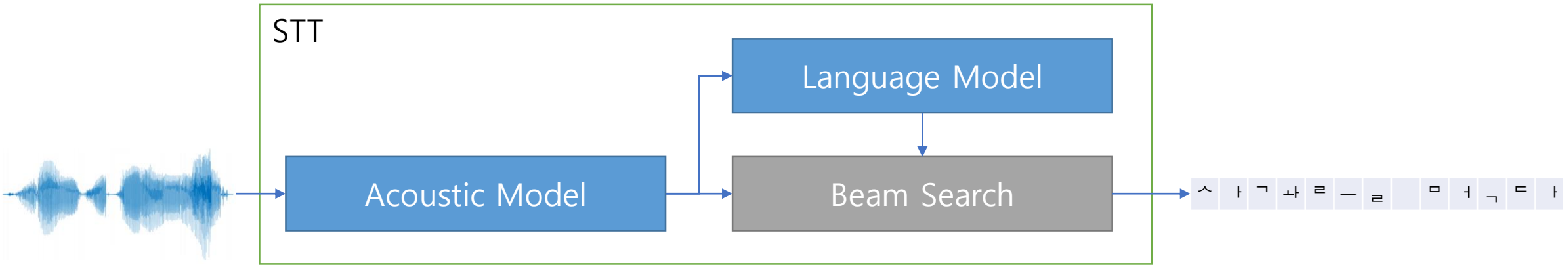
STT 엔진 개념

목차

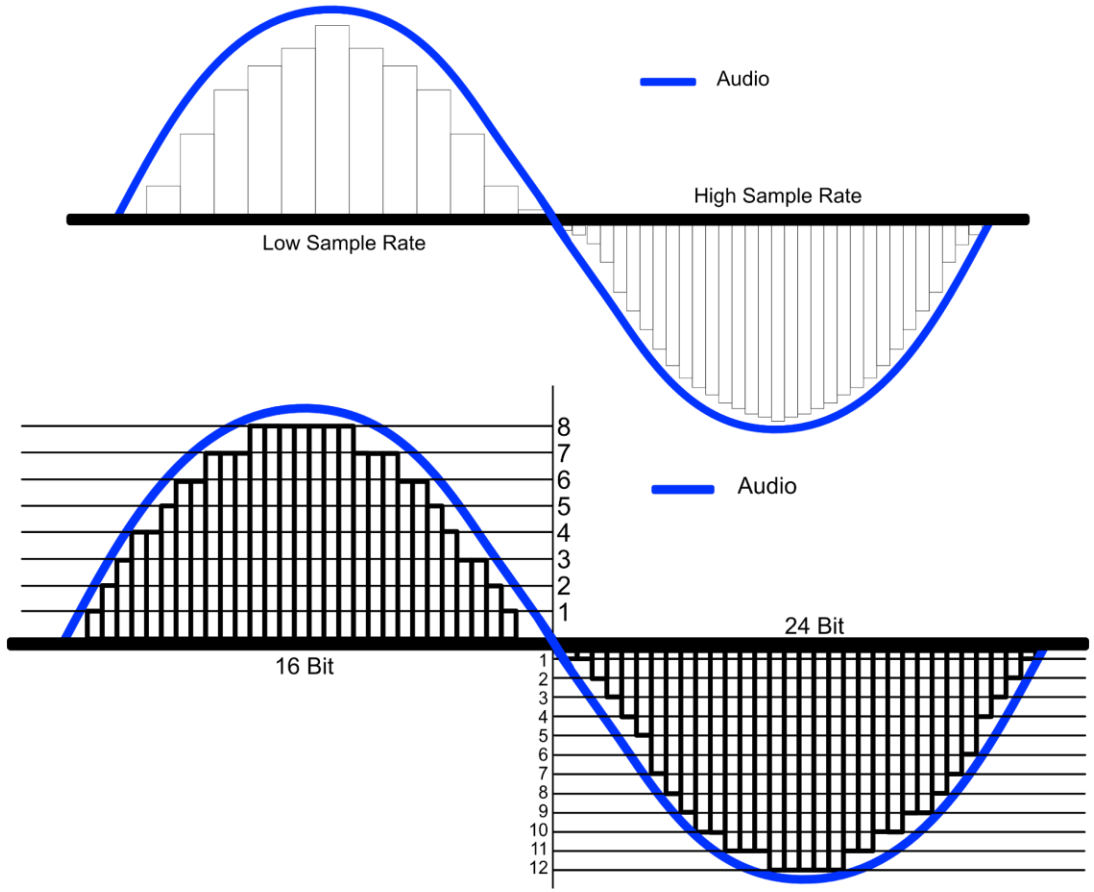
- Architecture
- Data
- Acoustic Model
- Language Model
- Beam Search
- Test
- Server
- ETRI

Architecture



Data - Speech

- File Format
  - Mp3를 제외한 대부분의 오디오 파일이 가능
  - http://www.mega-nerd.com/libsndfile/
- Sample Rate
  - 소리를 1초에 몇 개의 샘플로 표현하는지 설정
  - AM 모델의 sample rate와 같거나 높아야 됨
- Bit Depth
  - 샘플의 정밀도
  - pcm_s8, pcm_u8, pcm_16, pcm_24, pcm_32 등 다양하게 있고, 보통 pcm_16을 사용
- Channel
  - mono, stereo, ...
  - mono가 아니면 첫 channel(보통 left)만 사용

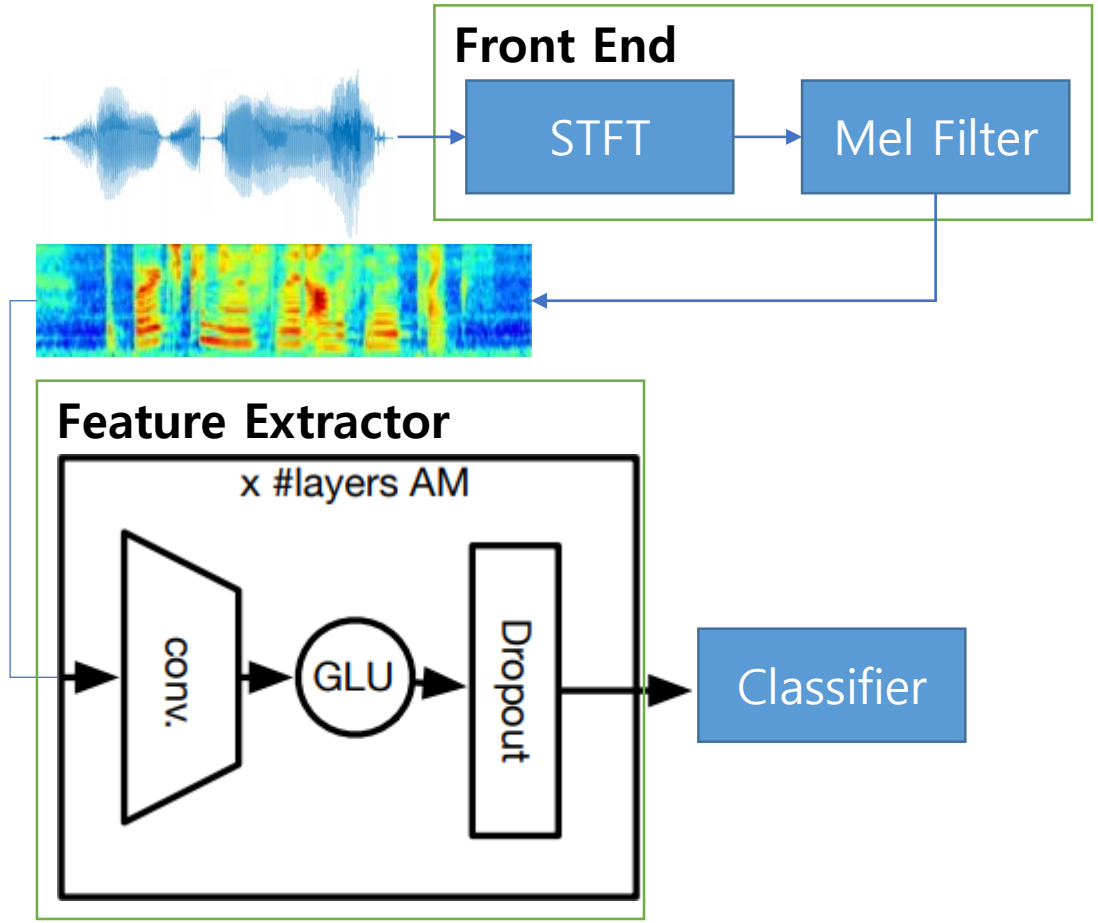


Data - Text

- AM 학습용 data는 음성 파일과 동일한 파일명.txt
- LM 학습용 data는 text만 있어도 가능
- 전사 작업을 할 때, 음성이 있는데 전사를 하지 않거나 음성이 없는데 전사를 하는 경우가 없어야 되고, 어느 정도 이상의 음량일 때부터 전사를 할지 명확하고 일관성 있게 작업을 해야 됨
- 문자 인코딩을 utf-8로 통일
- 언어 별로 사용할 수 있는 문자
  - 한국어: 가~힣, 띄어쓰기, 줄바꿈(optional)
  - 영어: a~z, ' (apostrophe), 띄어쓰기, 줄바꿈(optional)

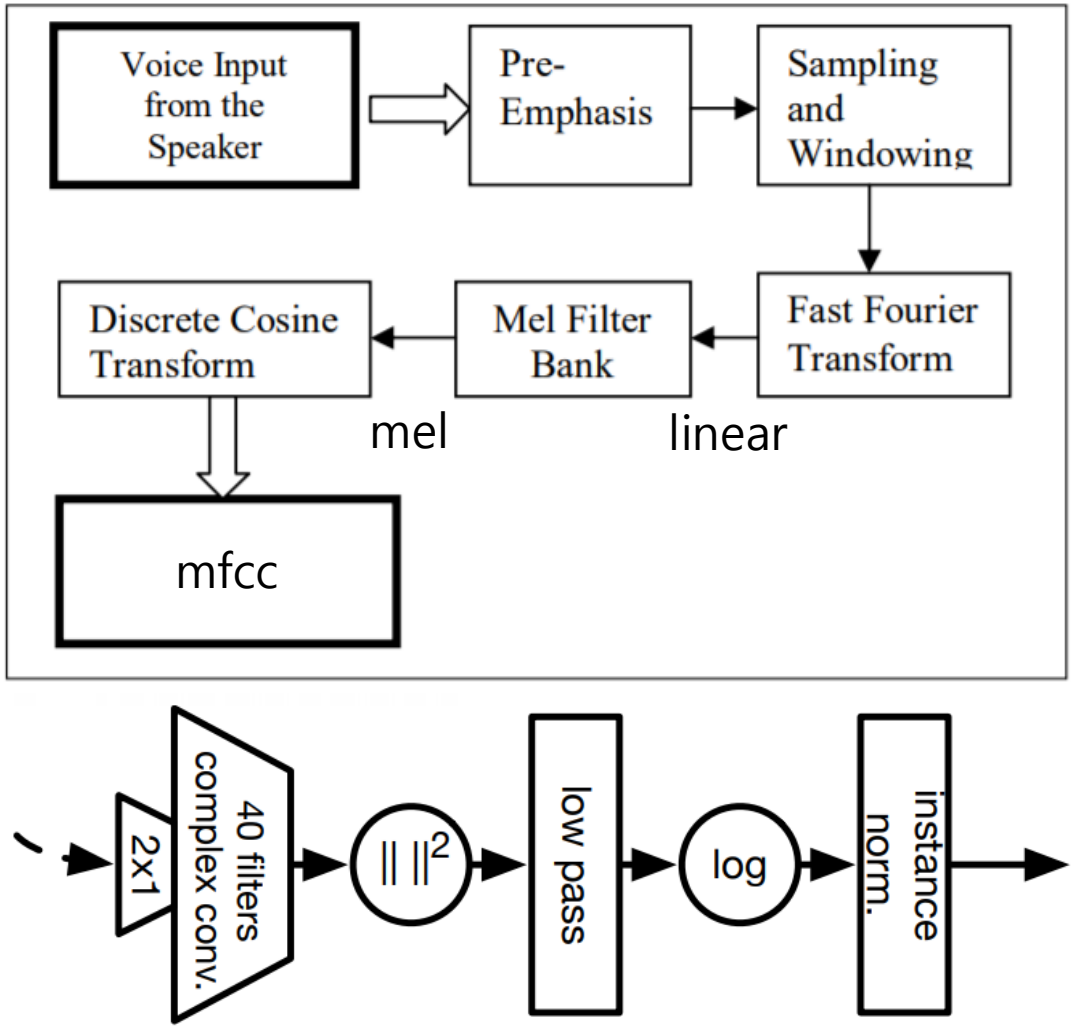
Acoustic Model

- 음성 정보만으로 text를 예측하는 모델
- 구성 요소
  - Frond End
  - Feature Extractor
  - Classifier
- Domain
  - 화자의 인종, 성별, 연령, 출신 지역
  - 녹음 장소, 녹음 장비 , 노이즈와 같은 환경 요소



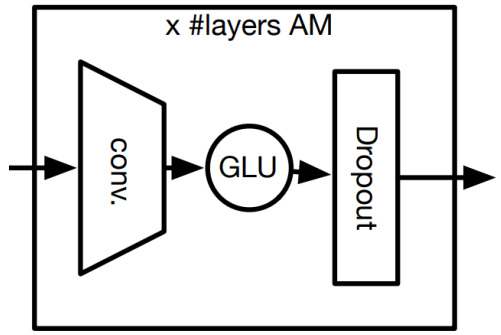
Acoustic Model - Frond End

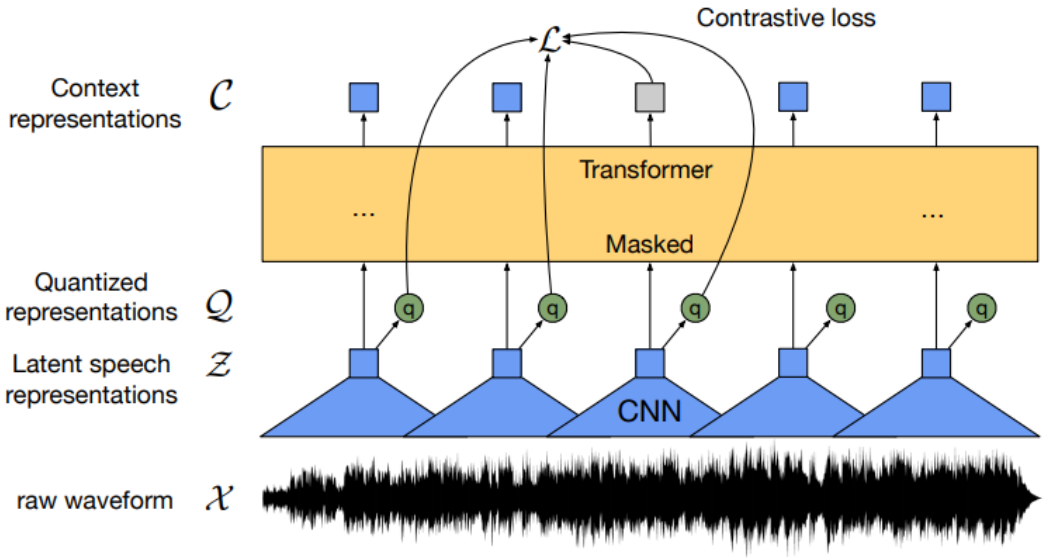
- Raw Audio에서 low-level feature를 추출하는 모듈
- Option
  - none
  - linear spectrogram
  - **mel spectrogram**
  - mfcc
  - learnable front end
- 주황색은 고전적인 방법으로 음성에 대한 특징을 추출
- 초록색은 neural network로 학습
- 현재는 40ch, mel을 사용



Acoustic Model - Feature Extractor

- Frond End가 추출한 low-level feature에서 high-level feature를 만들어내는 모듈
- Neural Network로 구성
- 현재
  - (Conv + GLU) X 17 layer
  - Activation Function 대신에 gated linear unit을 사용하여 convolution을 깊게 쌓음
- 연구 중
  - Wav2vec 2.0
  - Nlp에서 bert를 사용하듯이 음성에서 사용할 수 있는 pretrained model
  - Front End 역할을 하는 걸로 보이는 convolution 위에 transformer encode를 bert처럼 쌓아 올림
  - 이걸 사용하게 되면, Front End는 None으로 사용



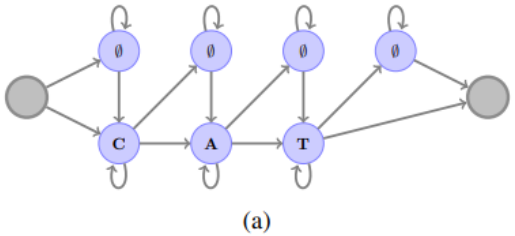


Acoustic Model - Classifier

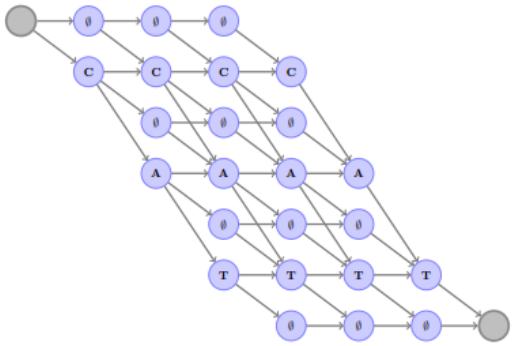
- Feature Extractor가 만든 high-level feature를 이용하여 text를 classification하는 모듈
- Criterion
  - Loss를 계산하는 방식
  - Option
    - CTC
    - **ASG**
    - Seq2seq
  - CTC와 ASG는 거의 같은 방식
  - Seq2seq는 Feature Extractor를 encoder로 사용하고 decoder를 붙임
  - 현재는 ASG 사용 중
  - Wav2vec 2.0에서는 CTC를 사용할 예정
- Token Dictionary
  - Classification하는 token을 정의
  - Option
    - **Letter**
    - Word Piece
  - 영어는 알파벳과 발음 사이의 괴리가 심하여 word piece가 확실히 좋지만, 한국어는 한글과 발음 사이의 LER가 10% 내외로 유사한 편이라 letter를 써도 무방
- End of Sentence Token
  - 문장 분리를 위한 token의 사용 여부
  - Option
    - **True**
    - False
  - EOS가 없는 모델을 baseline로 사용하여 eos 모델을 학습하는 기능도 구현

Acoustic Model - Classifier - CTC

- Connectionist Temporal Classification
- Input이 output보다 길 때 사용할 수 있는 방법
- 옆의 예시는 음성이 6 frame이고, text가 cat으로 3+2~4=5~7글자
- 정답으로 가능한 모든 경우를 loss로 먹이는 방식
- Success처럼 연속된 글자를 처리하기 위하여 "∅"와 같은 blank token을 글자 사이마다 삽입
- 긴 음성을 학습하면, text의 경우의 늘어나서 학습할 때 비효율적



(a)

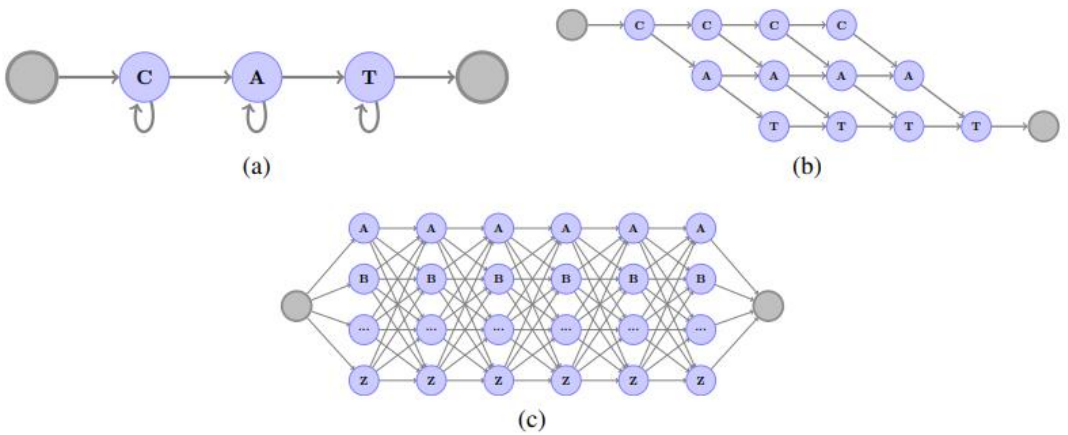


(b)

$$CTC(\theta, T) = - \underset{\pi \in \mathcal{G}_{ctc}(\theta, T)}{\text{logadd}} \sum_{t=1}^{T} f_{\pi_t}(x)$$

Acoustic Model - Classifier - ASG

- Automatic Segmentation Criterion
- blank token을 사용하지 않고, 숫자로 연속된 글자를 처리하기 때문에 정답의 경우의 수가 감소
  - Ex) caterpillar -> caterpil2ar
  - 학습 설정에서 replabel로 설정
  - 한국어는 초중종성이 연속되는 경우가 없어서 0으로 설정하고, 영어는 2로 설정
- 정답의 모든 경우를 분자로, 모든 경우를 분모로 하여 loss를 계산
- CTC가 cross entropy에서 분자만 쓴 느낌이고, ASG가 분모까지 사용한 느낌
- 수식의 g 함수는 글자 사이의 transition으로 학습하면서 만들어짐



$$ASG(\theta, T) = -\underset{\pi \in \mathcal{G}_{asg}(\theta,T)}{\text{logadd}} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x)) + \underset{\pi \in \mathcal{G}_{full}(\theta,T)}{\text{logadd}} \sum_{t=1}^{T} (f_{\pi_t}(x) + g_{\pi_{t-1},\pi_t}(x))$$

Acoustic Model - etc

- Noise Online Augmentation
  - Noise를 음성에 random한 크기로 random한 위치에 더하면서 학습
  - 학습 전에 미리 만들어 두지 않고, mini batch를 만들 때마다 다르게 합성
  - 음성에 noise를 랜덤하게 섞어서 모델이 noise를 무시하게 만듬
  - 학습 데이터가 적으면 음성에 있는 noise를 무시하게 학습이 되지 않기 때문에 이 기능을 사용
    - Ex) "네"라고 말할 때마다 우연히 새소리가 있었다고 하면, 새소리를 "네"로 인식하게 모델이 학습됨
  - 학습 데이터가 많으면 굳이 사용하지 않아도 무방

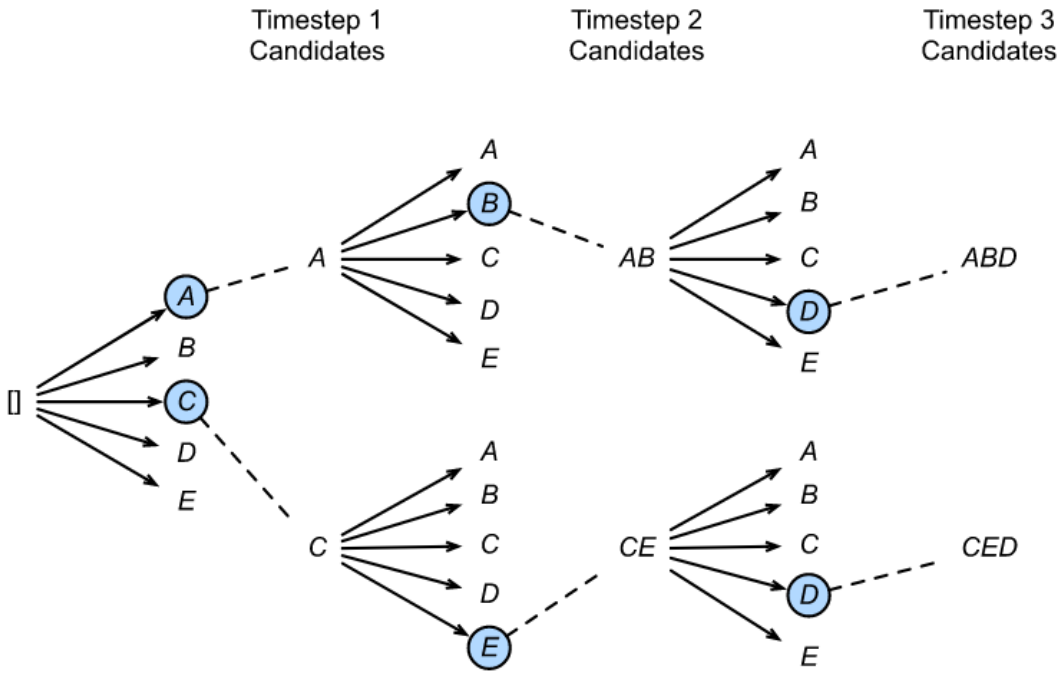
- 언어 확장
  - 언어의 token dictionary를 정의
    - 한국어: 한글 자소, 띄어쓰기
    - 영어: 알파벳, apostrophe, 띄어쓰기
  - Lexicon 정의
    - Word가 어떤 token으로 구성되는지 정의
    - Example
      - 한국어: 사과 = ᄉᆞᅡᄀᆞ과
      - 영어: bus = b u s
  - 음성을 lexicon에 있는 word로 구성되게 전사
  - 음성 파일, 전사 파일, token dictionary, lexicon으로 AM 학습
  - Word 혹은 token으로 쪼개진 text로 LM 학습
  - 다만, 한국어는 text를 NFD normalization 기능을 내장하여 AM 학습 시 lexicon이 필요하지 않음

Language Model

- Text에 대한 확률을 계산하는 모델
- 음성만 가지고 알 수 없는 문법적인 요소를 보정하는 역할
- Domain: 주로 사용하는 표현, 문구, 단어
- Model
  - **N-gram**
    - 통계적 언어 모델
    - n-1개의 symbol 다음에 어떤 symbol이 나올지 통계를 내고, 그 결과를 dictionary에 넣어서 사용
    - Word는 보통 3~4-gram, letter는 **20-gram**을 사용
  - ConvLM
    - Conv+GLU로 만든 딥러닝 모델
    - n-gram보다 더 좋은 성능이 나오지만 dictionary에서 n개의 symbol을 key로 value를 바로 가져오는 방식에 비하여 속도가 느려서 상용으로 사용하지 않음
- Lexicon
  - Word가 어떤 token으로 구성되는지 정의하는 dictionary
  - Lexicon을 사용하면서 unknown word에 대한 가중치 설정을 -∞로 하면, lexicon에 있는 단어만 나오게 됨
  - **Lexicon Free**: lexicon을 사용하지 않아서 모든 word가 나올 수 있는 상태
- Symbol
  - 언어 모델의 단위
  - Option
    - Word
    - **Token**(letter, word piece)
  - Word LM의 경우, lexicon 사용이 강제 됨

Beam Search

- AM, LM을 이용하여 아래의 score를 가장 크게 하는 text를 찾는 과정
- Score = AM(W)
  + **lmweight** * log(P_{lm}(W))
  + **wordscore** * |W_known|
  + **unkweight** * |W_unknown|
  - **silweight** * |{i| pi_i = <sil>}|
- LM Config
  - lmweight: lm score에 대한 가중치
  - wordscore: lexicon에 있는 단어에 대한 가중치
  - Unkweight: lexicon에 없는 단어에 대한 가중치. 기본값 -∞
  - Silweight: blank(띄어쓰기, 묵음)에 대한 가중치. 예를 들어 이 설정 값이 -0.8일 때 사람 목소리가 없는데 인식 결과가 나오고 있다면, 값을 -0.7 이런 식으로 증가시키면 저런 현상이 완화됨. 반대로 사람 목소리가 있는데 인식 결과가 안 나오면 이 값을 -0.9 이런 식으로 낮추면 됨
- https://pms.maum.ai/confluence/pages/viewpage.action?pageId=15342023

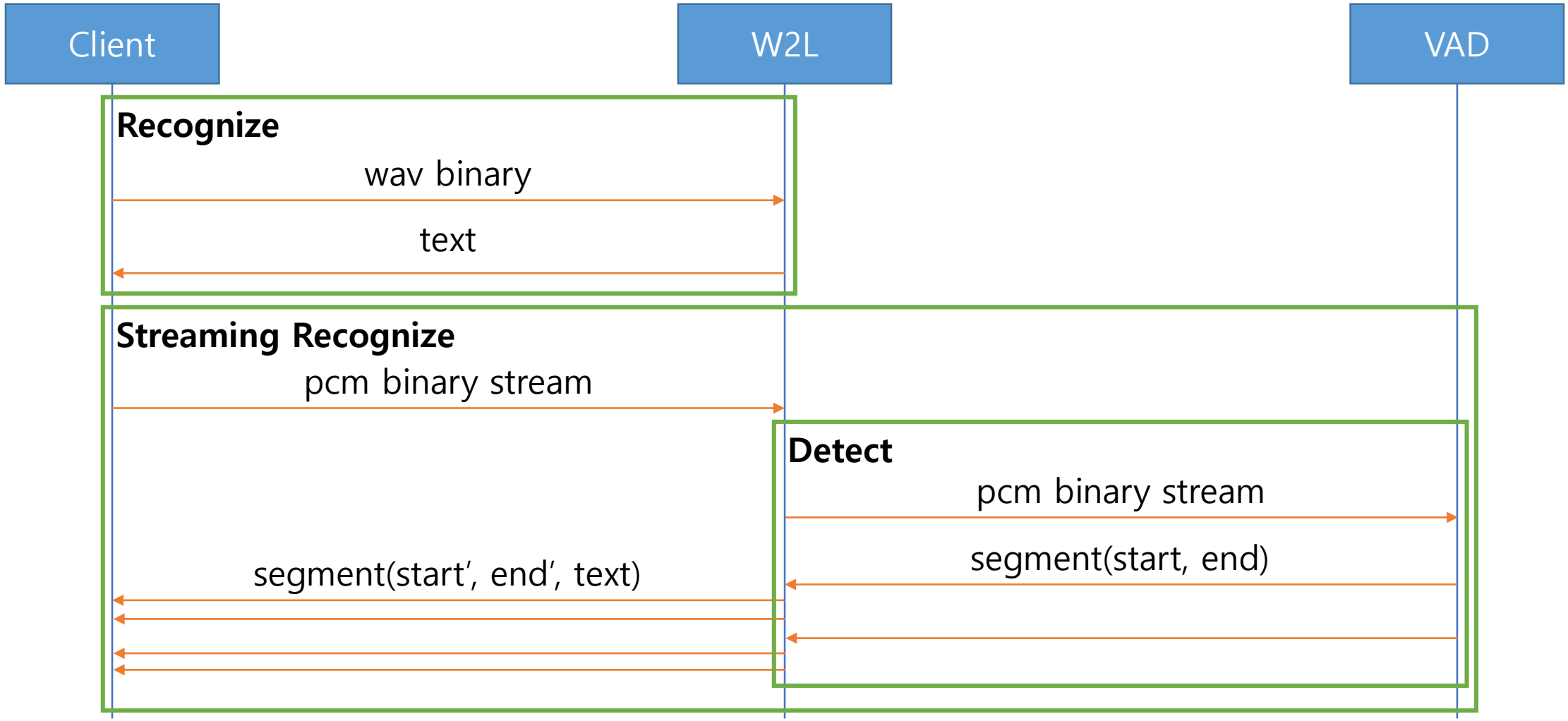


Test

- 정확도는 Error Rate로 측정
- 정답 symbol list와 예측한 symbol list가 있을 때, 예측한 결과와 정답을 동일하게 만들기 위하여 symbol을 추가, 삭제, 수정해야 되는 최소값이 분자, 정답 symbol list의 길이가 분모가 되어 계산
- LER
  - Letter 단위에서 error rate
  - 한국어는 조사 때문에 띄어쓰기 단위로 error rate를 측정하기 애매하여 이 지표를 사용함.
- WER
  - Word 단위에서 error rate
  - 영어는 이 지표가 표준
- https://pms.maum.ai/confluence/display/BRAIN/Evaluation+of+Speech+Recognition

- 속도는 TPS로 측정
- TPS = (인식한 음성 길이의 합) / (음성 인식 처리 시간)
- 이 값이 동시 사용 채널과 거의 동일
- AM은 GPU에 의존하고, LM은 CPU에 의존
- LM 비중을 낮춘 설정을 하고 하기 때문에, GPU는 거의 100%를 사용하게 됨
  - LM 비중이 높은 stt는 cpu만 100% 일을 하고 gpu가 거의 놀게 됨
- RNN 기반 모델보다 적은 동시 요청 수로도 GPU를 모두 사용할 수 있음
- https://pms.maum.ai/confluence/pages/viewpage.action?pageId=15340690

Server


Client
W2L
VAD
Recognize
wav binary
text
Streaming Recognize
pcm binary stream
Detect
pcm binary stream
segment(start, end)
segment(start', end', text)

ETRI

- 유사점
  - 전처리, 탐색 엔진, 언어 모델
- 차이점
  - 음향 모델이 DNN/LSTM
  - 음향 모델을 발음 기호로 학습
  - G2P 사용이 강제되어 언어를 확장하기 어려움
  - 언어 모델이 단위가 유사형태소
  - Lexicon 사용이 강제