J-pong Study: ASR system

Week 2 (2026.01.06)

Group1

Prof. Jae-Hong Lee

Table of Contents

Group 1

최서연	Data Preprocessing LM/ASR Model Training Evaluation + PPT 제작
장지수	Literature Review PPT 제작 발표 (Literature Review)
서원덕	서비스 배포 ASR Mode Inference PPT 제작 발표 (ASR Model)
강혜승	Literature Review PPT 제작

II Literature Review: Listen, Attend and Spell

Listen, Attend and Spell

William Chan
Carnegie Mellon University
williamchan@cmu.edu

Navdeep Jaitly, Quoc V. Le, Oriol Vinyals
Google Brain
{ndjaitly,qvl,vinyals}@google.com

- **Journal**: Listen, Attend and Spell
- **Year**: 2015 (Google)
- **Background**: 당시 음성 인식 시스템은 시스템 구성이 복잡, 각각 따로 최적화
- **Purpose**: End-to-End 시스템 구축, 가변 길이 시퀀스 문제 해결
- **Key Findings**:Pyramid BiLSTM, Attention Decoder, Rescoring, Scheduled Sampling

Background

기존 시스템의 한계

- 과거: 모듈 구조

DNN (Deep Neural Network)	HMM (Hidden Markov Model)	CRF (Conditional Random Field)
음성 feature 입력에 대해 음소 확률 계산	소리 - 글자 alignment	문맥상 어떤 라벨이 가장 자연스러운지 최종 결정

- 문제점1 : End-to-End 학습 불가
- 문제점 2: 데이터 확률 분포에 대한 너무 단순한 가정

Related Work

Sequence-to-Sequence Learning

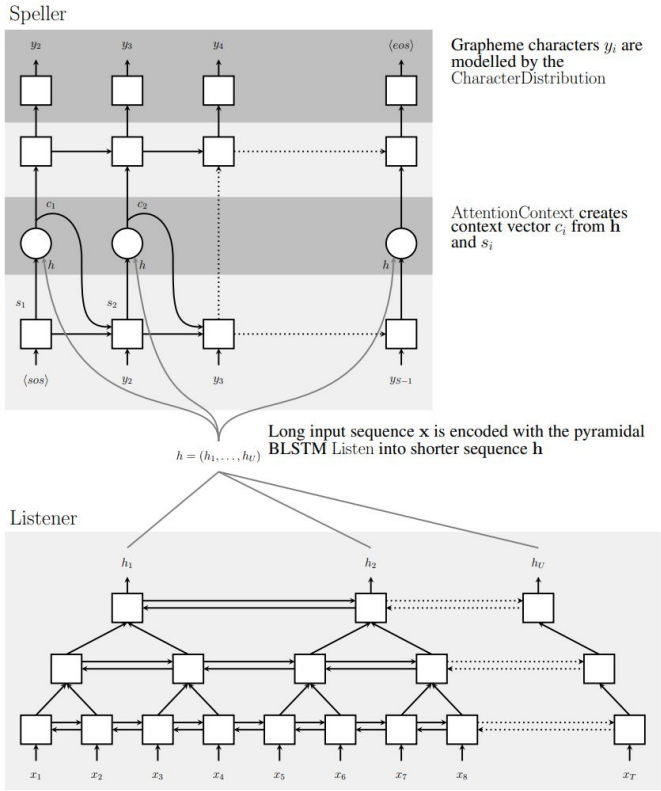
- 가변 길이의 입력을 가변 길이의 출력으로 매핑하는 프레임워크
- Encoder-Decoder 구조
  → Encoder: 입력을 고정된 길이의 벡터로 압축
  → Decoder: 벡터 참조하여 토큰 하나씩 생성
- 기계 번역, 대화 모델링 등에서 성공적으로 사용됨

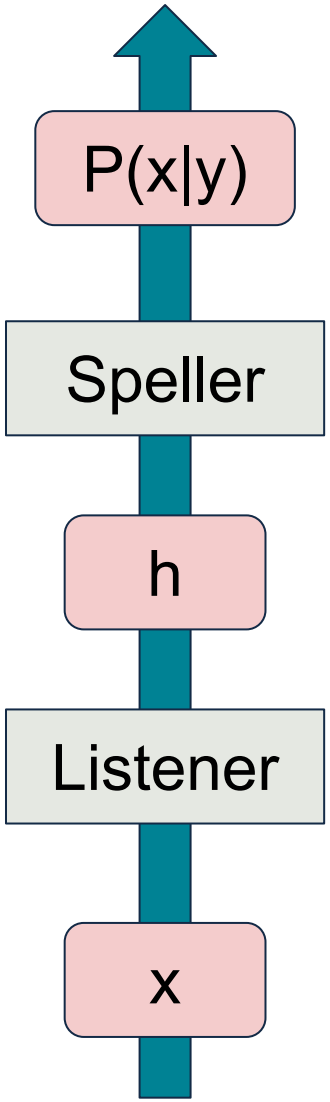
Attention Mechanism

- 기존 Seq2Seq는 인코더 정보를 한 번만 전달 → 긴 음성 처리에 한계 존재
- Las: 매 출력마다 Attention Vector 생성→긴 음성 처리 가능

Model

- LAS 모델은 입력 오디오 시퀀스 x를 문자 시퀀스 y로 매핑하는 확률 $P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, y_{<i})$
- 수식:
- 구성 요소
  - **Listener**: 원래의 신호를 high level representation (h)로 변환
  - **Speller**: listener가 만든 특징 h를 사용하여 문자 확률 분포 생성

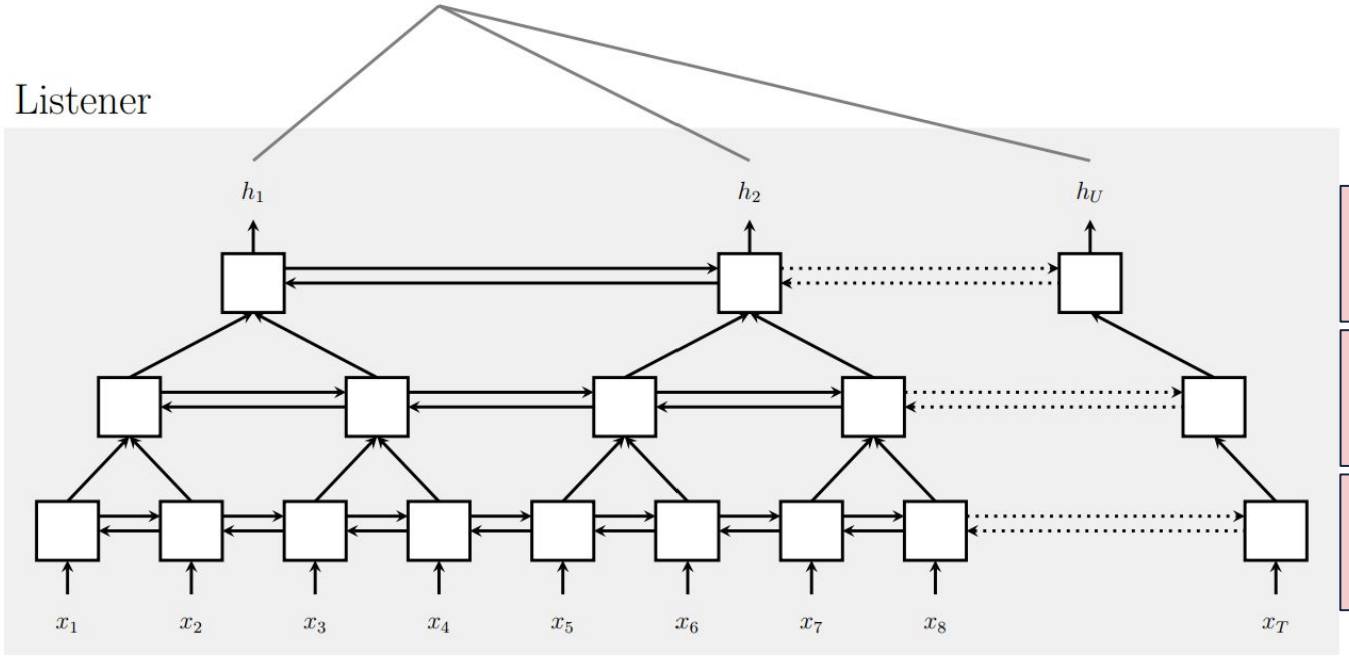




Model - Listen (Encoder)

Why Pyramidal Structure?

- 기존 BiLSTM의 한계: 음성 신호는 매우 길기 때문에 BiLSTM 사용시 학습 결과가 좋지 않음
  → Time Resolution 줄여주는 Pyramidal Bidirectional LSTM 도입



4

8

16

Model - Listen (Encoder)

작동 원리

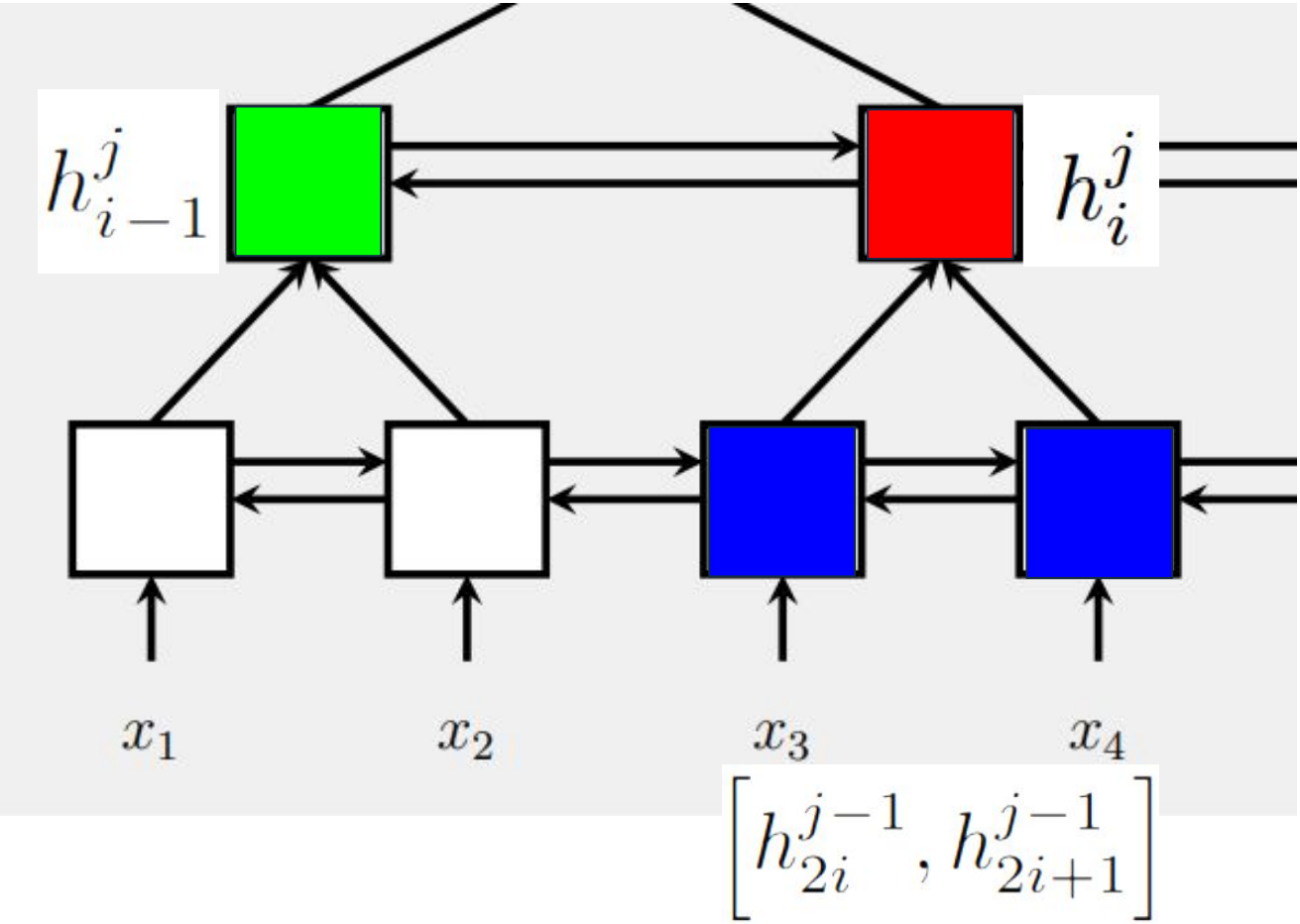
$$h_i^j = \text{pBLSTM}(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}])$$

concatenate

주요 성과

- 1층의 기본 BLSTM
- 3층의 pBLSTM 쌓아 -> time resolution 1/8
- 학습 , 추론 속도 향상
- 계층적 non-linearity 학습

1층: "아", "ㄴ", "녕" 짧은 소리 단위
2층: "안녕" 음절 단위
3층: “안녕 내 이름은…” 단어/문장 단위



Model - Attend and Spell (Decoder)

디코더 기본 메커니즘

- AttendAndSpell: 이전까지 생성된 모든 문자(y<i)와 입력 신호(x)를 바탕으로 다음 문자(y)의 확률 분포(P(y|x, y<i)) 출력

$$P(y_i|\mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

MLP with softmax

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \qquad c_i = \text{AttentionContext}(s_i, \mathbf{h})$$

Model - Attend and Spell (Decoder)

$$P(y_i|\mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

MLP with softmax

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \quad c_i = \text{AttentionContext}(s_i, \mathbf{h})$$

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$ 1 energy

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}$$ 2 attention weight

$$c_i = \sum_u \alpha_{i,u} h_u$$ 3 context vector

Model - Learning

Joint Training

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, y_{<i}) \xrightarrow{\log} \max_\theta \sum_i \log P(y_i|\mathbf{x}, y^*_{<i}; \theta)$$

Scheduled Sampling

- Exposure Bias → 학습 때 정답만 쓴 모델은 이 점이 취약 → 일정 확률로 모델이 예측한 문자를 사용 (10%)

Beam Search Algorithm

- 상위 n개의 후보군

Language Model Rescoring

- 많은 양의 텍스트 데이터 사용 위해 외부 언어 모델 결합
- Normalization & Scoring: 문자 수로 로그 확률 정규화하고, 언어 모델 확률을 가중치와 함께 합산

Experiments

Dataset: Google Voice Search Utterances

- 300만 건
- 2000시간 (이 중 10시간: validation set)
- data augmentation → x 20 (room simulator)
- 40d log-mel filter bank features computed every 10ms
- testset: separate 22,000건 (16시간)

Text Normalization

- convert to lower case
- <unk> token
- <sos> <eos>

Experiments

CLDNN-HMM [20]
LAS
LAS + LM Rescoring
LAS + Sampling
LAS + Sampling + LM Rescoring

Experiments

SOTA: CLDNN-HMM
- unidirectional

CLDNN-HMM [20]
LAS
LAS + LM Rescoring
LAS + Sampling
LAS + Sampling + LM Rescoring

Listen and Spell

Listen (Pyramid BLSTM)	Spell (LSTM)
구조: 입력층 → BLSTM → pBLSTM 3 노드 수: 각 방향 당 256개 씩 (총 512개)	구조: LSTM 2층 노드 수: 512개 씩 (총 1024개)

(initialization: U(-0.1, 0.1))

Experiments

Rescoring
- Beam search → 32 beams
- N-gram LM → rescored 32 beams

- decoding with LM (x)
- only resecoing with LM (o)

CLDNN-HMM [20]
LAS
LAS + LM Rescoring
LAS + Sampling
LAS + Sampling + LM Rescoring

Scheduled Sampling
- Context: Exposure Bias
- Solution: sampling from previous character distribution (10%)

Experiments

Experimental Setup

- ASGD
- Learning Rate 0.2 (geometric decay of 0.98 per 3M utterances)
- DistBelief framework with 32 replicas (복제 서버)
- Mini Batch Size: 32 utterances
- sequences were grouped (based on frame length)
- Decoding using “beam search algorithm” (width β = 32)

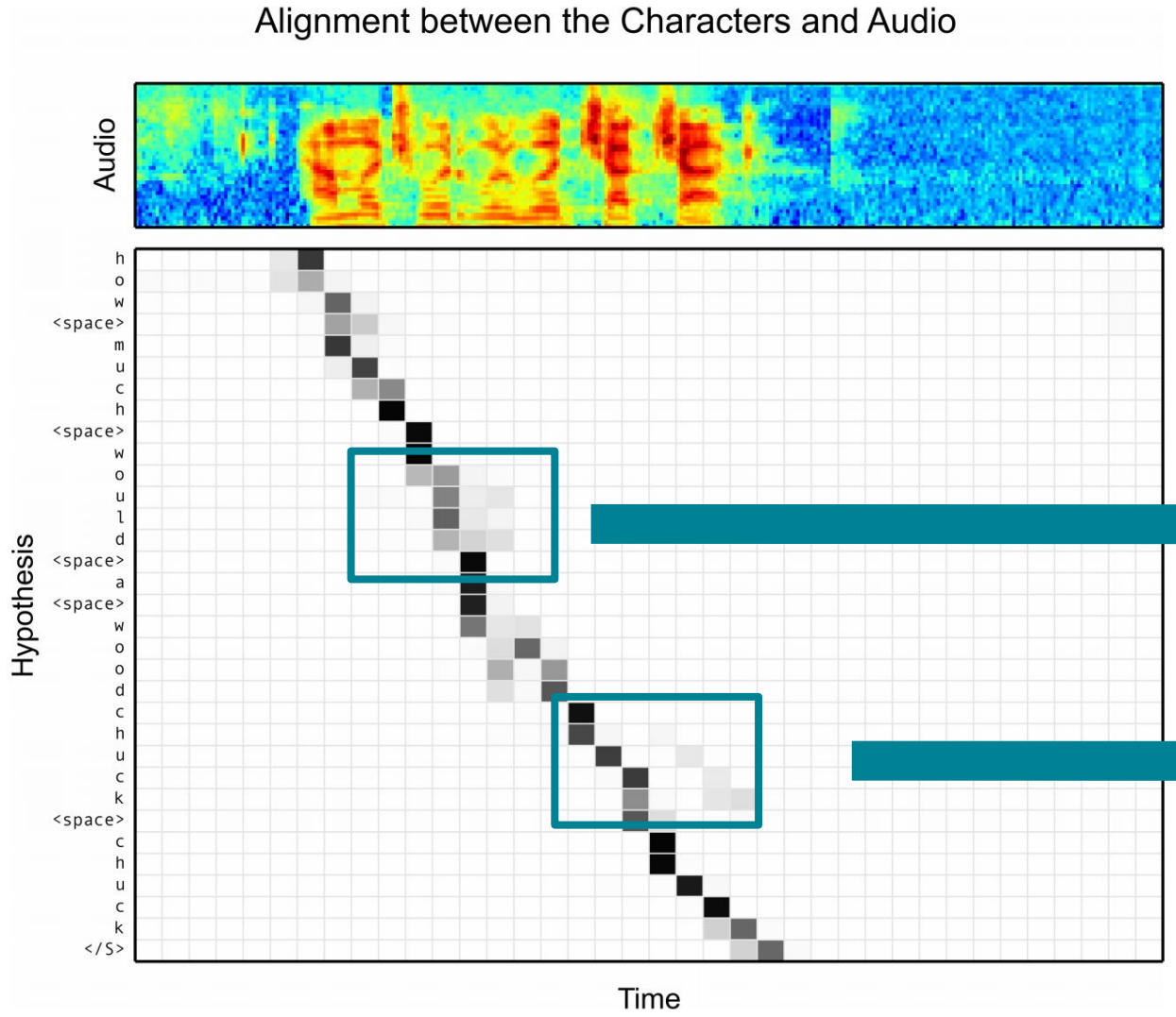
Training

- repeat until convergence (on validation set)
- without any dictionary or language model
- constraining on dictionary → no improvement

Experiments

Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

Experiments - Attention Visualization

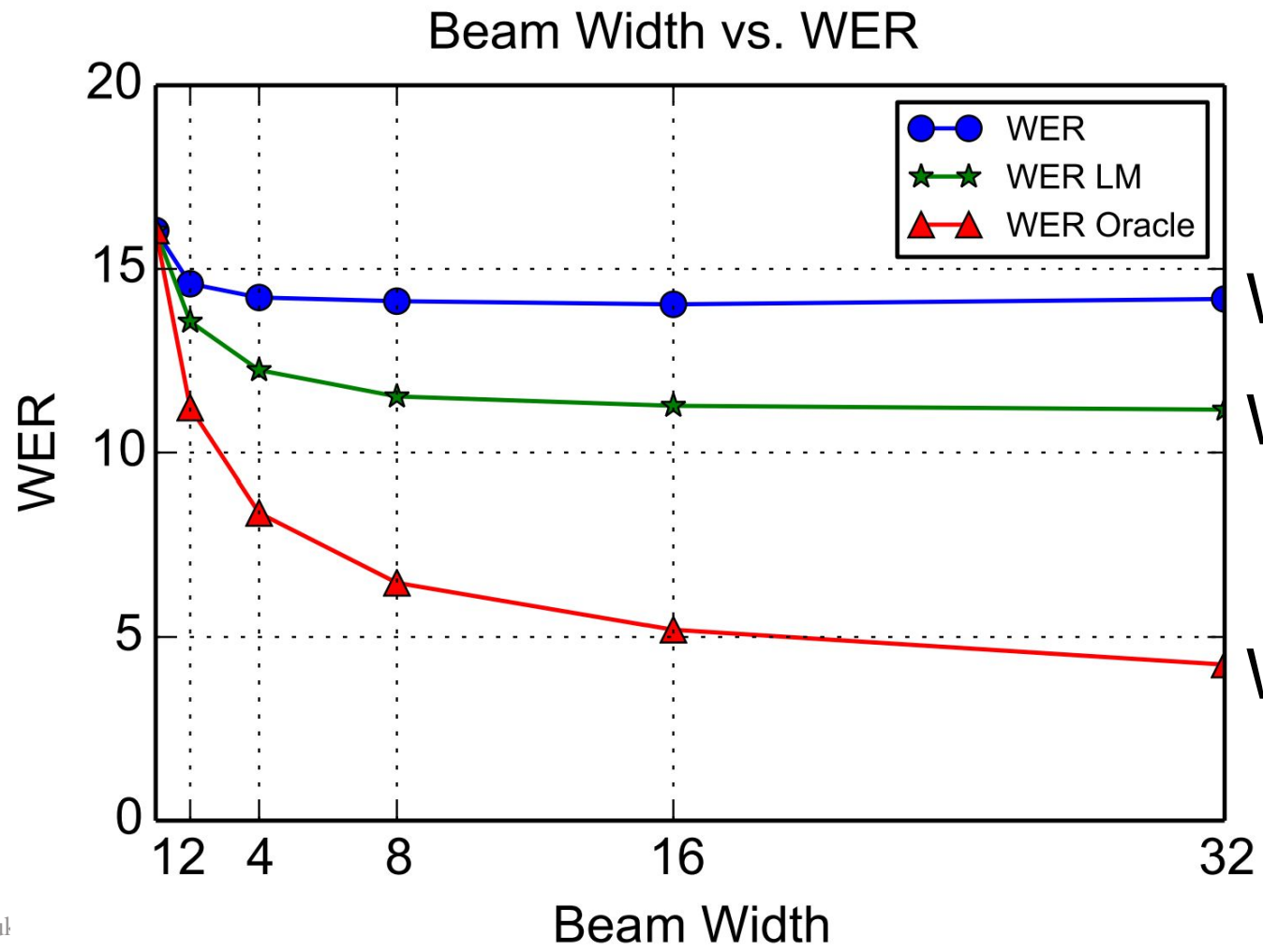


Dilution

How much would a woodchuck chuck

How much would a woodchuck chuck

Experiments - Effects of Beam Width

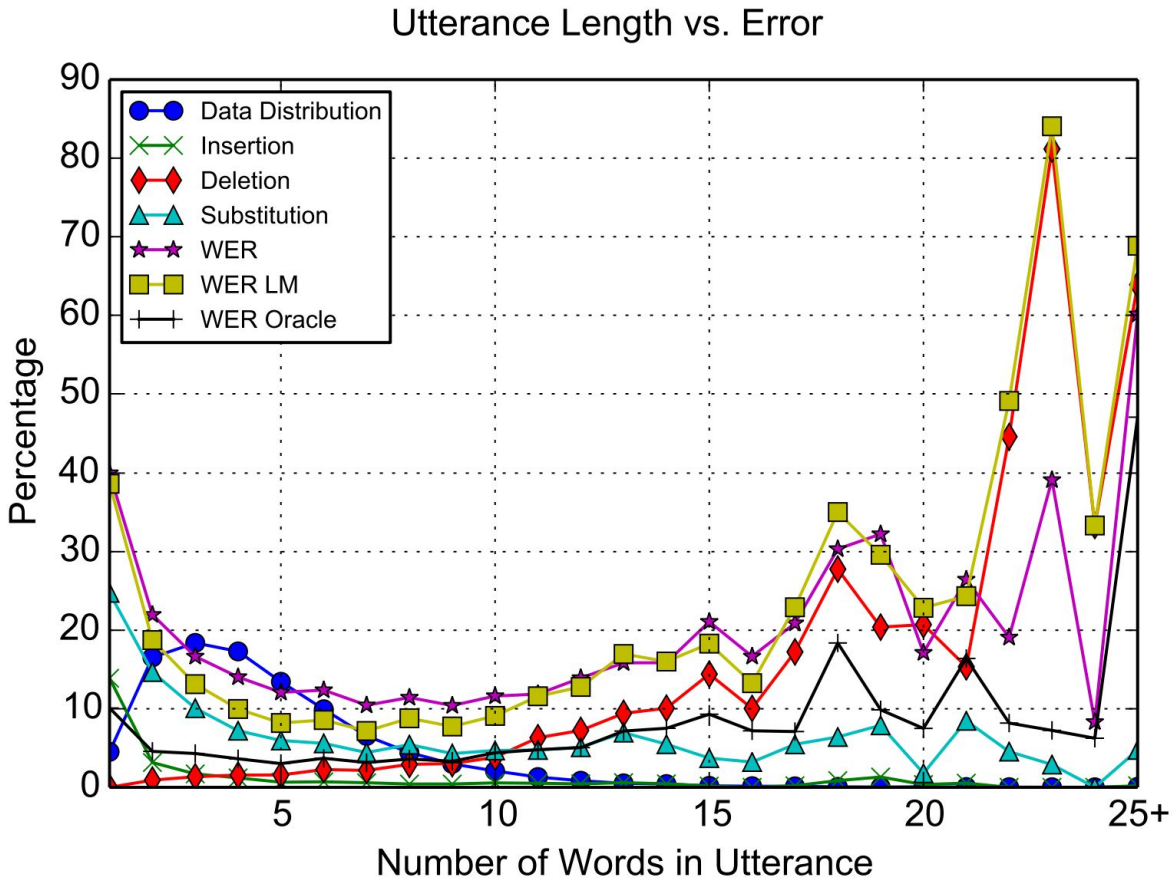


WER

WER (rescoring)

WER (oracle): 완벽한 가상의 존재

Experiments - Effects of Utterance Length

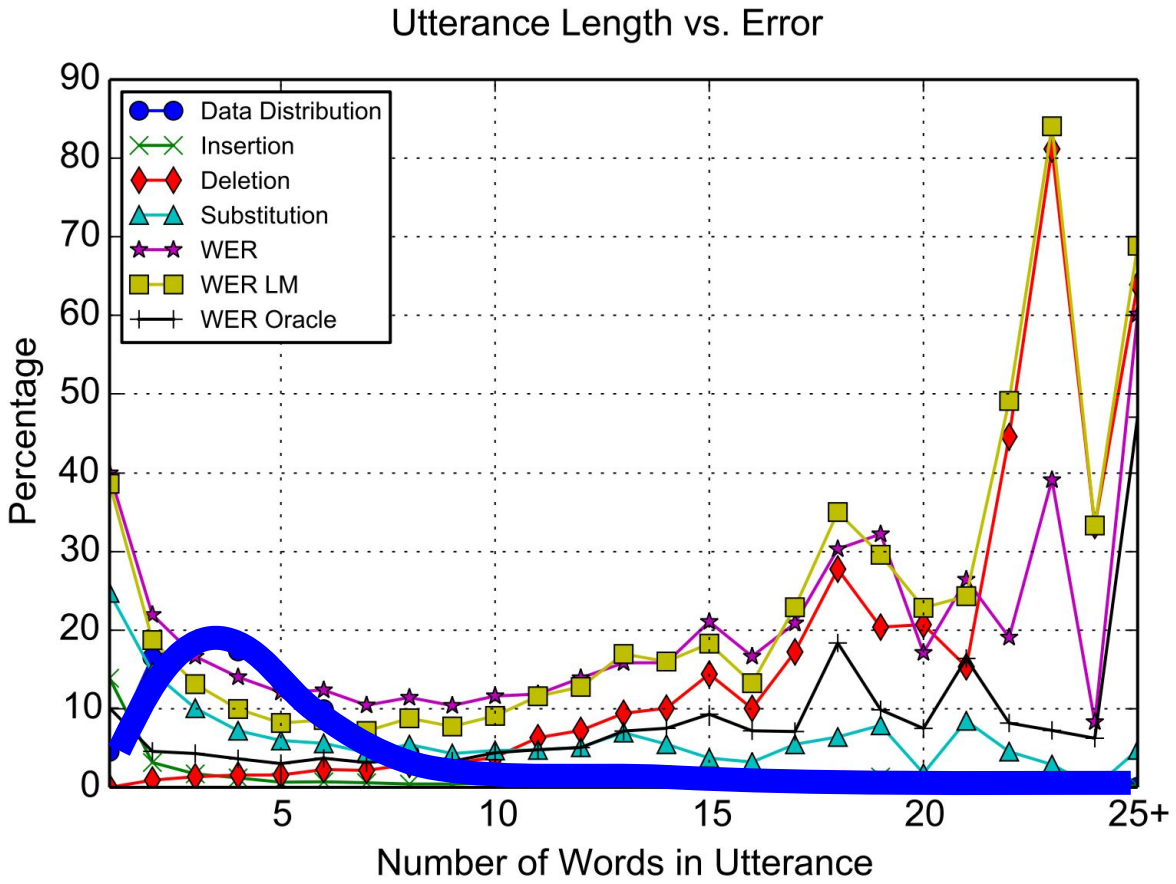


★ WER

■ WER (rescoring)

┼ WER (oracle)

Experiments - Effects of Utterance Length

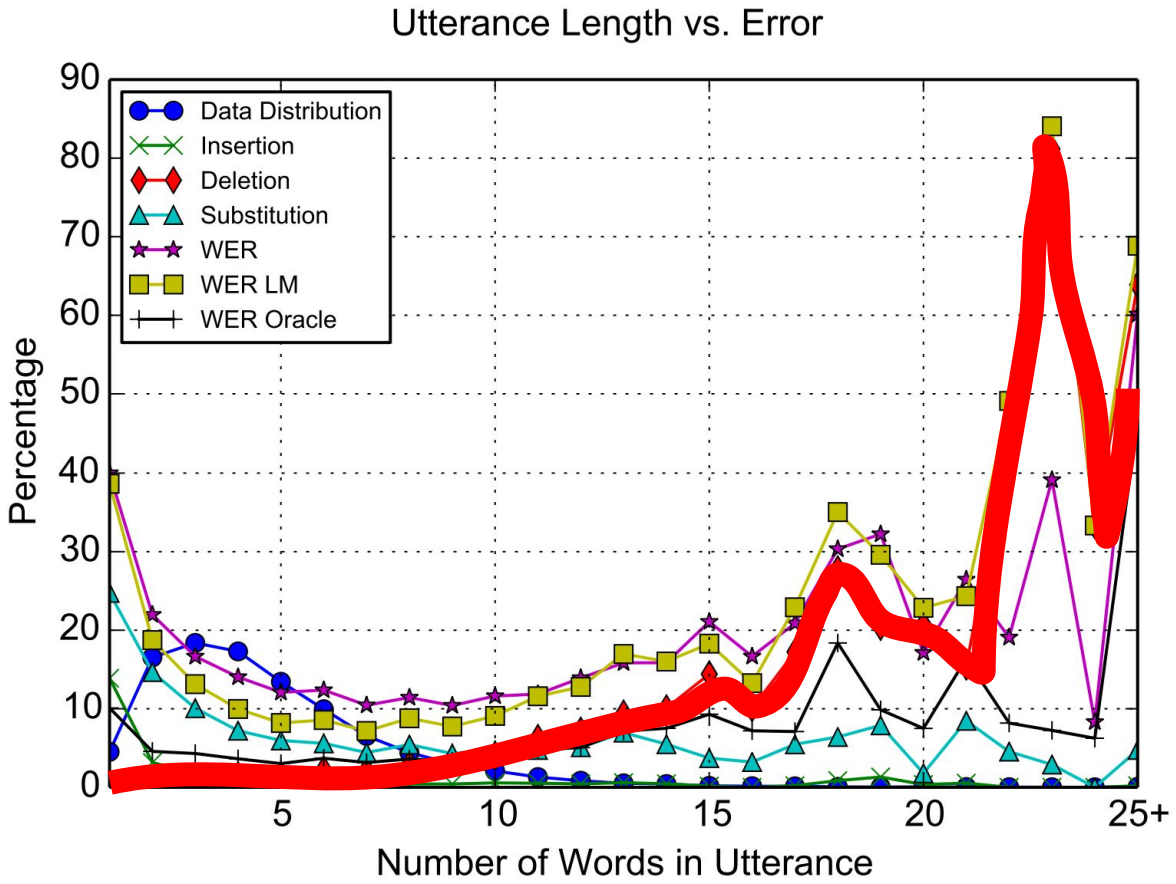


★ WER

■ WER (rescoring)

┼ WER (oracle)

Experiments - Effects of Utterance Length

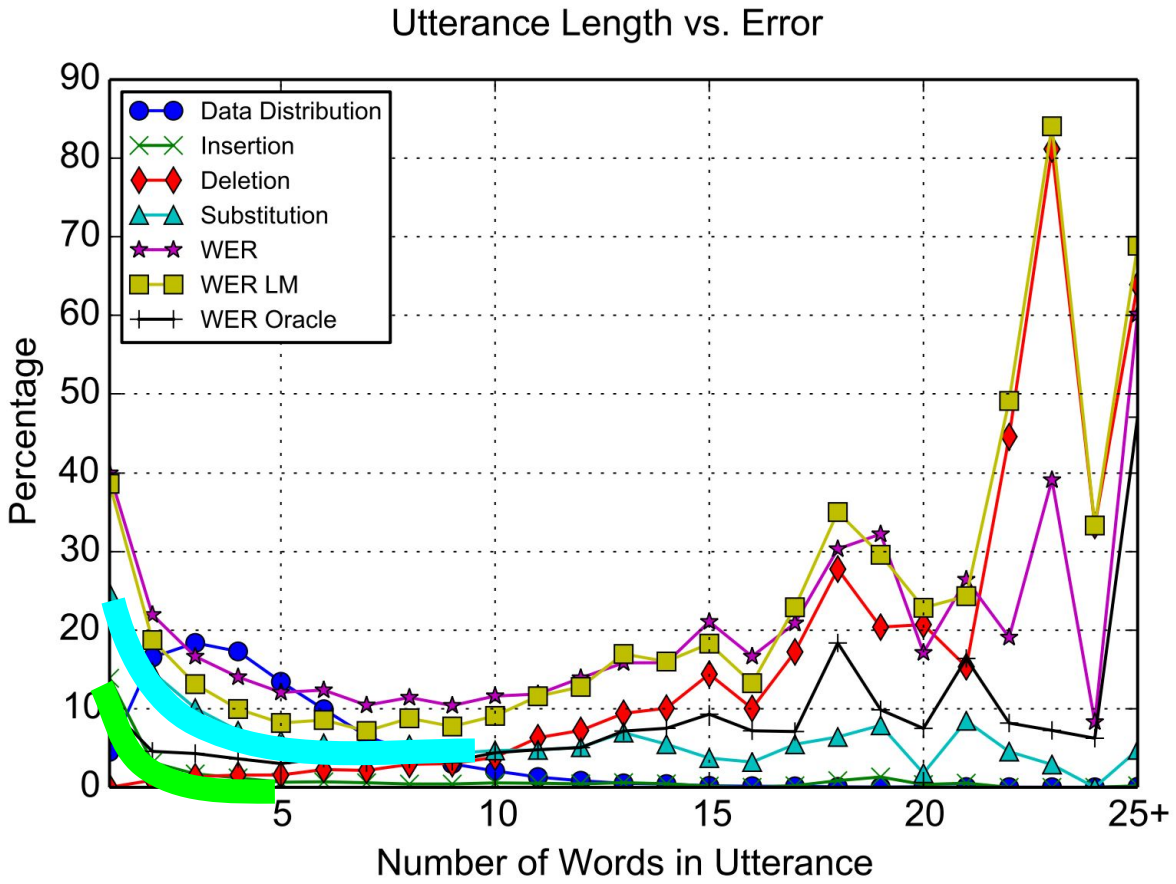


★ WER

■ WER (rescoring)

┼ WER (oracle)

Experiments - Effects of Utterance Length



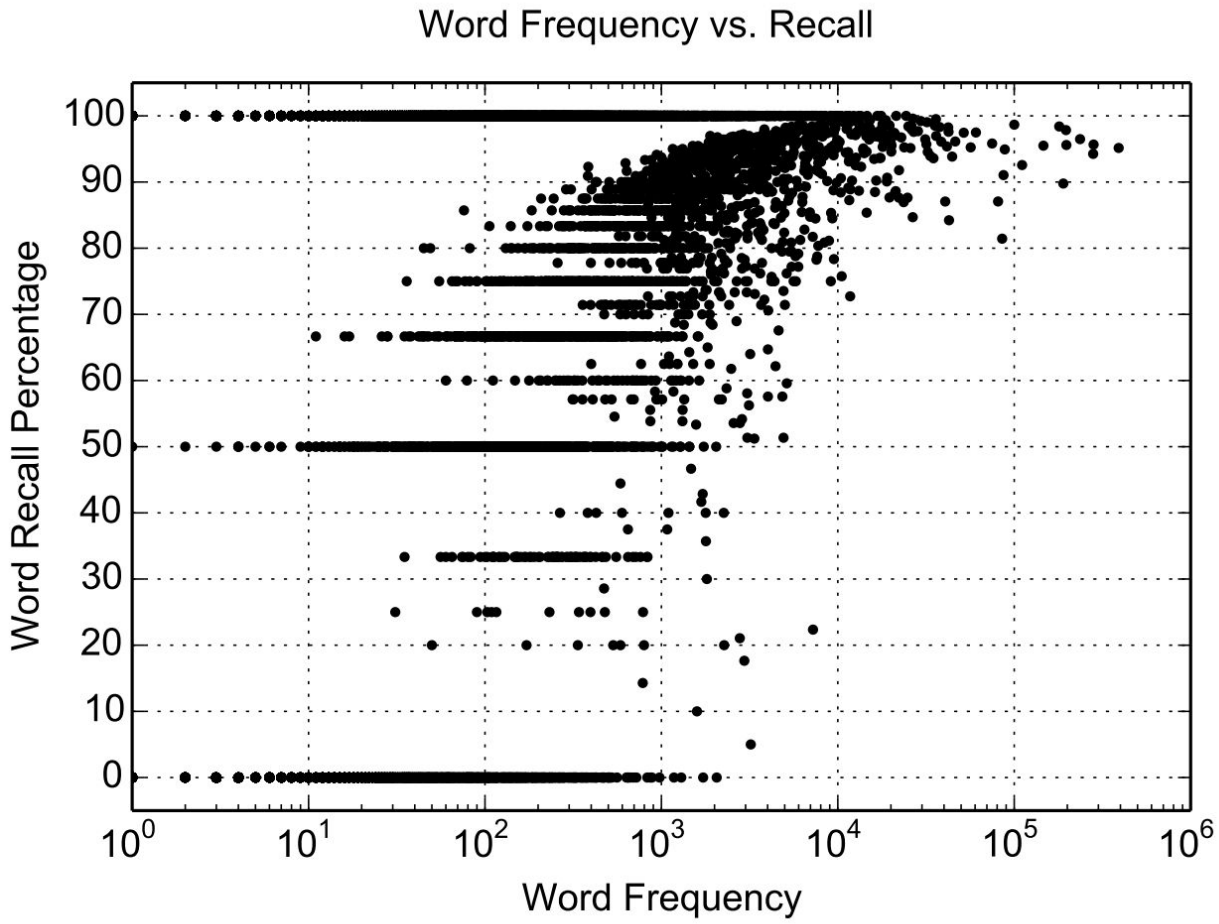
★ WER

■ WER (rescoring)

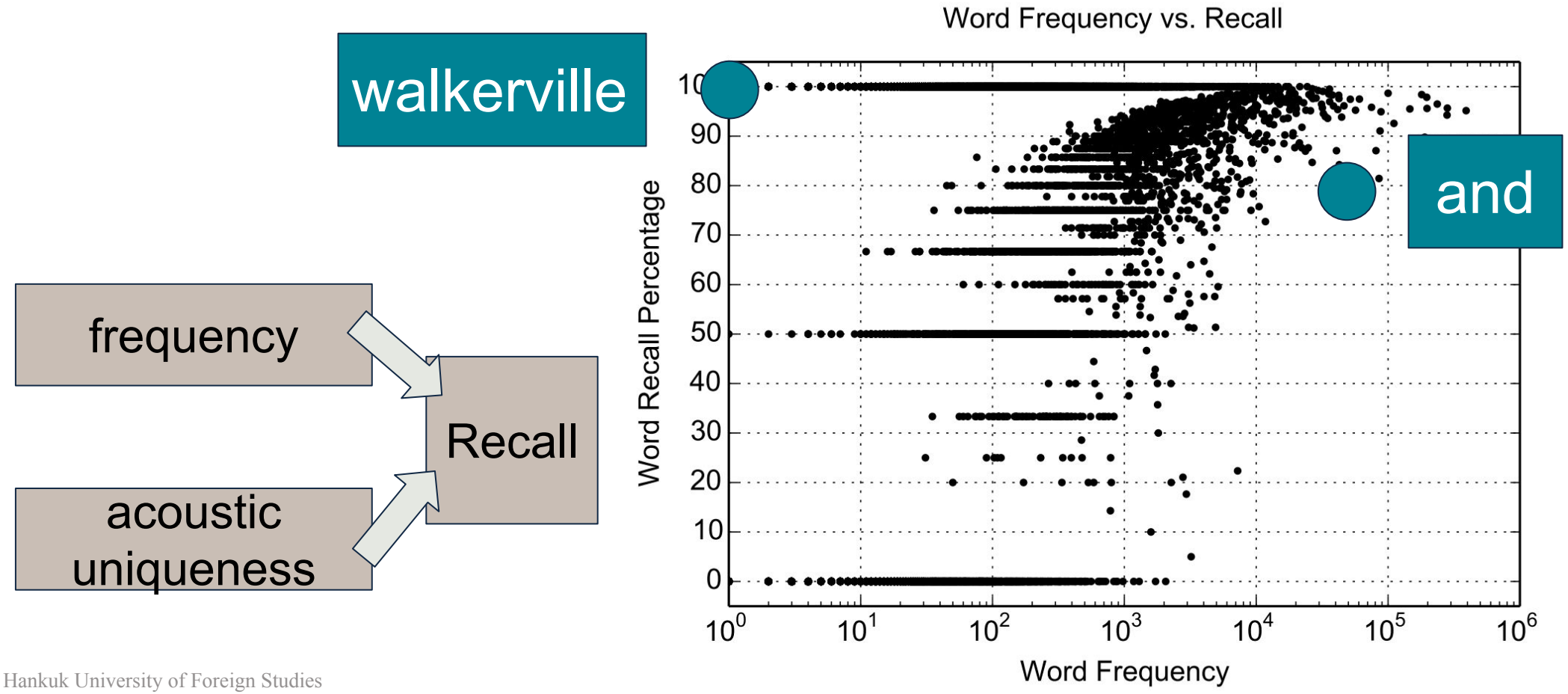
┼ WER (oracle)

Experiments - Word Frequency

recall
= 맞춘 정답/실제 정답



Experiments - Word Frequency



Experiments - Interesting Decoding Examples

1. One Acoustic → Multiple Spelling Variants

"Triple A"

Beam	Text	Log Probability	WER
Truth	call aaa roadside assistance	-	-
1	call aaa roadside assistance	-0.5740	0.00
2	call triple a roadside assistance	-1.5399	50.00
3	call trip way roadside assistance	-3.5012	50.00
4	call xxx roadside assistance	-4.4375	25.00

Experiments - Interesting Decoding Examples

1. One Acoustic → Multiple Spelling Variants

"Triple A"

Beam	Text	Log Probability	WER
Truth	call aaa roadside assistance	-	-
1	call aaa roadside assistance	-0.5740	0.00
2	call triple a roadside assistance	-1.5399	50.00
3	call trip way roadside assistance	-3.5012	50.00
4	call xxx roadside assistance	-4.4375	25.00

Experiments - Interesting Decoding Examples

2. Handling Utterance with Repeated Words

"seven seven seven"

Beam	Text	Log Probability	WER
Truth	eight nine four minus seven seven seven	-	-
1	eight nine four minus seven seven seven	-0.2145	0.00
2	eight nine four nine seven seven seven	-1.9071	14.29
3	eight nine four minus seven seventy seven	-4.7316	14.29
4	eight nine four nine s seven seven seven	-5.1252	28.57

II Literature Review: Further Studies

모델	장점	단점
LAS 2015	최초의 완전한 음성 end-to-end model 문맥 파악 능력이 뛰어남	실시간 streaming 불가 학습 수렴이 어려움
RNN-T 2018(상용화)	음성이 들어오는 즉시 출력 가능 (streaming), 실시간 서비스 최적	순차적 처리 (학습 속도 저하, 긴 문장 맥락 파악 한계)
Conformer 2020	CNN+Transformer → 전역/지역 특징 동시 파악 (최고 성능)	긴 입력에 대한 연산 복잡도 높음 실시간 적용 시 별도 최적화 필요
Wav2Vec 2.0 2020	대규모 non labeling data pre-training 적은 학습 데이터로 fine-tuning	매우 높은 사전 학습 비용 추론 시 연산 부하가 큼

what's next?

1. 프로젝트 개요

ESPnet과 LibriSpeech를 활용한 End-to-End ASR 모델 학습 및 데모 구

1.1 실험 환경

- **GPU : NVIDIA RTX 5090 (32GB VRAM)**
- **Framework : ESPnet2, PyTorch 2.9.1, Python 3.11**

1.2 핵심 과제

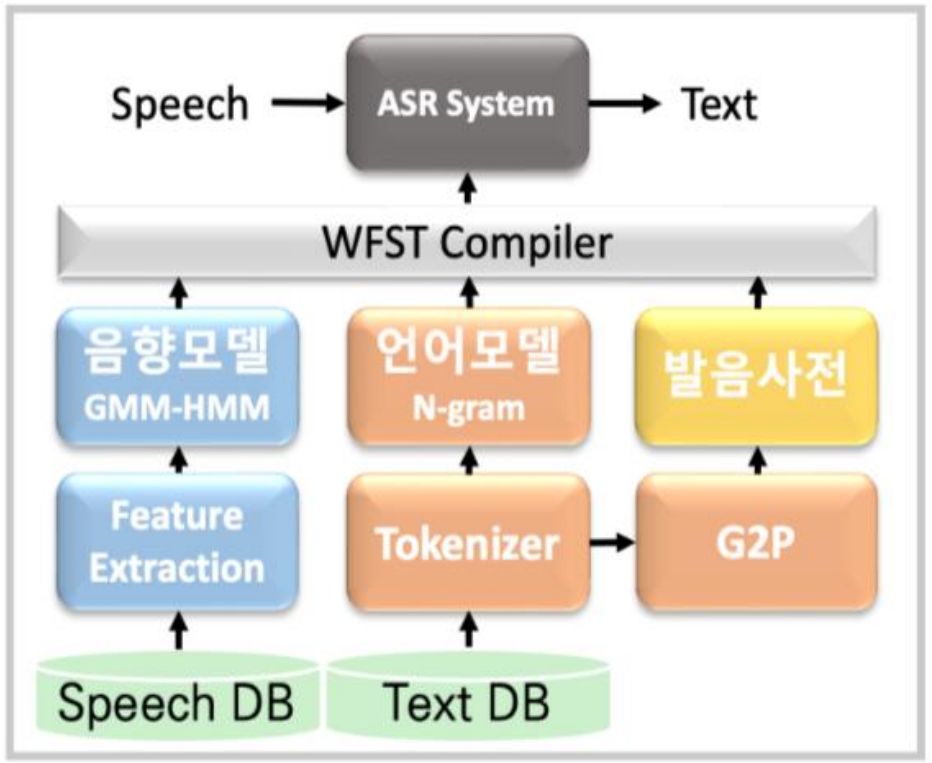
- 기존 ESPnet 레시피는 대규모 클러스터(Multi-GPU) 환경에 최적화되어 있어, 실습 환경인 단일 GPU(RTX 5090)에서는 **메모리 부족(OOM) 문제 발생.**

-> **Hyperparameter Optimization**

- 고성능 모델(Conformer/Transformer)을 제한된 자원에서 학습시키기 위한 **경량화 및 학습 파이프라인 재설계** 필요

-> **Training Stability**

음성 인식 (Speech-to-Text, STT)

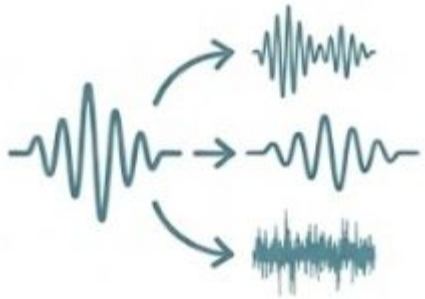


구성요소	전통 방식	ESPnet
음향모델	GMM-HMM	Conformer 인코더 (12 blocks)
언어모델	N-gram	Transformer LM (8 layers)
발음사전	수동 구축 필요	불필요 (BPE 사용)
토크나이저	단어/음소 단위	BPE 5000 서브워드
디코딩	WFST 그래프 탐색	Beam Search
학습	각 모듈 따로 학습	End-to-End 통합 학습
특징	모듈별 최적화	전체 시스템 Joint 최적화

2. 데이터 증강 및 전처리

2.1 Data Augmentation (데이터 증강)

- Speed Perturbation : 원본 음성의 속도 0.9배, 1.0배, 1.1배로 변형
- SpecAugment : Time warping, Time/Frequency Masking



2.2 Feature Extranction (특징 추출)

- Input : Raw Waveform (16kHz)
- Feature : 80-dim Log-Mel Spectrogram
- Normalization : Global MVN(Mean Variance Normalization) 적용

2.3 Tokenization (토큰화)

- SentencePiece를 활용한 BPE(Byte Pair Encoding) 적용 (Vocab Size: 5000)

3. LM Training Optimization

- **Problem**
  기본 설정(Batch bins: 500M, Layers: 16, GPU x16 기준) 적용 시 **GPU 메모리 초과(OOM) 발생**

- **Optimization Strategy:**
  a. **Batch Resizing :** 메모리 한계를 고려하여 batch_bins를 500M -> 30M (1/16)
  b. **Gradient Accumulaton** : accum_grad 2 -> 4 **(Effective batch size 보완)**
  c. **Model Compression :** Transformer Layer 16 -> 8로 경량화
  d. **Regularization :** 과적합 방지를 위해 Dropout(0.1) 추가 및 Early Stopping 도입
  e. **Epoch 조정** : 25 -> 8 epoch

```
=== LM 8 Epoch 최종 결과 ===
[train] loss=0.912
[valid] loss=3.561 (PPL=35.2)
[total] time=9시간 43분 (35039초)
```

3. ASR Model Training Optimization

- **Model Architecture**
  - **Encoder : Conformer (12 Blocks)** - CNN과 Transformer 장점 결합(local/global 특징 동시 포착)
  - **Decoder : Transformer (6 Blocks)**
  - **Loss : CTC/Attention Hybrid (CTC 30% + Attention 70%)**

- **Single-GPU Tuning Strategy**
  a. **Memory Management :** batch_bins 35M -> 15M으로 조정하여 OOM 방지
  b. **Gradient Accumulation :** accum_grad 4 -> 8로 2배 증가 **(Effective batch size)**
  c. **Epoch 최적화 :** Max Epoch 50 -> 40으로 조정

```
=== ASR 40 Epoch 최종 결과 ===
[train] loss=3.206, acc=88.9%
[valid] loss=5.018, acc=96.1%, wer=40.9%
[total] time=50시간 42분 (1-35 epoch: 159,493초 + 36-40 epoch: 23,058초)
```

4. Evaluation

4.1 WER (Word Error Rate) 계산

WER = (S + D + I) / N × 100%

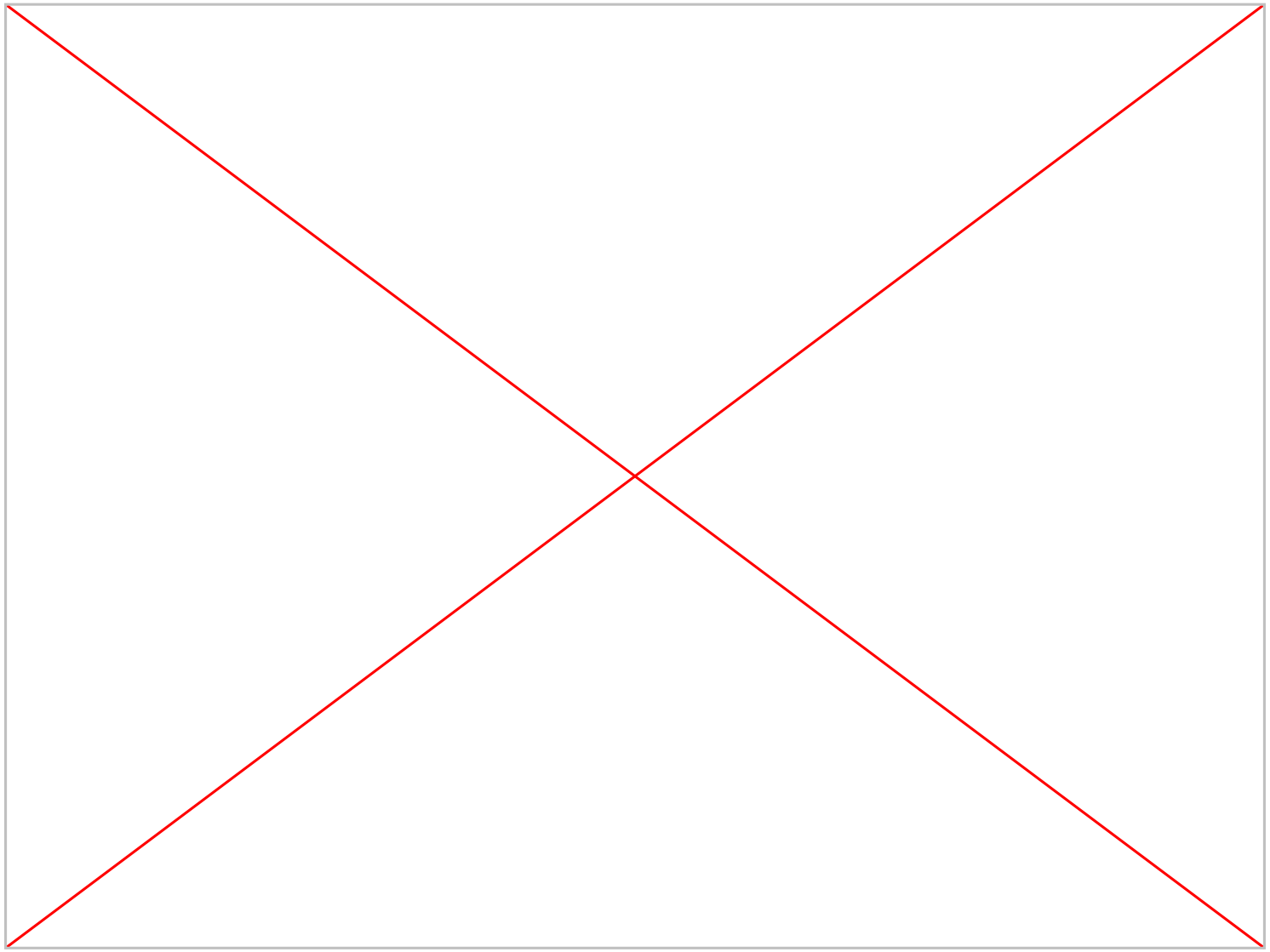
S: Substitution (대체 오류)
D: Deletion (삭제 오류)
I: Insertion (삽입 오류)
N: 정답 문장의 총 단어 수

4.2 최종 성능(WER)

테스트셋	WER
test_clean	2.28%
test_other	4.88%

테스트셋	샘플 수	총 단어 수	특징
test_clean	2,620	52,576	깨끗한 녹음 환경
test_other	2,939	52,343	노이즈/악센트 포함

III ASR Model: Inference



III ASR Model: Conclusion

1. **Project Summary:** 단일 GPU(RTX 5090) 환경에서 최신 ASR(Conformer) 모델 학습 성공

2. **Technical Achievement:**

- **Optimization:** Batch Resizing & Gradient Accumulation을 통한 자원 한계 극복
- **Performance:** WER 2.28% (Clean), 4.88% (Other) 달성

Thank You!

Q&A