




# 2024년도 동계 SCI 인공지능 부트캠프

하모니



강연주  
박울  
진민찬  
최승훈

# contents

- 1.역할 분담
- 2.목표
- 3.기존 음성인식기 파악
- 4.실험 진행 순서
- 5.실험
  - Stage1- 데이터셋, 이중 전사 표기 선택
  - Stage2- 속도증강
  - Stage5- 토큰 list 변경
  - Stage6, 7, 8- 언어모델 학습
  - Stage10, 11- asr 모델 학습
  - Stage12,13- 실제 모델 디코딩
6. 음성인식 서비스 제안-음성 chatGPT
7. 결론



강연주-발표, 자료 준비



박 율-ppt 제작, 자료 준비

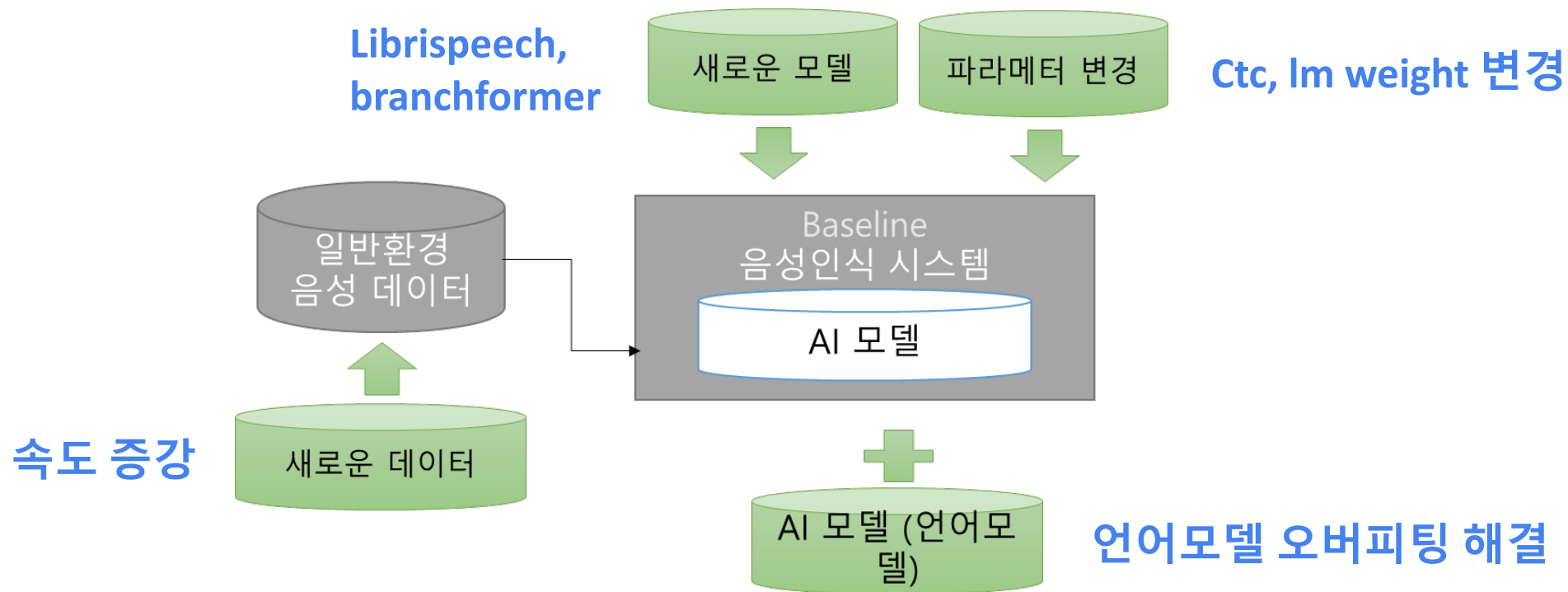


진민찬-데이터 전처리, 음성인식 서비스 개발



최승훈-데이터 전처리, 모델 학습, ppt 자료 준비, 음성인식 서비스 개발

- 현재 음성인식기의 문제점 파악을 통한 개선점 도출
- 음성인식 성능 향상
  - Err(Error rate reduction), CER 값의 감소



# | 기존 음성인식기 파악

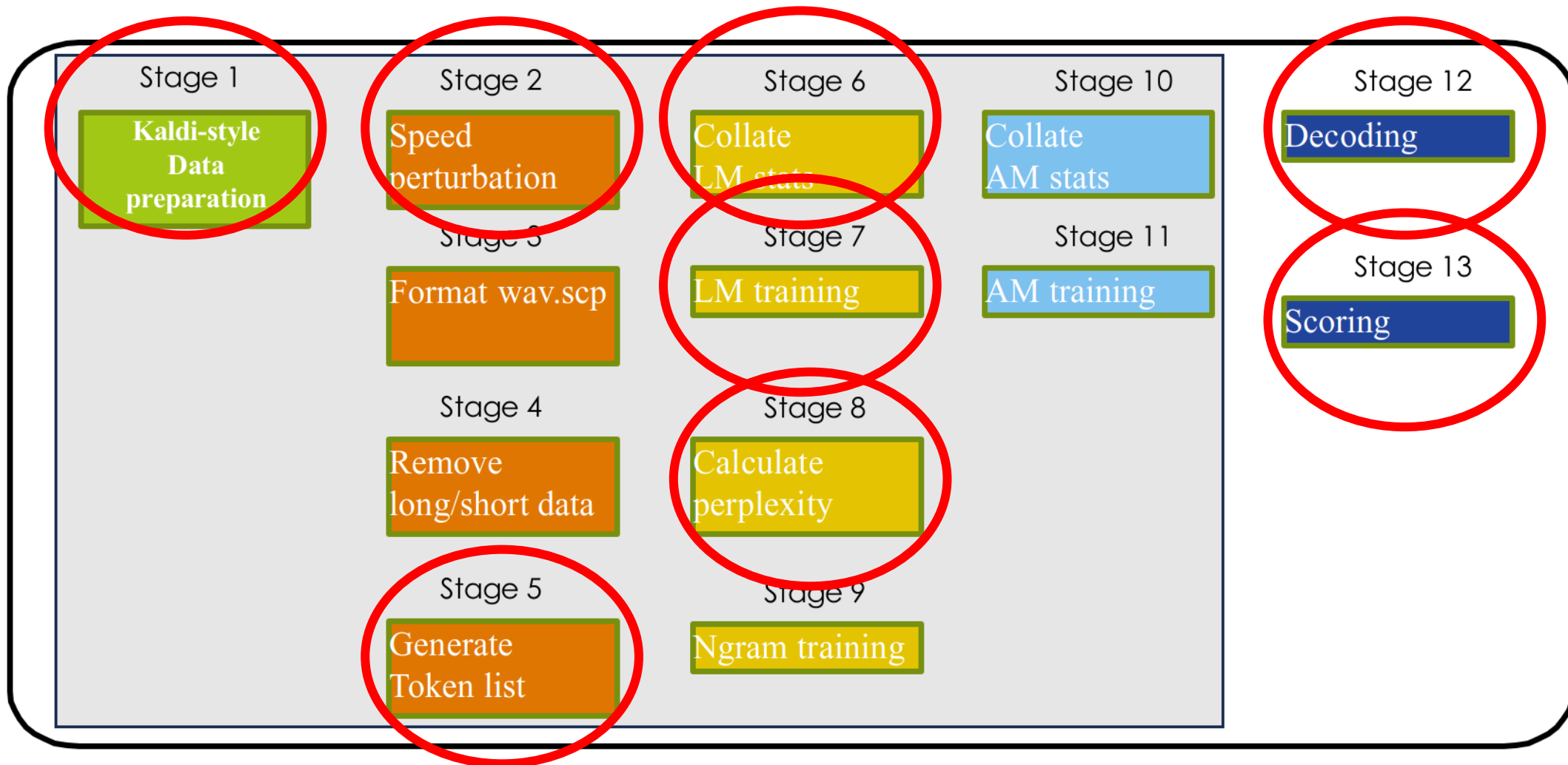
## 1. 레시피

- AIHUB의 한국어 음성 데이터로 학습하는 ksponspeech 사용
- LM을 사용하지 않는, 기본 unit이 음절인 음성인식 모델로 수정해서 사용
- **Conformer (CNN + transformer) 사용**

## 2. 데이터

- AIHUB에서 제공되는 **한국어 대학 강의 데이터셋 (4800 시간) 사용**
- 데이터셋 줄 수 : 170만줄(앞뒤 길이 조정 시 150만줄)
- 학습에 걸리는 시간 : 약 2일

# 실험 진행 순서



## Stage 1 데이터셋 크기

- 학습 1회 당 대략 2일로 너무 많은 시간이 소요됨  
--> 제한된 시간 안에 많은 실험을 할 수 없음
- **다양한 실험**을 하기 위한 데이터셋 크기 조정 (head - \$total\_train\_size 추가)
- 7만줄/30만줄/20만줄 학습을 진행해보고 **성능과 학습시간의 trade-off**를 판단
- 우수한 실험결과만 원래 데이터(150만줄)에 적용

```
filter_scp.pl --exclude -f 1 ${targetdir}/dev/wav.scp $train_wavscp \  
| head -$total_train_size | sort > ${targetdir}/train/wav.scp
```

```
filter_scp.pl --exclude -f 1 ${targetdir}/dev/wav.scp $train_text \  
| head -$total_train_size | sort > ${targetdir}/train/text
```

## Stage 1 데이터셋 크기

dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err	Time
7만 줄	6256	133054	35.1	49.4	15.5	13.0	77.9	99.4	2h
20만 줄	6256	133054	77.7	14.7	7.6	5.4	27.7	90.2	6h
30만 줄	6256	133054	83.3	10.2	6.4	4.5	21.2	82.4	8h
150만 줄	6256	133054	92.1	4.0	3.9	4.5	12.4	63.7	2d

- 20만 줄 데이터셋이 시간 대비 Err이 낮음  
➔ 20만 줄 데이터셋에서 실험을 진행한 뒤에 원래 데이터셋 (150만 줄)에서 최종 학습을 진행



## Stage 1 이중전사 표기 선택

- "외국어 문자/소리나는 대로의 한글" 2가지 형태로 전사되어 있음  
--> 이중 전사로 오류값 높임
- **이중전사**에 대해서 텍스트 전처리가 필요함
- **외국어 문자 vs 한글 선택 실험 진행**

예) (KBS)/(케이비에스), (MBC)/(엠비씨), (AT&T)/(에이티앤티), (ETRI)/(에트리), (ETRI)/(이티알아이), (OPEC)/(오펙), (OPEC)/(오피씨), (FIFA)/(피파) 등

1.5.2. 우리말로 표기하여 자연스러운 것은 한글로 표기한다. 애매한 경우도 한글로 표기한다.

예) 뉴욕, 시카고, 파티, 버스, 핸드폰, 모바일, 인터넷, 호텔 등등

## Stage 1 이중 전사 표기 선택

dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
비교군 20만 줄	6256	133054	77.7	14.7	7.6	5.4	27.7	90.2
외국어 선택	6256	133054	76.8	17.1	6.1	5.0	28.1	91.7
한글 선택	6256	133054	77.3	16.5	6.2	4.5	<b>27.2</b>	91.8

- 한글 선택시 Err 감소 --> 성능 향상

## Stage 1 전사 처리

## • 외국어 선택

Speaker sentences5517: eng\_elec\_c06787\_u00262  
 id: (eng\_elec\_c06787\_u00262-eng\_elec\_c06787\_u00262)

Scores: (#C #S #D #I) 17 5 6 0

REF: 그 다음에 <space> P R O J E C T I O N <space> 시 키 면 <space> 여 기 <space> 보 이 시 죠 ?

HYP: 그 다음에 <space> \* \* \* \* \* 프 로 젝 션 <space> 시 키 면 <space> 여 기 <space> 보 이 시 죠 .

Eval: D D D D D

D S S S S

S

**C(Correct):** 올바르게 인식된 음소의 수  
**S(Substitution):** 잘못 대체된 음소의 수  
**D(Deletion):** 누락된 음소의 수  
**I(Insertion):** 삽입된 음소의 수

## • 한글 선택

Speaker sentences5517: eng\_elec\_c06787\_u00262  
 id: (eng\_elec\_c06787\_u00262-eng\_elec\_c06787\_u00262)

Scores: (#C #S #D #I) 22 1 0 0

REF: 그 다음에 <space> 프 로 젝 션 을 <space> 시 키 면 <space> 여 기 <space> 보 이 시 죠 ?

HYP: 그 다음에 <space> 프 로 젝 션 을 <space> 시 키 면 <space> 여 기 <space> 보 이 시 죠 .

Eval:

S

## Stage 2 속도 증강

- 가정: 다양한 데이터 확보--> 성능 향상
- run.sh에서 **speed\_perturb\_factors** 옵션 추가
- 기존 데이터 속도를 0.9, 1.0, 1.1로 증강

```
./asr.sh \  
  --lang kr \  
  --train_set "${train_set}" \  
  --valid_set "${valid_set}" \  
  --test_sets "${test_sets}" \  
  --max_wav_duration 30 \  
  --speed_perturb_factors "0.9 1.0 1.1" \  
  --asr_config "${asr_config}" \  
  --inference_config "${inference_config}" \  
  --token_type "${token_type}" \  
  --lm_train_text "data/${train_set}/text" \  
  --use_ngram "false" \  
  --use_lm "true" \  
  "$@"
```

## Stage 2 속도 증강 (CER)

dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
비교군 20만 줄	6256	133054	77.7	14.7	7.6	5.4	27.7	90.2
속도 증강 없는 7만 줄	6256	133054	35.1	49.4	15.5	13.0	77.9	99.4
속도 증강한 7만 줄	6256	133054	74.7	16.4	8.9	5.4	30.7	92.4

- 성능 향상 크지만 약 3배의 시간 소요
- 150만 줄에 적용할 경우 6일 소요 -> 보류하기로 결정

## Stage 5 Token List 변경

- Tokenizer: 텍스트를 작은 단위인 토큰으로 분리하는 도구
- 토큰(token) 목록: 토큰화 과정에서 생성되거나 사용되는 토큰들의 집합
  - > 특정 언어나 데이터셋에 대한 사전(vocabulary)을 형성
  - > 텍스트 데이터를 숫자 형태로 변환하는 데 사용됨.
- char tokenizer의 **토큰(token) 목록에** 포함된 토큰들
  - > **모델이 학습 과정에서 인식하고 처리할 수 있는 모든 가능한 글자들을 나**  
**타냄**

1	이
2	는
3	.
4	에
5	그
6	가
7	다
8	고
9	서
10	하
11	어
12	요
13	을
14	기
15	지
16	게
17	로
18	있
19	니
20	면

## Stage 5 Token List 변경

	char	kobart	kocharElectra
The number of tokens	1926	30583	11364
tokenizer	char	char bpe	char
token_type	char	hugging_face	hugging_face

- Stage 5에서 hugging\_face token\_type을 지원하지만 LM이 char, bpe 두 종류만 지원
- LM 적용 위해서 Stage 5 진행 후
- token\_type = char로 다시 변경

```
./asr.sh \  
  --lang kr \  
  --train_set "${train_set}" \  
  --valid_set "${valid_set}" \  
  --test_sets "${test_sets}" \  
  --max_wav_duration 30 \  
  --speed_perturb_factors "0.9 1.0 1.1" \  
  --asr_config "${asr_config}" \  
  --inference_config "${inference_config}" \  
  --token_type "${token_type}" \  
  --lm_train_text "data/${train_set}/text" \  
  --use_ngram "false" \  
  --use_lm "true" \  
  "$@"
```

## Stage 5 Token List 변경 (CER)

dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
비교군 20만, char	6256	133054	77.7	14.7	7.6	5.4	27.7	90.2
kobart-base-v2	6256	133054	82.9	10.3	6.8	4.9	<u>22.0</u>	82.3
KoCharELECTRA	6256	133054	84.4	9.3	6.3	4.7	<u>20.3</u>	80.0

- kobart-base-v2, KoCharELECTRA 모두 높은 성능 향상
- 성능 향상 가설: Hugging Face 토큰 목록 사용으로 인한 성능 향상은,  
기본(char) 목록에 없는 발음 가능한 글자들이 포함되었기 때문일 것이다



## Stage 5 Token List 변경

- kocharElectra 토큰 목록에는 있지만  
기본 토큰 목록에는 없는 토큰 목록을 출력
- 실제로 사용하지 않는 **희귀 글자만** 존재  
-> 희귀 글자의 유무가  
음성인식 성능에 큰 영향 없음

```
def read_tokens_from_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        tokens = set(file.read().splitlines())
    return tokens

tokens_set_1 = read_tokens_from_file("tokens_char.txt")
tokens_set_2 = read_tokens_from_file("tokens_charElectra.txt")

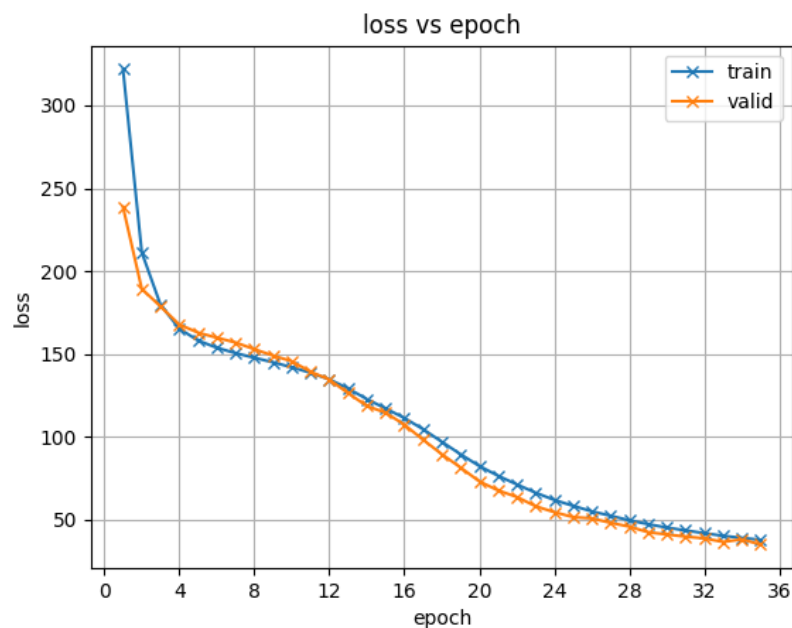
missing_in_token = tokens_set_2 - tokens_set_1

print(missing_in_token)
```

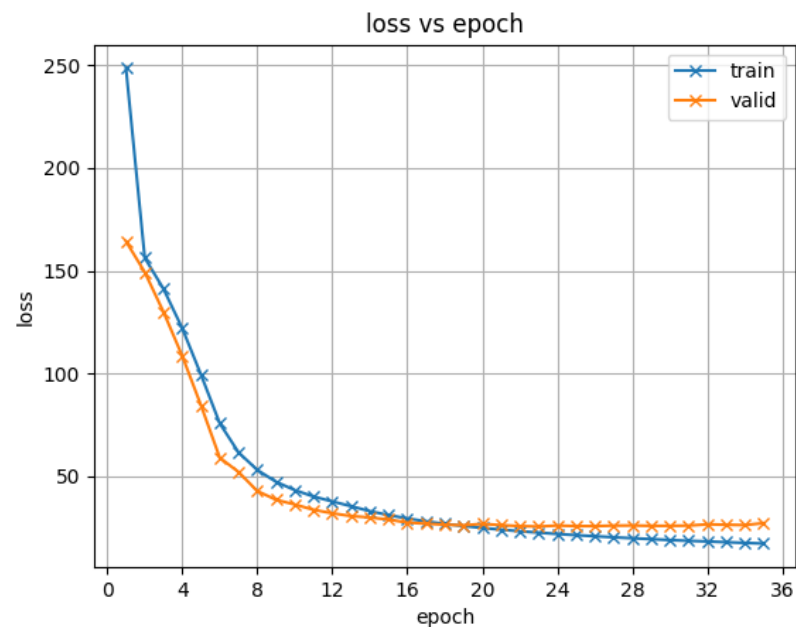
```
{',', '쪽', '뽕', '쿡', '똥', '켠', '긱', '틀',
'괘', '젯', '땃', '클', '덥', '괏', '뎡', '괘',
'릎', '윙', '괘', '랏', '등', '땃', '뎡', '뽕',
'괘', '색', '쟁', '뽕', '넉', '취', '윙', '왓',
'삽', '썩', '썩', '썩', '뽕', '괘', '뎡', '땃',
'땃', '땃', '괘', '넉', '틀', '뽕', '괘', '괘',
'민', '뽕', '괘', '뽕', '괘', '괘', '윙', '뎡',
'괘', '괘', '괘', '넉', '온', '괘', '괘', '뎡',
'괘', '괘', '괘', '괘', '괘', '괘', '괘', '괘',
'괘', '괘', '괘', '괘', '괘', '괘', '괘', '괘',
'윙', '괘', '괘', '괘', '괘', '괘', '괘', '괘', ... }
```

## Stage 5 Token List 변경

- 작은 데이터셋 (20만 줄) -> 큰 성능 개선
- 드문 음절을 추론하고 인식하는 데 큰 도움!!
- Hugging face 토큰 목록 사용시 loss 값 크게 감소



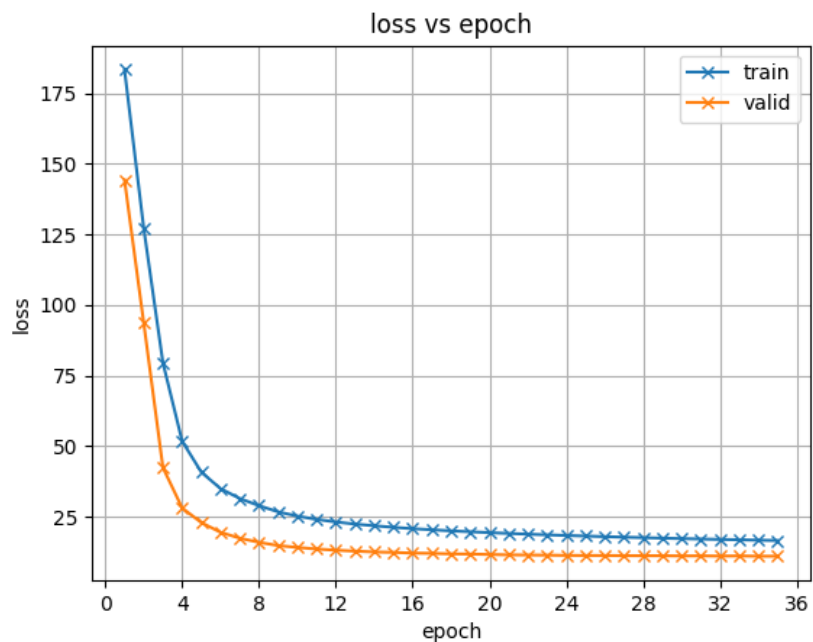
기본 토큰 목록 사용



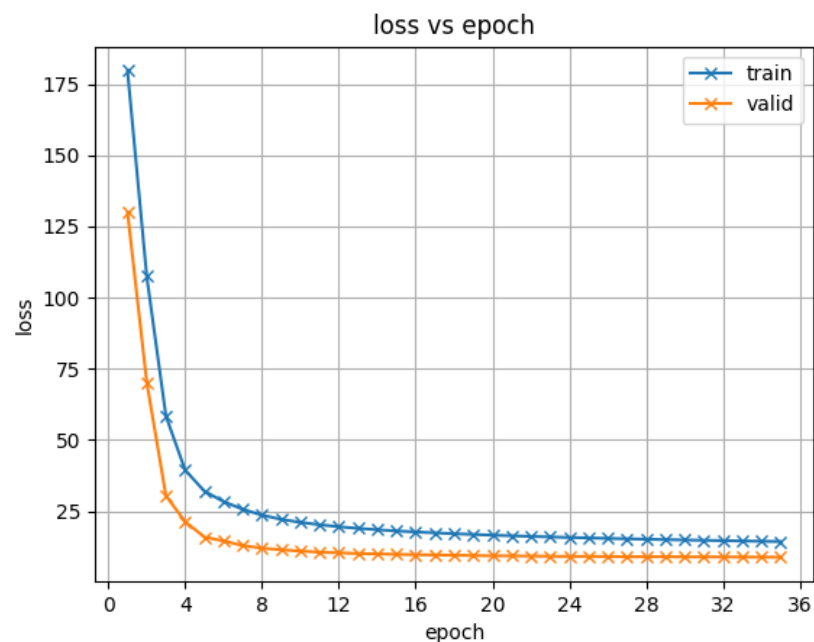
Hugging face 토큰 목록 사용

## Stage 5 Token List 변경

- 큰 데이터셋 (150만 줄) -> 작은 성능 개선
- 이미 다양한 음절들을 포함하고 있어 **성능에 큰 영향 없음**



기본 토큰 목록 사용



Hugging face 토큰 목록 사용

## Stage 6-8 LM (CER)

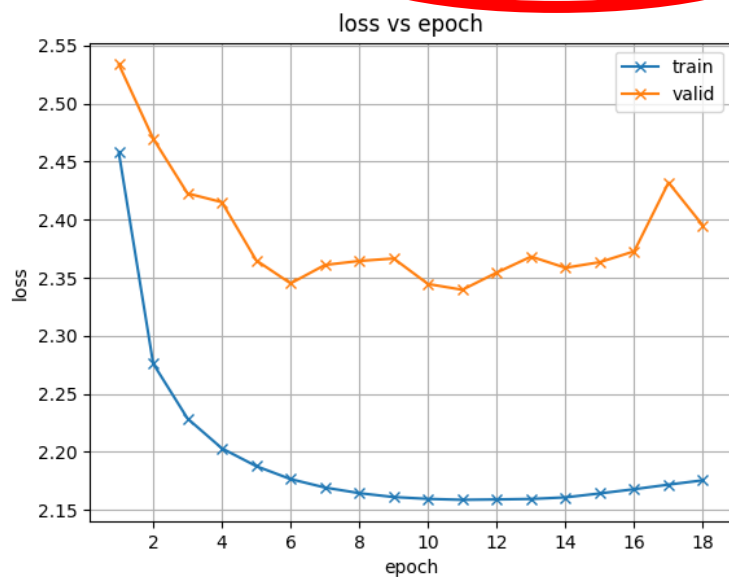
dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
비교군 20만	6256	133054	77.7	14.7	7.6	5.4	<b>27.7</b>	90.2
20만, LM	6256	133054	77.1	15.2	7.7	6.2	<b>29.0</b>	88.5

- LM 적용 시 CER(Err) 증가
- 150만 데이터셋 중 20만을 랜덤 추출  
20만에 Test dataset인 공학용 데이터 적게 포함.... 성능 저하처럼 보일 수 있음

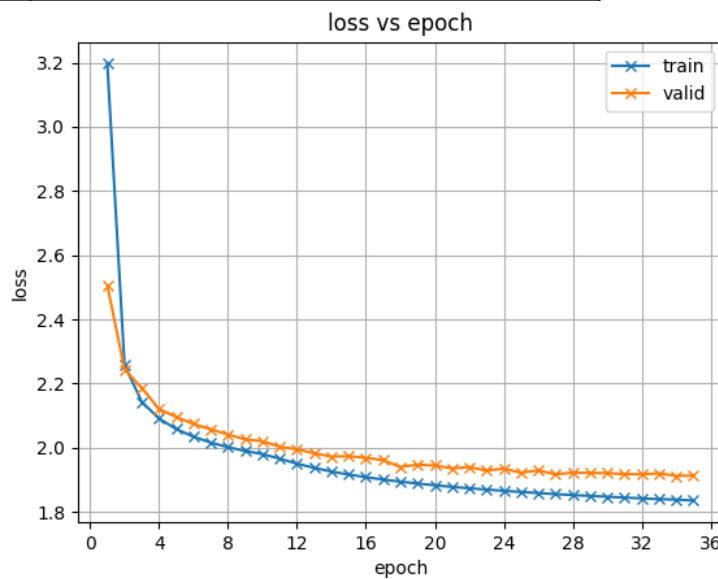
➔ 공학용(eng) 데이터셋(30만)에서 실험

## Stage 6-8 LM

LM	PPL
20만	13.51
30만 (eng)	8.80
150만	8.79
150만 dropout 0.3	6.96



LM(150만)



LM(150만 dropout 0.3)

- Overfitting!!
- Learning rate  
0.005  $\rightarrow$  0.004
- Dropout  
0.0  $\rightarrow$  0.3
- LM 150만 dropout 0.3 사용
- PPL: 문장의 길이로 정규화된 문장 확률의 역수
- PPL이 낮을수록 인식 성능이 좋다는 것을 의미함

## Stage 10-11 ASR Model (30만 공학용 데이터셋, CER)

dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
Conformer-ksponspeech	6256	133054	89.2	4.9	5.9	2.9	13.7	<u>68.3</u>
conformer-librispeech	6256	133054	89.3	5.0	5.7	3.0	13.7	69.0
branchformer	6256	133054	89.3	5.0	5.7	3.0	<u>13.6</u>	69.1

- Ksponspeech conformer에서 가장 낮은 S.Err, branchformer에서 가장 낮은 Err
- ➔ conformer-ksponspeech, branchformer epoch 10 이후 **valid loss** 비교해서 최종 모델 선택

- CER CTC : CTC 방법을 사용하여 계산된 CER
- CTC : Connectionist Temporal Classification의 약자, 음성 인식에서 시퀀스 학습 문제를 해결하는 방법

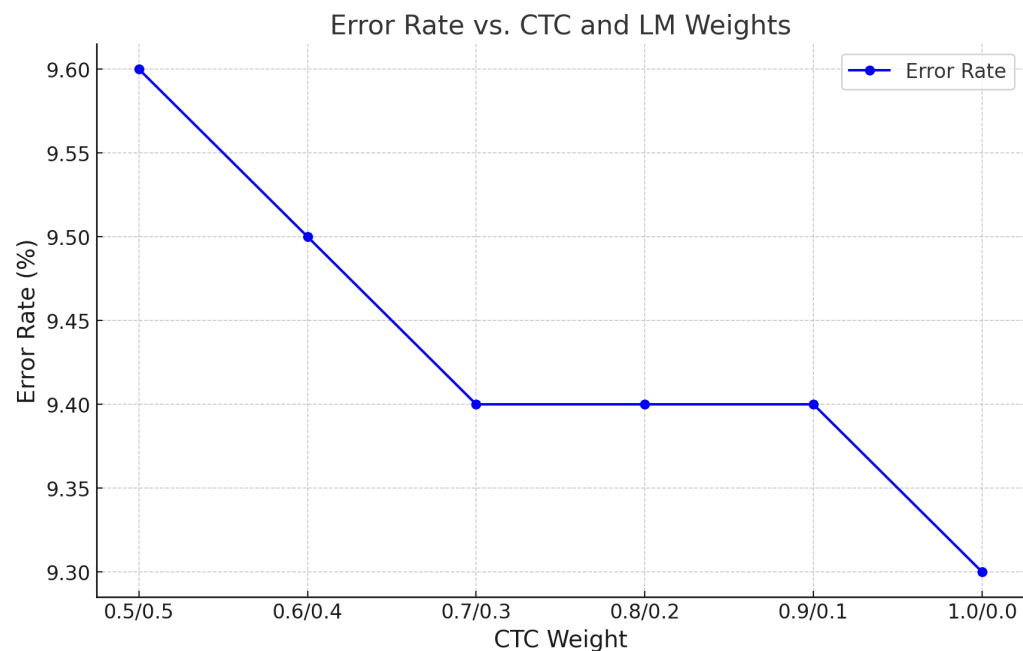
Stage 10-11 ASR Model (150만 원래 데이터셋, 10 epoch)

Metric	Conformer	Branchformer
Loss (Train)	21.023	33.814
Acc (Train)	0.870	0.529
Loss CTC (Valid)	15.267	30.701
<b>CER CTC (Valid)</b>	<b>0.105</b>	0.233
Loss ATT (Valid)	9.123	29.367
Acc (Valid)	0.937	0.838
<b>CER (Valid)</b>	<b>0.086</b>	0.213
WER (Valid)	0.235	0.487
Learning Rate	1.580e-04	2.471e-04

- conformer가 150만 데이터셋에서 epoch 대비 CER 값이 낮음 -> 성능이 좋음  
➔ conformer - ksponspeech 사용

## Stage 12-13 모델 디코딩 (150만 원래 데이터셋 CER)

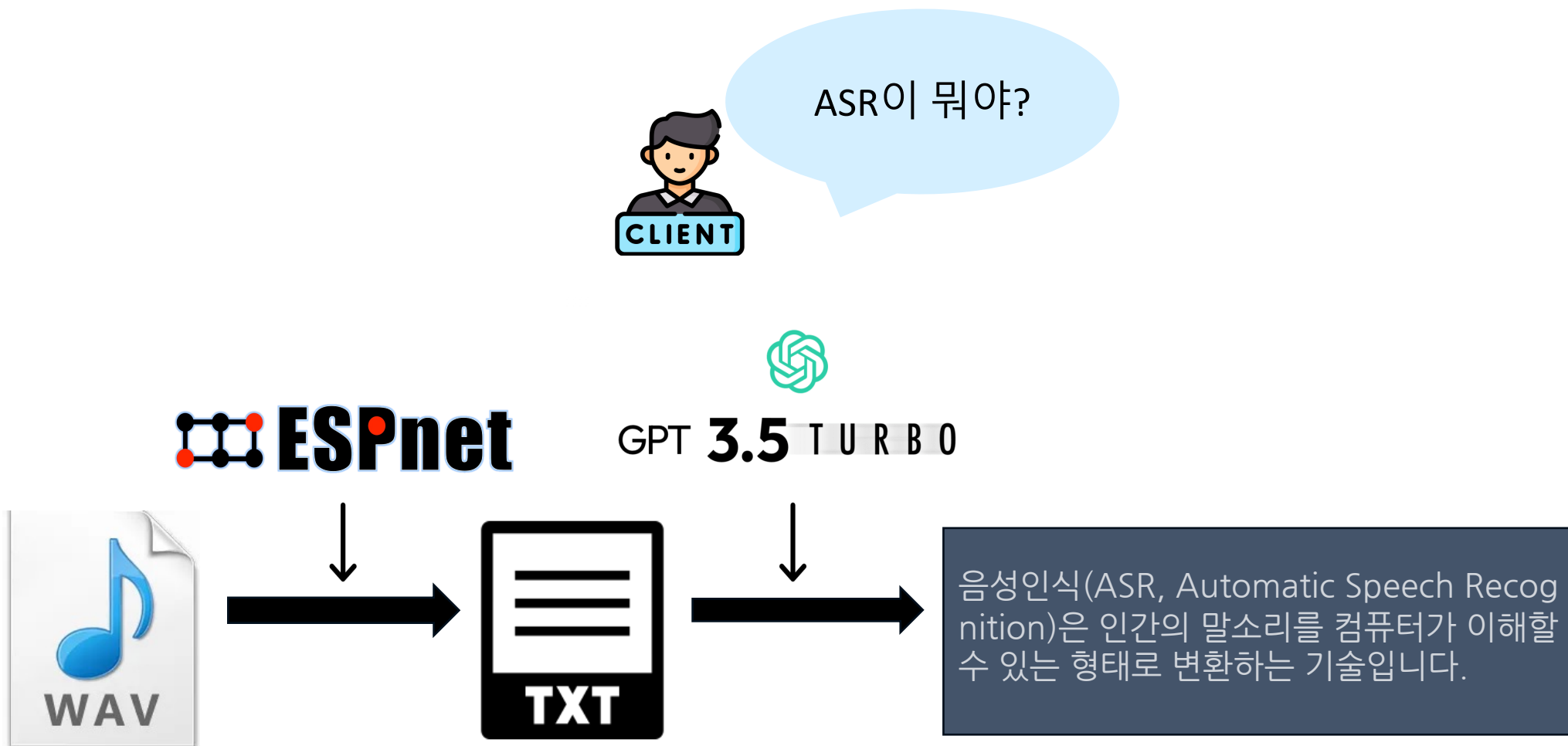
dataset	Snt	Wrd	Corr	Sub	Del	Ins	Err	S.Err
baseline	6256	133054	92.1	4.0	3.9	4.5	12.4	63.7
전사 처리 token 목록 변경	6256	128784	93.4	4.0	2.5	2.8	9.3	61.9



- LM 적용했을 때 성능 소폭 하락
- LM은 문맥에 따라 가능성 높은 단어 예측  
→ 조사, 어... , 자... 와 같이 뜻이 없는 말들을 대체 혹은 삭제
- ERR (error rate reduction)  
 $(12.4 - 9.3) / 12.4 * 100 = 25\%$



# 음성인식 서비스-음성 chatGPT



# | 음성인식 서비스-음성 chatGPT



⚠ Not Secure 117.17.185.207:9000



Understand  
Assistant

Choose File No file chosen

Submit

- ERR (error rate reduction) = 25% --> 기존 베이스라인에서 25% 성능 향상
- 성능 향상 요인
  - 이중 전사에서 한글 선택
  - 토큰 목록 변경
- 실험을 통해 알아낸 점
  - 토큰 수가 많은 토큰 목록을 사용하는 것은 데이터가 부족할 때 유용
  - LM 성능 소폭 하락
  - 속도 증강은 많은 시간 필요함

- ERR (error rate reduction) = 25% --> 기존 베이스라인에서 25% 성능 향상
- 느낀점
  - 언어 모델 성능 소폭 하락이 아쉬움. 왜 성능이 나빠졌는지 이유를 알고 싶다.
  - 인공지능 학습은 제한된 시간 속에서 선택의 연속인 것 같다.

# 감사합니다

2024년도 동계 SCI 인공지능 부트캠프  
하모니