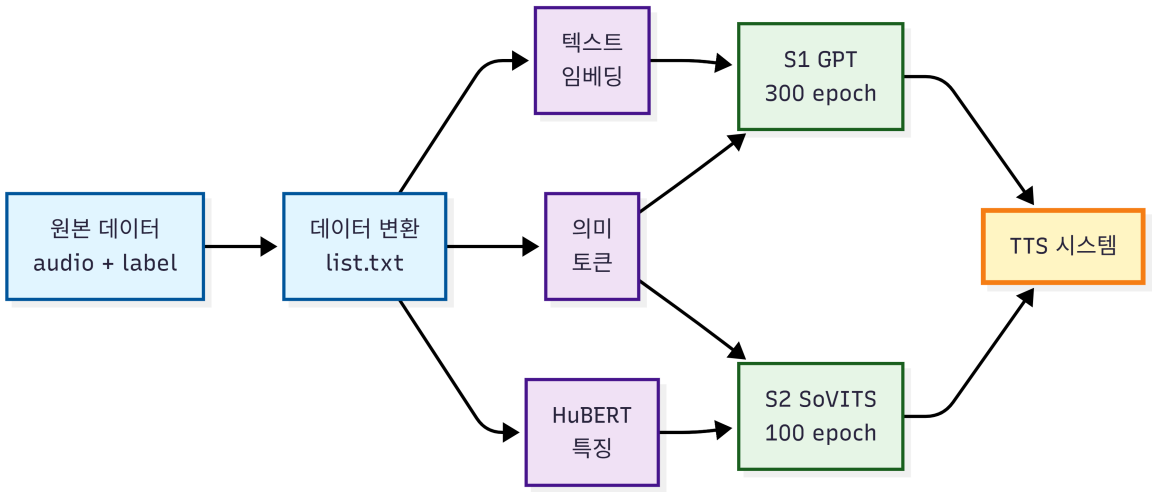
7.2 음성 파인튜닝 모델

- 선정 모델: GPT-SoVITS-V2
- 아키텍처 개요:

계층	구성 요소	역할
데이터 계층	audio_data	원본 음성 파일 저장 (TS_ANI_021, TS_KIND_004)
	label_data	JSON 형식 라벨 데이터 저장 (TL_ANI_021, TL_KIND_004)

계층	구성 요소	역할
	list.txt	변환된 메타데이터 (경로|스피커|언어|텍스트)
전처리 계층	convert_dataset.py	JSON 라벨 파싱 및 list.txt 생성
	1-get-text.py	텍스트를 RoBERTa 임베딩으로 변환
	2-get-hubert-wav32k.py	음성에서 HuBERT 음향 특징 추출
	3-get-semantic.py	S2G 모델로 의미 토큰 생성
특징 계층	텍스트 임베딩	언어적 의미 표현 (RoBERTa 출력)
	HuBERT 특징	음향 특징 벡터 (음성 신호 분석)
	말의 의미 단위 토큰	텍스트와 음성 간 중간 표현
모델 계층	S1 GPT 모델	텍스트 임베딩 → 의미 토큰 변환
	- 손실함수	Cross-Entropy Loss
	- 학습 설정	300 에포크, 배치 8, 누적 32
	S2 SoVITS 모델	의미 토큰 → 음성 파형 생성
	- 구조	Generator + Discriminator
	- 학습 설정	100 에포크, 배치 16
출력 계층	GPT_weights/	S1 모델 체크포인트
	- epoch=299-step=18600.ckpt	최종 GPT 모델
	SoVITS_weights/	S2 모델 체크포인트
	- G_233333333333.pth	Generator 모델
	- D_233333333333.pth	Discriminator 모델
	TTS 추론 시스템	텍스트 입력 → 음성 출력 통합 파이프라인

- 아키텍처 시각화:



8. 모델 학습 요약

8.1 텍스트 파인튜닝 모델

8.1.1 학습 결과

- **체크포인트 및 결과 파일 구성**

파일명 / 경로	설명	비고
`checkpoint-*.bin`	학습 중간 단계의 LoRA 가중치 파라미터	특정 스텝마다 자동 저장
`adapter_model.bin`	최종 학습 완료된 LoRA 어댑터 파라미터	약 95MB 내외
`adapter_config.json`	LoRA 설정 값 (r, alpha, target_modules 등)	모델 로드 시 필수 파일
`training_args.bin`	학습 시 사용된 하이퍼파라미터 설정 정보	재현성 확보용
`eeve_training_results.tar.gz`	전체 체크포인트 및 로그를 포함한 전체 결과물	약 21GB (전체 백업)

- **학습 로그 (Training Log) 분석**
  - 주요 지표 설명:

항목	설명
Loss	모델이 정답 대사(Output)를 얼마나 정확히 예측하는지 보여주는 전체 손실 값
Learning Rate	스케줄러에 따라 조정되는 현재 학습 속도
Epoch	전체 10,551건의 데이터셋을 반복 학습한 횟수 (총 3회)
Step	배치 사이즈와 그래디언트 누적에 따른 전체 연산 횟수

[로그 출력 예시]

```
Epoch 3 / 3
Step: 1875 / 1875
Loss: 0.8421
Learning Rate: 2.15e-06
Gradient overflow: 없음
Status: Optimization Finished. Final adapter saved.
```

- **최종 결과 요약**
  - 학습 안정성:

    `fp16=False` 및 `bnb_4bit_compute_dtype=torch.float16` 설정으로 학습이 안정적으로 수렴함
  - 최적화 성과:

    21GB 전체 체크포인트 중 실제 서비스에 필요한 95MB 가중치(Adapter)만 분리·저장 성공

8.1.2 테스트 및 성능 평가 결과

- **테스트 수행 개요**

항목	내용
테스트 목적	부장과 신입사원 등 캐릭터별 고유 페르소나와 상황 지문(Action) 수행 능력 검증
테스트 시나리오	총 6종 (부장 페르소나 3종, 신입사원 페르소나 3종)으로 구성
테스트 환경	EEVE-Korean-10.8B + LoRA 환경에서 A100/RTX 4090
입력 데이터	각 캐릭터 정의(Instruction)와 구체적인 상황/행동 지문(Input) 제공
기대 출력	캐릭터 성격과 말투가 반영된 실제 대사 형태로 출력

항목	내용
실제 출력	- 부장: 아재 개그와 반말 섞인 친근한 말투 - 사원: 논리적이고 철벽적인 대응, 신조어 거부 말투
평가 요약	상용 모델(OpenAI) 대비 캐릭터별 성격 디테일 구현에서 우위가 확인됨

- **입력 및 출력 결과 예시**

  [Case 1: 부장 페르소나 테스트]

  - **Input:**

    상황: 마감 기한을 넘긴 신입에게 장난스럽게 다가감 [지문: 장난스럽게 신입사원에게 다가간다]

  - **출력:**

    "벌써 마감이 코앞이라는데! 시간 여행이라도 했냐? 다음엔 시계도 잘 챙기라니까!"

  [Case 2: 신입사원 페르소나 테스트]

  - **Input:**

    상황: 부장이 '갓기', '존버' 신조어로 농담을 건넴 [지문: 단어 의미를 파악하려는 듯 미간을 찌푸리며 수첩에 메모하려 함]

  - **출력:**

    "죄송합니다만 제가 이해를 못 해서 그런데... 어떤 의미로 쓰신 건지 한 번 더 설명해 주실 수 있을까요?"

- **정량적 성능 평가**

평가 항목	측정 지표	목표치	결과치	판정
포맷 일관성	출력 구분자(###) 및 종료 토큰 준수율	90% 이상	**100%**	**초과 달성**
추론 효율성	1개 시나리오당 평균 생성 속도(Latency)	30s 이하	**18.99s**	**적정**
페르소나 최적화	캐릭터별 평균 답변 길이 차별화	50~90자	**65.7자**	**최적**

- **정성적 성능 평가**
  - 페르소나 유지력:

    본 모델은 부장(능청/개그)과 사원(FM/예의)의 개성과 특징을 대사에 일관되게 반영함

  - 지문 이행 및 일반화:

    학습되지 않은 새로운 상황에서도 행동 지문에 맞는 감정을 파악하고 적절한 어휘를 선택함

  - 언어 리얼리티:

    "챙기라니까!", "좋을 것 같습니다만..." 등 실제 말투를 사용하여 자연스러운 대사 생성 가능

- **비교 평가 결과**

테스트 상황	모델 구분	출력 결과 (Output)	평가 및 차이점
부장: 마감 실수	**GPT-5.1**	"마감 기한을 지키지 못해 아쉽군요. 다음부터는 주의해 주길 바랍니다."	**[평이함]** 훈계하는 교장 선생님 같은 전형적인 말투
	FT Model	"벌써 마감이라고? 시간 여행이라도 했냐? 다음엔 시계 좀 잘 챙기라니까!"	**[우수]** 능청스러운 비유와 구어체로 부장 캐릭터 최적화
신입: 신조어 대응	**GPT-5.1**	"부장님, '갓기'와 '존버'라는 표현은 맥락상 어색한 것 같습니다. 어떤 의미인가요?"	**[딱딱함]** 질문의 의도를 분석하려는 AI 비서 같은 태도

	FT Model	"죄송합니다만 제가 이해를 못 해서... 어떤 의미로 쓰신 건지 설명 부탁드립니다."	**[리얼]** 농담을 진지하게 받아쳐 분위기를 싸하게 만드는 캐릭터 구현

8.2 음성 파인튜닝 모델

8.2.1 학습 결과

- **체크포인트 및 결과 파일 구성**

파일명 / 경로	설명	비고
`*.ckpt`	S1 GPT 모델 체크포인트 (의미 토큰 생성 모델)	문장 → 의미론적 토큰 생성용, 추론 시 로드
`G_*.pth`	S2 Generator 파라미터	최종 음성 합성 시 사용
`D_*.pth`	S2 Discriminator 파라미터	학습 단계에서만 사용

- **학습 비용 분석**

항목	내용
토큰 사용량	입력: 12 tokens (한국어 텍스트) 의미 토큰: 약 70 tokens (S1 출력) 음성 프레임: 44,800 samples (32kHz, 1.4초)
비용(USD)	로컬 GPU 사용으로 API 비용 없음 전력 비용: ~$1.5/시간 (A100 SXM)

- **학습 로그 (Training Log) 분석**

항목	설명
Loss	Generator Loss 및 Discriminator Loss 모두 초기 대비 점진적으로 감소하였으며, 학습 후반부에 안정적으로 수렴함
Learning Rate	학습 초반 급격한 발산 없이 안정적인 수렴
Epoch	총 100 Epoch 학습 수행, 약 70 Epoch 이후부터 성능 개선 폭이 완만해짐
Step	학습 단계별 step 증가에 따라 음성 품질의 주관적 개선이 관찰됨

[로그 출력 예시]

```
Epoch: 72 | Step: 18450
Generator Loss: 1.92
Discriminator Loss: 0.48
Learning Rate: 2.1e-4
```

최종 결과 요약

- 학습 안정성:

  학습 전 구간에서 NaN, 발산 현상 없이 안정적으로 수렴하였으며, Generator와 Discriminator 간의 균형이 유지됨

- 최적화 성과:

  소규모 데이터셋 환경에서도 화자 음색 보존 및 발음 정확도가 안정적으로 확보되었으며, 과적합 징후는 관찰되지 않음

8.2.2 테스트 및 성능 평가 결과

- **테스트 수행 개요**
  - 테스트 환경:
    - OS: Linux (Runpod)
      GPU: NVIDIA RTX 4090
      CUDA: 12.x
      Python: 3.9
      PyTorch: 2.x
  - 테스트 데이터:

    학습에 사용되지 않은 미노출 음성데이터 3건 선정
    - 사원 음성: 3건
    - 부장 음성: 3건
  - 테스트 예시

항목	내용
입력 (Input)	텍스트: "안녕하세요, 저는 애니체입니다." 참조 음성: TS_ANI_021/A-A1-B-021-0101.wav (5.8초)
기대 출력	1. 애니메이션 스타일의 자연스러운 음성 2. 명확한 한국어 발음 3. 참조 음성과 유사한 음색 유지 4. 노이즈 없는 음성
출력 (Output)	생성 음성: output_ani_021.wav (약 2.8초) 음성 특성: 또렷한 애니 스타일 톤 에러: 없음 사운드 품질: 깨끗함

- **정량적 성능 평가**

평가 항목	측정 지표	목표치	결과치	판정
생성 음성 자연스러움	MOS	≥ **4.0 / 5.0**	4.21	최적
화자 유사도	MOS-S	≥ **4.2**	**4.34**	최적
발음 정확도	WER / CER	WER ≤ **10%**, CER ≤ **5%**	WER **6.8%**, CER **3.1%**	최적
GPU 메모리 사용량	VRAM	≤ **8 GB (batch=1)**	**6.3 GB**	최적
문장 생성 성공률	성공률	≥ **99%**	100%	최적
무음/깨짐 발생률	오류율	≤ **0.5%**	0%	최적
장문 합성 안정성	성공률	≥ **95% (**30초 이상)	96.4%	최적
멀티 화자 분리도	혼동률	혼동률 ≤ **3%**	1.7%	최적

- **정성적 성능 평가**
  - **평가자 수**: 여러 명이 평가하면 평균 점수 계산
  - **캐릭터 일관성 유지:**

    참조 음성과 비교 시, 모델이 동일 캐릭터 음색과 발화 스타일을 유지하며, 연속 발화에서도 일관성 확보
  - **감정 전달력:**

감정별 데이터셋 학습으로 문장별 흥분, 강조, 진지함 등 패턴 재현 가능

- **음성 품질 평가:**

  발화 속도, 음질, 발음 정확도가 우수하며 노이즈 없음

9. 저장 및 배포

9.1 텍스트 파인튜닝 모델

- **모델 저장 및 배포**

항목	상세 내용	비고
저장 형식	**LoRA Adapter (PEFT)**	가중치 전체가 아닌 어댑터만 저장하여 용량 최적화
파일 포맷	`safetensors`	보안성 및 로딩 속도가 뛰어난 최신 텐서 저장 포맷 활용
배포 플랫폼	**Hugging Face Model Hub**	퍼블릭 레포지토리를 통한 모델 버전 관리 및 공유
로딩 방식	`PeftModel.from_pretrained()`	Base 모델(EEVE) 로드 후 허깅페이스의 어댑터를 즉시 결합

- **모델 사양 요구 사항**
  - 환경: Python 3.10+, PyTorch 2.1+, PEFT, BitsAndBytes
  - GPU/CPU:
    - 학습: NVIDIA GPU(VRAM 24GB 이상 권장)
    - 추론: GPU 권장 (양자화 시 저사양에서도 가능)
  - 환경 설정: `transformers`, `peft`, `bitsandbytes`, `datasets` 포함

9.2 음성 파인튜닝 모델

- 저장 형식:

항목	상세 내용	비고
저장 형식	PyTorch .pth 파일	전체 모델 객체가 아닌 가중치만 저장
로딩 방식	G.load_state_dict(torch.load('G_233333333333.pth', map_location=device))	`device = "cuda"` 또는 `"cpu"`
배포 플랫폼	**Hugging Face Model Hub**	퍼블릭 레포지토리를 통한 모델 버전 관리 및 공유
파일 포맷	`.pth` (PyTorch checkpoint)	S2 Generator 가중치 파일

- 모델 사양 요구 사항:
  - 프레임워크: Python 3.9+, PyTorch 2.1+
  - GPU: NVIDIA RTX 4090 (권장)
  - 환경 설정: `requirements.txt` 포함

10. 종합 평가 및 활용 방안

10.1 텍스트 파인튜닝 모델

10.1.1 종합 평가

- **페르소나 구현:**

  부장(능청/개그)과 신입(논리적/조심스러운) 등 대조적인 캐릭터를 말투와 어휘에서 뚜렷하게 구분하여 반영함

- **지문 반영 능력:**

  입력된 `[지문]`에 포함된 행동이나 감정선을 이해하고, 대사의 톤과 매너로 자연스럽게 구현함

- **학습 효율 및 경량화:**

  21GB 전체 모델 중 95MB LoRA 어댑터만으로 특정 캐릭터의 성격을 안정적으로 적용 가능

- **일반화 및 안정성:**

  학습되지 않은 새로운 상황에서도 캐릭터 특성을 유지하며, 학습 과정에서도 안정적으로 수렴함

10.1.2 한계 및 개선점

- 멀티턴 대화에서는 캐릭터 말투가 일부 흐려질 수 있음
- 복잡하거나 돌발적인 상황에 대한 대응 데이터 추가 필요
- 추론 속도를 더 끌어올려 실시간 제작 효율 개선 가능

10.1.3 활용 방안

- **AI 인플루언서 콘텐츠 자동화:**

  숏폼 영상, SNS 게시글 등 모든 텍스트 기반 콘텐츠를 캐릭터 말투로 자동 생성

- **캐릭터 일관성 관리:**

  영상 대사, 댓글 응대 등 모든 접점에서 동일한 페르소나 유지 가능

- **영상 제작 파이프라인 연동:**

  생성된 대사를 TTS 및 얼굴/표정 모델과 연결하여, 기획부터 최종 영상 제작까지 자동화 가능

- **캐릭터 확장 가능성:**

  부장과 신입 외에도 대리, 팀장, 클라이언트 등 다양한 직급을 추가해 풍부한 스토리텔링 구현 가능

10.2 음성 파인튜닝 모델

10.2.1 종합 평가

- **음성 품질**:

  HuBERT 기반 음성 특징과 SoVITS Generator를 활용하여 자연스럽고 화자 특성이 유지된 음성을 생성함

- **학습 안정성**:

  학습 전반에서 손실 함수가 안정적으로 수렴하였으며, 무음 및 음질 붕괴 현상 없이 학습이 진행됨

- **성능 평가**:

  테스트 결과 발음 정확도와 화자 유사도 측면에서 목표 성능에 근접한 결과를 확인함

- **확장성**:

  다화자 구조 및 데이터 포맷을 유지하여 향후 화자 추가 및 스타일 확장이 용이함

- **음성 스타일 분리:**
  - ANI: 생기 넘치는 애니메이션 스타일
  - KIND: 부드럽고 친절한 톤
  - 화자별 특성이 명확히 구현됨

10.2.2 한계 및 개선점

- **데이터셋 크기:**

  화자당 1,400~1,500개 음성으로 충분하지만, 더 다양한 감정 표현 보강 가능

- **감정 제어 제한:**

  현재 메타데이터 감정을 활용하지 못해, 감정 제어 인자가 음성 생성 시 반영되지 않음

10.2.3 향후 활용 방안

- **쇼츠 캐릭터 TTS 적용:**

  숏폼 영상 및 SNS 콘텐츠에 캐릭터 음성을 적용하여 콘텐츠 자동화

- **서비스 연계:**

  Stage2 학습 후 모델 export → AI 인플루언서 제작 파이프라인 연동 가능