주간보고

주제: enhancement in pseudo-stream-asr using ctc-segmentation – with its evaluation

학생 최재원

Prof. Jae-Hong Lee

Table of Contents

진행 상황 보고 | 250825

- 현재의 rule-based는 상당히 취약한 부분이 많음
  - Speech2Text의 nbest=1를 수정하면 더 많은 hypothesis를 제공함을 확인함
    - 여러 개의 후보를 두고 가장 잘 연결되는 후보를 gold로 결정하는 부분이 추가되면 좋을 것 같음

- KD를 통해 stream 가능하도록 시도해보고 싶음
  - 충분한 시간이 필요하며, 당장 진행하기에는 어려움이 많음

- librispeech pretrained model을 성공적으로 학습하지 못했는데, 관련 hyper-parameters을 공부하고 잘 학습시켜 본 데모에 적용해보고 싶음
  - ex. 어떤 모델은 어떤 optimizer가 좋은 성능을 내고, 그 때 lr이나 weight_decay 등은 얼마나 조절?
  - ex. gradient accumulation 공부로 VRAM이 부족하면 batch_bins를 줄이고, 그 비율만큼 accum_grads 올리면 거진 비슷한 걸 알게 되는 등

문제 정의

- **pseudo-stream-asr-250825**
  - 문자열 처리의 비정당성
    - concatenate strings with no structural information
  - 정확한 성능 측정의 어려움
    - real-time-factor
    - WER compared to original method

문제 정의 – 문자열 처리의 비정당성

- key idea of pseudo-stream-asr-250825
  - 1. infer using sliding-window concept
  - 2. join duplicated tokens – Algorithmic Approach
  - limitations
    - 의도적으로 같은 단어를 발화한 경우
    - 더 많은 컨텍스트를 통해 다른 토큰의 확률이 더 커진 경우
    - 이전 추론 결과와 현재 추론 결과가 상당히 다른 경우
    - …

문제 정의 – 정확한 성능 측정의 어려움

- performance of pseudo-stream-asr-250825 depends on running devices!
  - the more elapsed time inferencing, the lesser context it gets

    - case 1 – inference takes 1.5s in average
      - t = 9
        - model gets context (3, 4, 5, 6, 7, 8) and wait for 3 epoch
      - t = 10, 11, 12
        - not start inferencing thread under awareness of concurrency led by nested inference

    - case 2 – inference takes 0.3s in average
      - t = 9
        - model gets context (3, 4, 5, 6, 7, 8) and waiting is not required
      - t = 10
        - model gets context (4, 5, 6, 7, 8, 9) and waiting is not required

문제 정의 – 정확한 성능 측정의 어려움 | RTF

- **r**eal-**t**ime-**f**actor – RTF
  - The real-time factor (RTF) of a device measures how fast the embedded speech model can process audio input.
  - It's the ratio of the processing time to the audio length.
  - For example, if a device processes a 1-minute audio file in 30 seconds, the RTF is 0.5.

src: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/embedded-speech-performance-evaluations#metrics--terminology

기존 연구

- N/A

문제 해결 방안 및 실험 결과 – 문자열 처리의 비정당성

- justify pseudo-stream-asr using ctc-segmentation
  - 추론 결과(str)와 음성 데이터(wav)를 통해 start, end, confidence_score 획득
    - 부정확한 정보지만 일관성이 있으므로 이를 활용해보기로 함

문제 해결 방안 및 실험 결과 – 문자열 처리의 비정당성


```python
self.segmentizer = CTCSegmentation(
    **self.config,
    fs=sr,
)
self.segmentizer.set_config(gratis_blank=True, kaldi_style_text=False)
```

```python
@typechecked
def __call__(
    self,
    speech: Union[torch.Tensor, np.ndarray],
    text: Union[List[str], str],
    fs: Optional[int] = None,
    name: Optional[str] = None,
) -> CTCSegmentationTask:
    """Align utterances.

    Args:
        speech: Audio file.
        text: List or multiline-string with utterance ground truths.
        fs: Sample rate in Hz. Optional, as this can be given when
            the module is initialized.
        name: Name of the file. Utterance names are derived from it.

    Returns:
        CTCSegmentationTask object with segments.
    """
```

```python
def __str__(self):
    """Return a kaldi-style ``segments`` file (string)."""
    output = ""
    num_utts = len(self.segments)
    if self.utt_ids is None:
        utt_names = [f"{self.name}_{i:04}" for i in range(num_utts)]
    else:
        # ensure correct mapping of segments to utterance ids
        assert num_utts == len(self.utt_ids)
        utt_names = self.utt_ids
    for i, boundary in enumerate(self.segments):
        # utterance name and file name
        utt_entry = f"{utt_names[i]} {self.name} "
        # segment start and end
        utt_entry += f"{boundary[0]:.2f} {boundary[1]:.2f}"
        # confidence score
        if self.print_confidence_score:
            utt_entry += f" {boundary[2]:3.4f}"
        # utterance ground truth
        if self.print_utterance_text:
            utt_entry += f" {self.text[i]}"
        output += utt_entry + "\n"
    return output
```

문제 해결 방안 및 실험 결과 – 문자열 처리의 비정당성

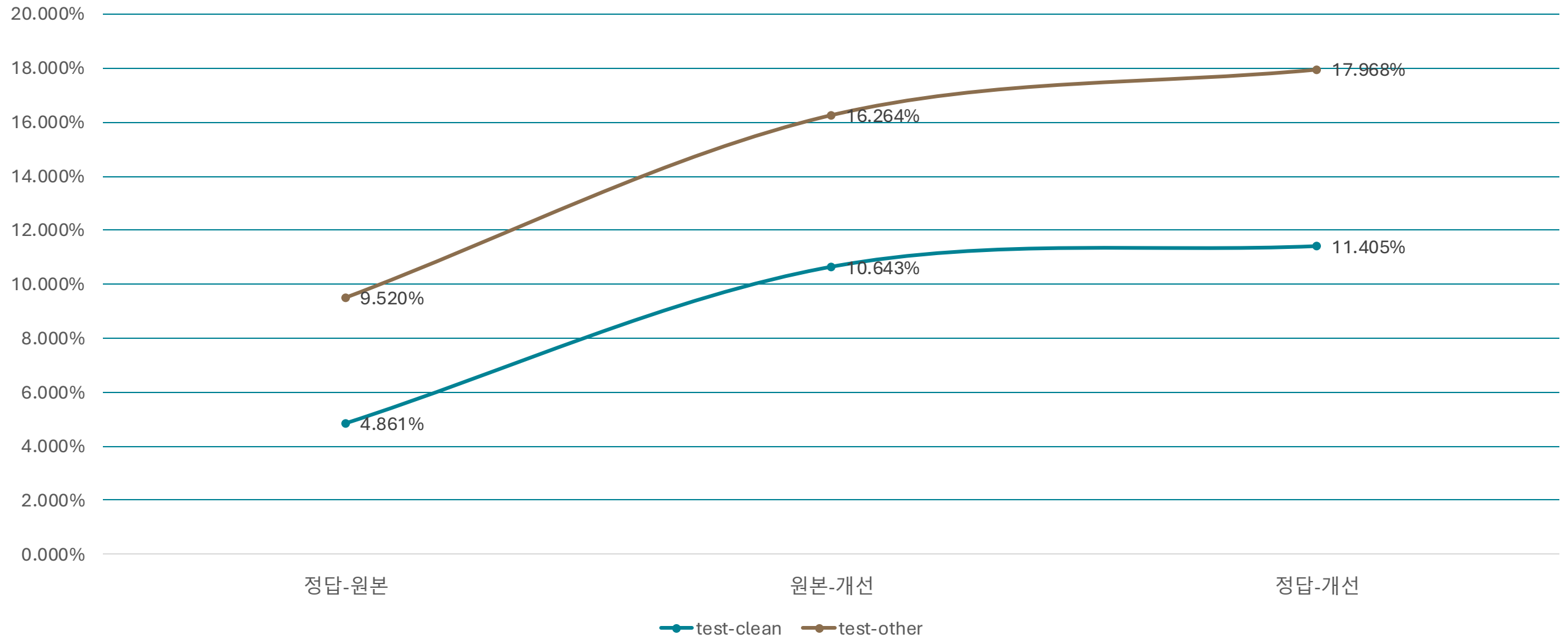

```
seg = self.segmentizer(wav, tokens)
```

```
for i, (start, end, min_avg) in enumerate(seg.segments):
    joiner.append_context(
        start_window=window[0],
        start_second=start,
        end_second=end,
        recognized=seg.text[i],
        min_avg=min_avg,
        segment_index=i,
    )
```

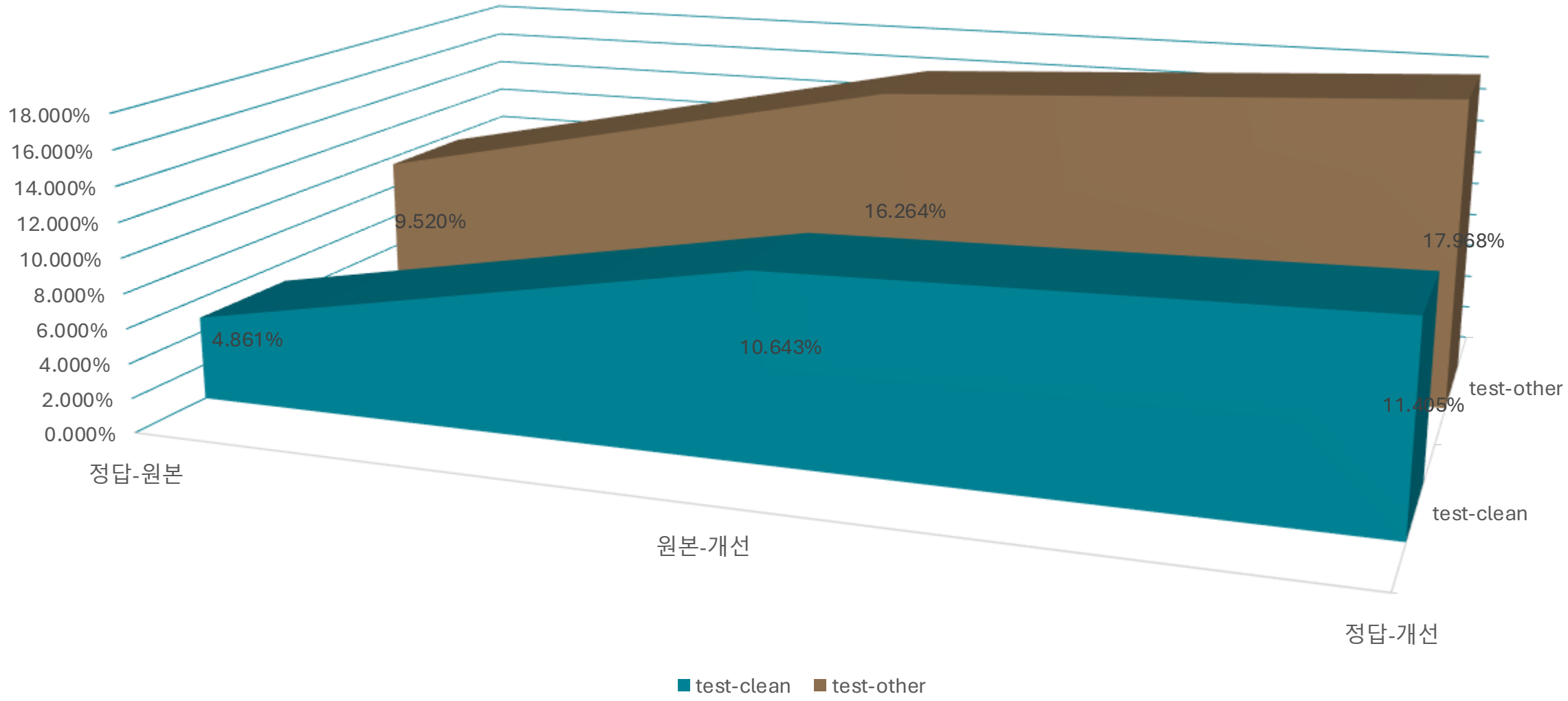
문제 해결 방안 및 실험 결과 – 정확한 성능 측정의 어려움 | WER

- justify pseudo-stream-asr using ACCURATE METRICS
  - 최선의 시나리오에서 얻어진 WER을 제시함
    - 최선이란?
      - 각 시점에서의 추론이 0.5초 이내로 끝나 최대한 많은 context를 모델에 넣어주었을 때
      - 실사용 환경과 유사하도록, 맨 마지막 음성이 혼자 모델에 들어갈 때까지 추론
    - 예시 | 0.5초씩 최대 6개 | 원본 음성이 context 15까지 있을 때
      - …
      - (10, 11, 12, 13, 14, 15)
      - (11, 12, 13, 14, 15, 00)
      - (12, 13, 14, 15, 00, 00)
      - (13, 14, 15, 00, 00, 00)
      - (14, 15, 00, 00, 00, 00)
      - (15, 00, 00, 00, 00, 00)

문제 해결 방안 및 **실험 결과** – 정확한 성능 측정의 어려움 | WER



문제 해결 방안 및 **실험 결과** – 정확한 성능 측정의 어려움 | WER



결론 및 추후 진행 예정 사항

- 단순히 비슷한 토큰을 기준으로 하여 이어 붙이던 기존 방식에서 벗어나 근거를 가지고 이어 붙일 수 있도록 하였음
  -
  -
    -
- 정확한 지표를 통해 개선 사항을 수치적으로 확인할 수 있도록 하였음
  -
    -
  -
    -

결론 및 **추후 진행 예정 사항**

- 단순히 비슷한 토큰을 기준으로 하여 이어 붙이던 기존 방식에서 벗어나 근거를 가지고 이어 붙일 수 있도록 하였음
  - 현재는 추론 당시에 가장 그럴 듯한 문장 하나를 “시간대”, “confidence” 등을 통해 연결함
  - n개의 hypothesis를 적당히 오래 유지하며 이들에 대해서도 ctc-segmentation을 적용할 예정
    - 추론 과정에서의 score와 ctc-segmentation을 통해 얻은 confidence score를 적당히 고려하며 늘림

- 정확한 지표를 통해 개선 사항을 수치적으로 확인할 수 있도록 하였음
  - 각 시점에서의 infer와 ctc-segment를 캐시하여 알고리즘을 빠르게 테스트할 수 있도록 할 예정
    - 추가로, 두 작업에서 공통적으로 인코딩을 진행하므로 한 번만 인코딩 후 이를 공용하도록 할 예정
  - 현재는 단순히 각 WER들의 산술평균만을 구하는데, 각 WER들의 분산이 매우 큰 것으로 보임
    - WER이 지나치게 큰 케이스에 대해서도 잘 맞추도록 알고리즘을 개선해야 함

결론 및 **추후 진행 예정 사항**

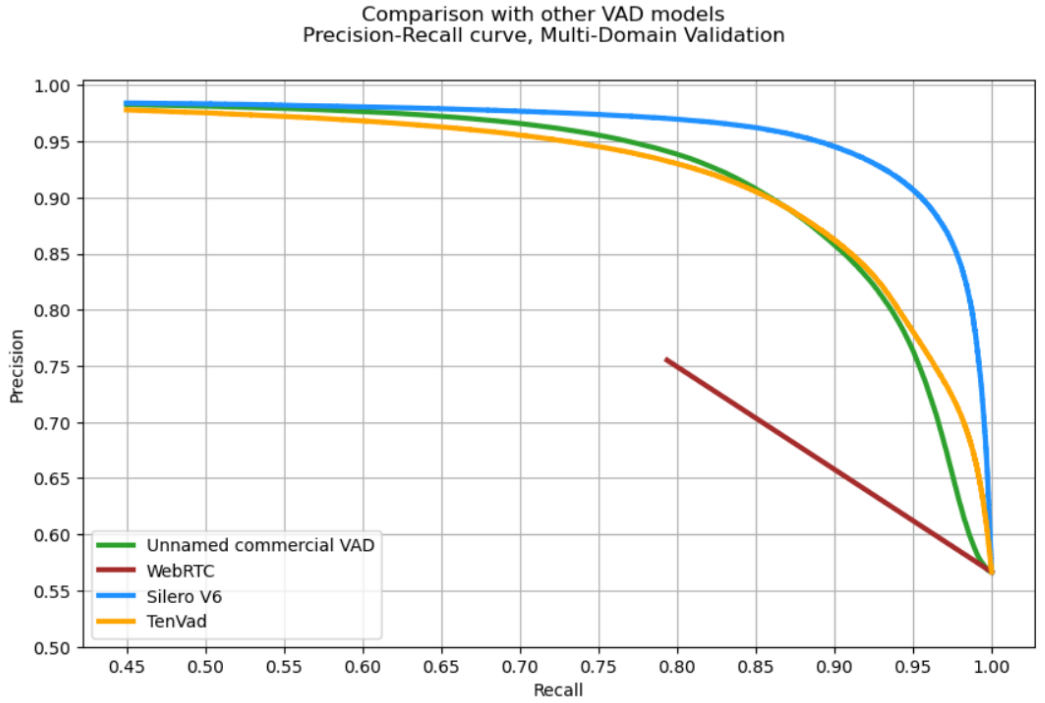
- 평균 추론 시간에 따른 RTF와 WER 등의 지표를 도출해내기
  - 현재는 최선의 WER 지표만을 도출함
  - 평균 추론 시간별로 RTF와 그 때의 WER을 측정하고자 함
    - |substitutions|, |deletions|, |insertions|, |correct words|로 구해지므로 각각의 지표에 대해서도 수집해야

- 이전 버전들에 대한 지표들도 측정해 비교하기
  - 음성을 안 겹치게 추론하고 단순히 이어 붙인 결과 – 이것 보다는 좋아야 의미 있음
  - 음성을 겹치게 추론하고 단순히 이어 붙인 결과 – 250825, 250912와의 비교를 위함
  - 음성을 겹치게 추론하고 알고리즘을 통해 적절히 이어 붙인 결과 – 250825
  - 음성을 겹치게 추론하고 ctc-segmentation과 알고리즘을 통해 이어 붙인 결과 – 250912 | 완료

결론 및 추후 진행 예정 사항

- 최선에 대한 고민
  - 맨 마지막 context 하나가 제로 패딩된 추론이 끝날 때까지를 완료로 보았음
  - 오프라인 모델의 구조를 생각하면, 필요할 것 같지 않음
  - 의사 결정을 위해 실험 후 데이터를 보고 판단해야

- 데모 제작 | Application Approach
  - Silero VAD (github | snakers4/silero-vad)
    - Fast
    - One audio chunk (30+ ms) takes less than 1ms to be processed on a single CPU thread.
    - Using batching or GPU can also improve performance considerably.
    - Under certain conditions ONNX may even run up to 4-5x faster.
  - 디노이즈



주간보고

학생 최재원
Language & AI융합전공

2025. 09. 12. 3 PM.
한국외대 교수회관 401호