

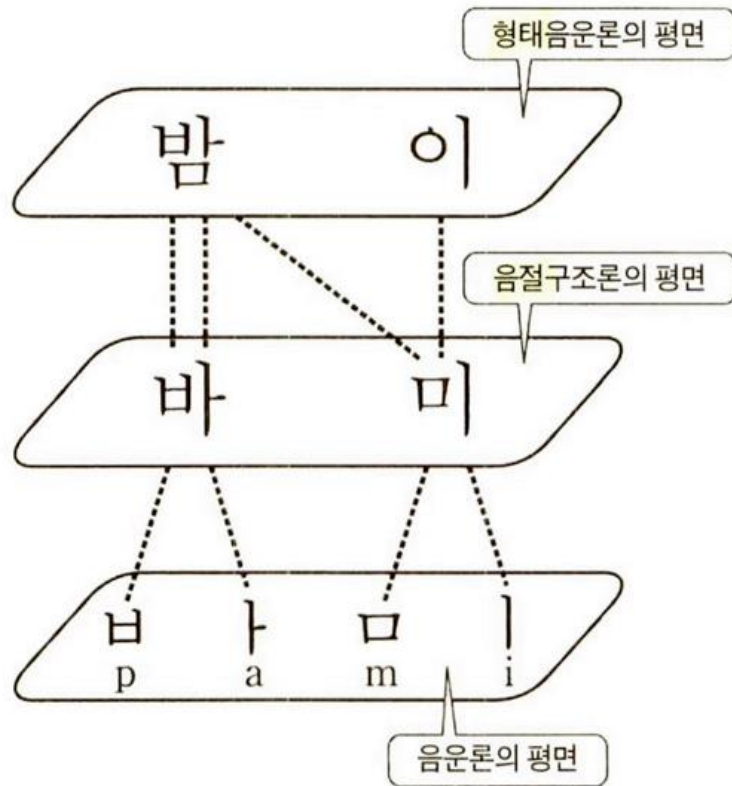
음성인식 시작하기

구자현, 임성민

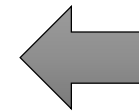
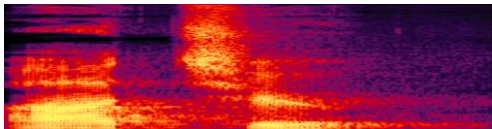
SSSCLAB @KAIST

sssclab.kaist.ac.kr

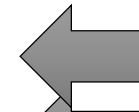
ASR (Automatic Speech Recognition)?



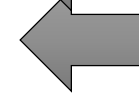
노마 히데키, "한글의 탄생", 2011.



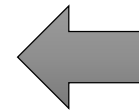
(문법에 맞는) 단어열=문장



문자열 (한글자모, 알파벳, ...)



음소 (발음기호)



음성 (음향학적 신호)

Goal

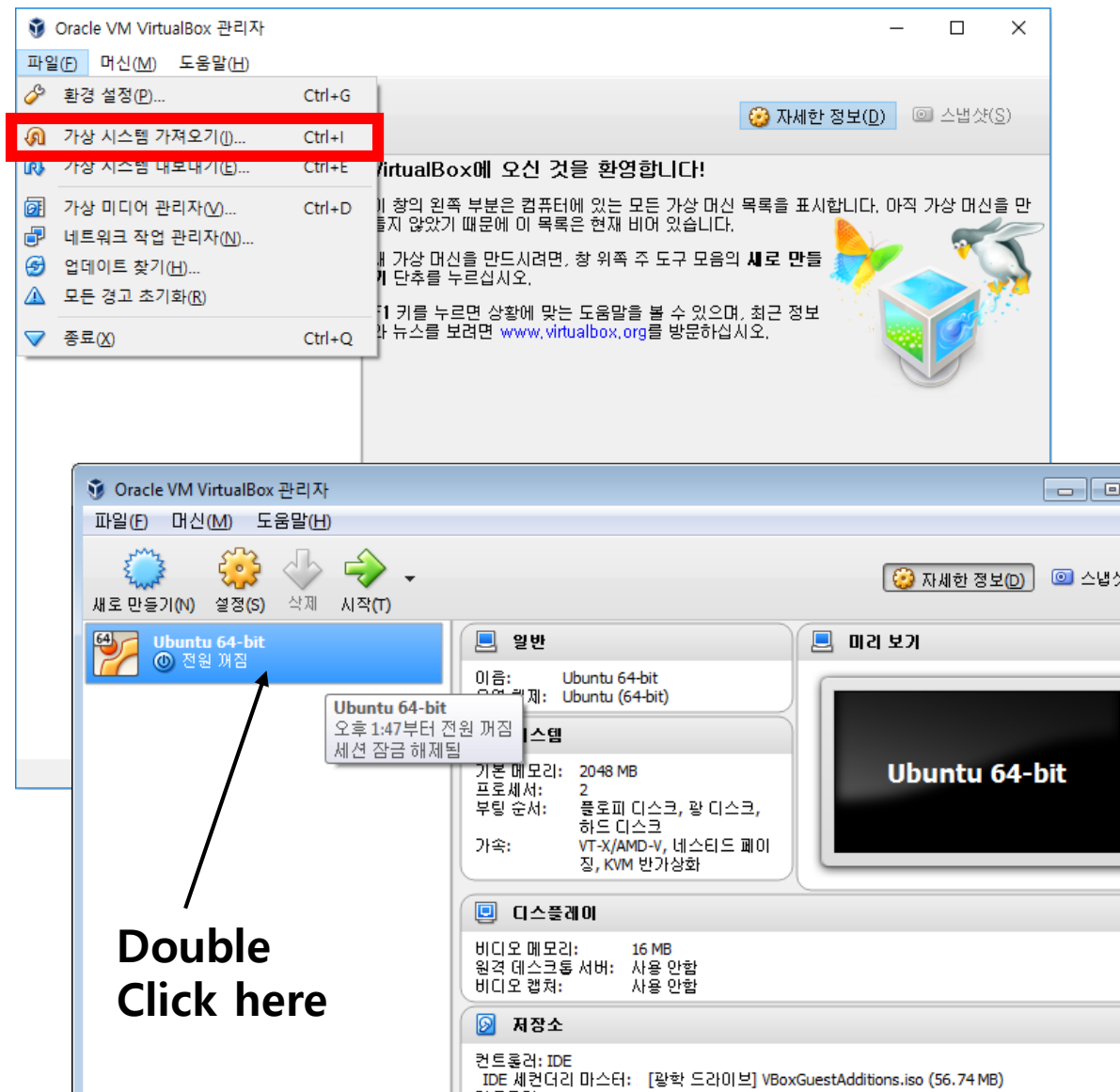
- Ubuntu 환경에서 음성인식/화자인식용 공개 SW인 Kaldi 사용법을 익힌다
- 고전적 방법론의 음성인식 시스템을 구성하는 모듈을 이해한다
- 공개 digit database인 FSDD를 이용하여 고전적 방법론의 음성인식 시스템을 구축한다
- 보다 현대적인 음성인식 모델에 대해 이해하고, Kaldi에서 이를 구축하는 방법을 고안한다

실습. 가상 Ubuntu 실행

- 가상 시스템 가져오기 / 실행
- *우분투 로그인*
- *터미널 사용법 학습*

실습용 VirtualBox 이미지 실행

- 적당한 장소에 실습용 이미지 복사
(ubuntu_kaldi_tutorial.ova)
- Windows에서 Oracle VM VirtualBox 실행
- [파일] - [가상 시스템 가져오기]
(오른쪽 그림 참조)
- 미리 복사한 실습용 .ova 이미지파일
선택하여 가져오기 진행
- 가져오기 완료시 [Ubuntu 64-bit]
실행하여 가상머신 기동



ASR toolkit



- 대표적인 음성인식/화자인식용 오픈소스 툴킷
- Shell script, C++ (및 일부 python)
- 음성인식에 필요한 전처리 및 후처리에 필요한 여러 Library들이 갖춰져 있음
- 다양한 음성 데이터에 맞는 recipe/tutorial이 존재
- 이번 실습에서 사용

<http://kaldi-asr.org>

<https://github.com/kaldi-asr/kaldi>

그 외

CMU Sphinx : 2000년대 사용하던 오픈소스 툴킷. C/Java/Python

HTK : 과거 주로 쓰였던 HMM 기반 음성인식 툴킷으로,
사용하려면 라이선스 필요. C언어로 구성됨

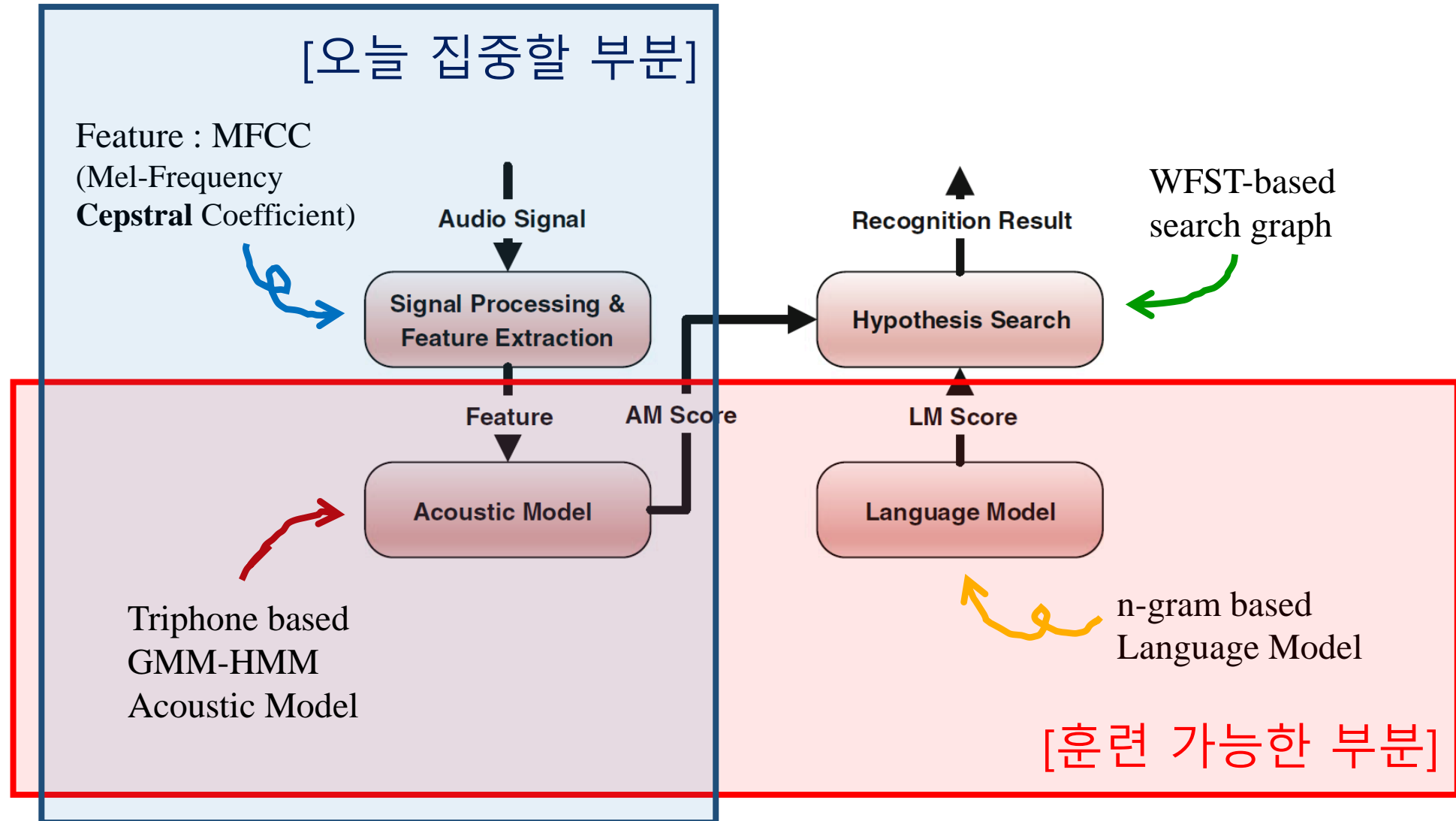
최근에 딥러닝 기술의 발전으로 전/후처리를 모두 하나의 신경망에 포함시킨
End-to-End 모델이 (Tensorflow/Torch 등을 이용하여) 다양하게 연구되고 있지만,
충분한 성능을 내기 위해선 훨씬 많은 데이터가 필요

Database

- FSDD (Free Spoken Digit Dataset)
 - “음성인식을 위한 MNIST”
 - Digit recognition task (0~9)
 - 총 1,500개의 8kHz .wav 파일
 - 3 English speaker, 50 utterances for each “zero”, “one” .. “nine”.
 - Spoken digit recognition
 - 매우 단순한 task로, 고전적 시스템의 성능 검증에 사용
 - 빠른 훈련 및 검증이 가능하여 본 tutorial에서 활용

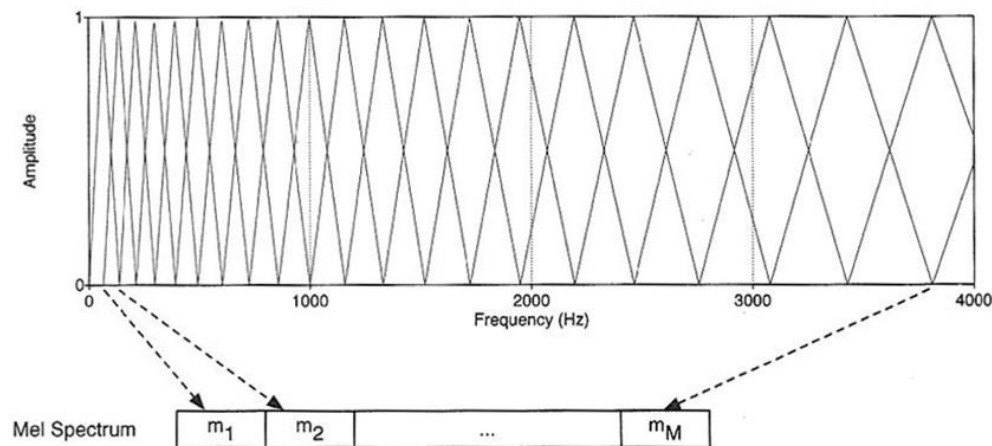
<https://github.com/Jakobovski/free-spoken-digit-dataset>

Classic ASR system

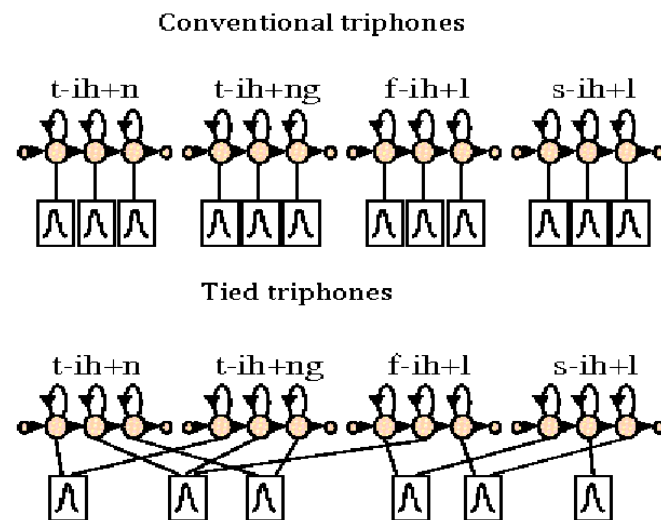


Miscellaneous

- MFCC features?
 - GMM-HMM 등의 고전적 방법론에서 주요하게 사용
 - Mel-filterbank에 DCT를 적용하여 스펙트럼의 주기성을 한 번 더 분석
 - 음성 특징을 저차원에 잘 요약하며, 성분간 의존성이 낮음



- Triphone?
 - 같은 음소라도 전후 문맥에 따라 신호 특성이 달라질 수 있음
 - 이를 반영하여 앞 1개, 뒤 1개 음소에 따라 같은 음소를 다르게 모델링
 - 가능한 경우의 수가 커 실제로는 HMM state 단위에서 tying



Miscellaneous

- n-gram?

- 문법을 확률적으로 모델링하는 데 사용되는 가장 고전적인 방법으로, 단어 간 관계를 단순한 확률로 모델링
- 3-gram LM의 경우 특정 언어, 특정 문법에서 각 단어가 나올 확률을 직전 2개 단어에만 종속되도록 정의

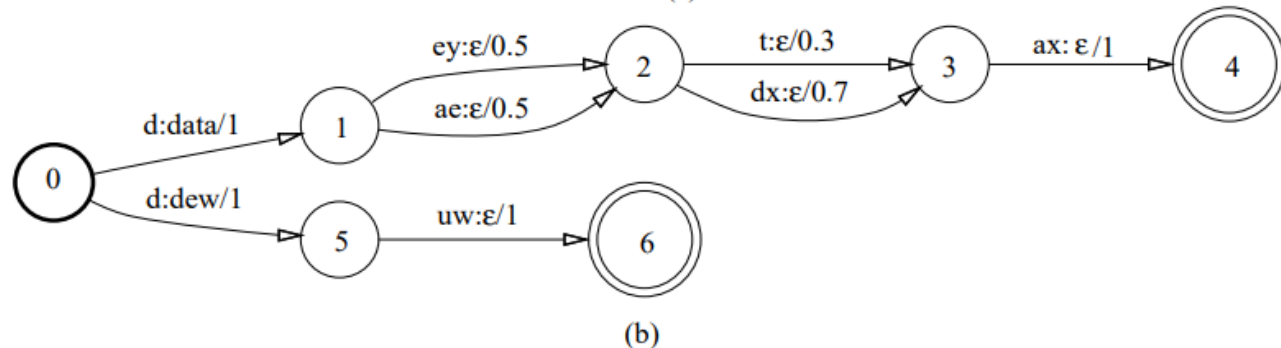
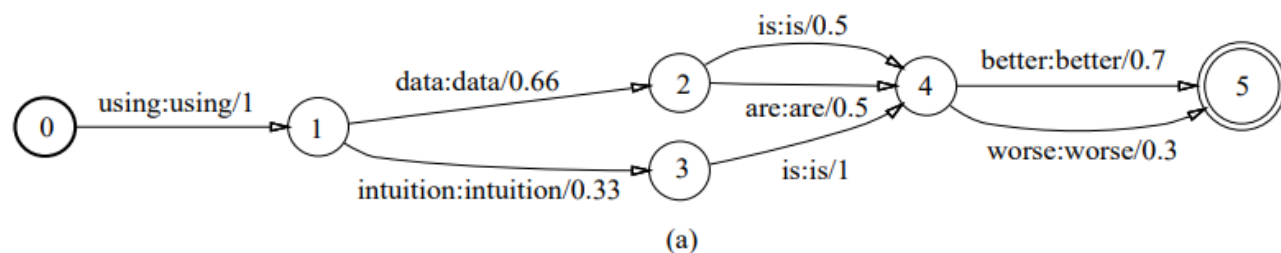
- $P(\text{사랑해} \mid \text{난, 널}) = 0.1$

- “난 널 [...]”라는 단어열의 [...] 부분에 “사랑해” 단어가 들어갈 확률이 0.1

- 대량의 text data를 통해 학습

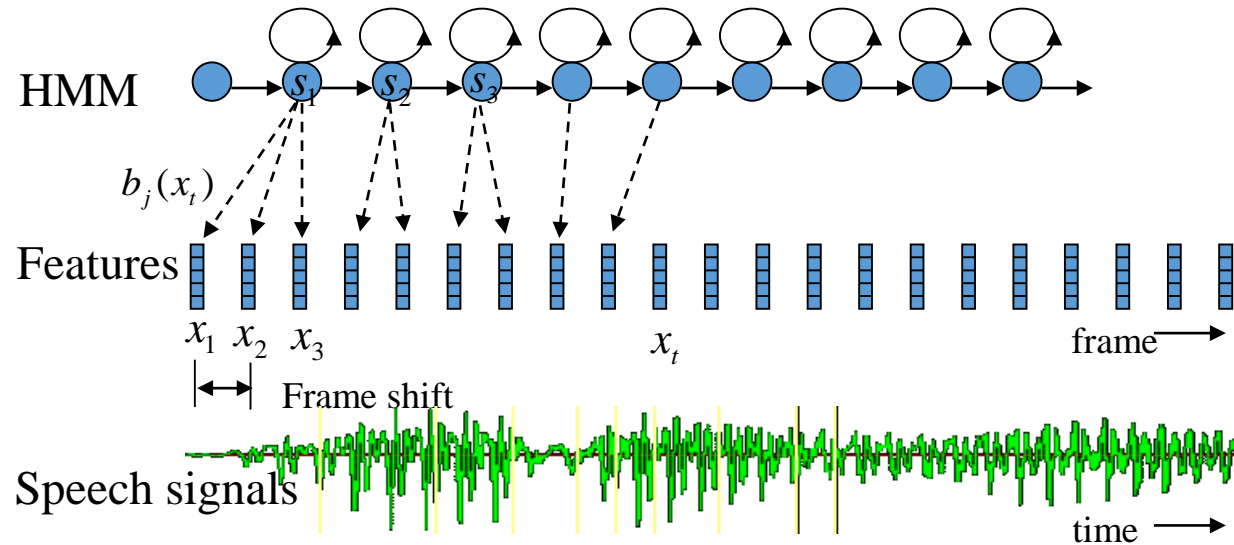
- WFST?

- 매 arc마다 천이 확률이 정의되고, 시작과 끝이 정해진 유한 오토마타 그래프
- 단계별로 정의된 WFST는 결합이 용이해 따로 만들어진 음향모델과 발음사전, 언어모델을 결합하는 데 사용



How to train **Acoustic Model**?

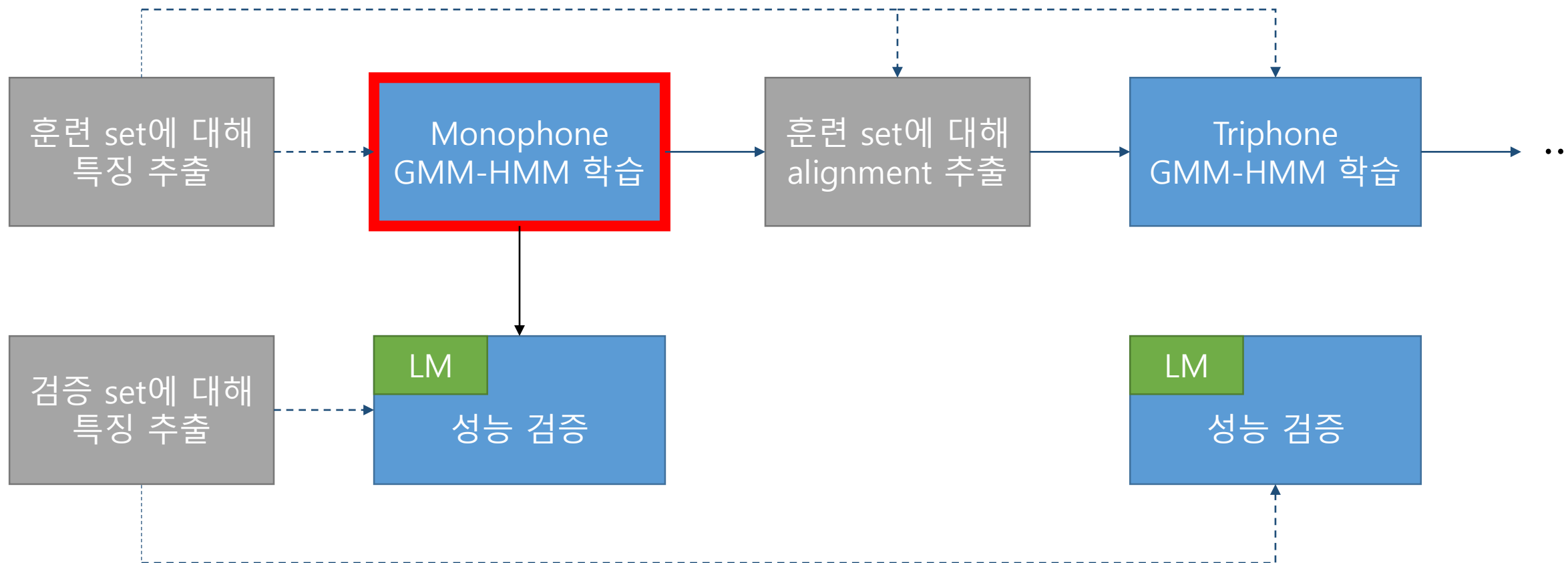
- 훈련할 때는 훈련용 데이터에 대해, 매 10ms 길이 frame마다 신호가 "어떤 음소에 속하는지" 정보를 줄 필요가 있음



- 매 음성 frame이 어떤 음소에 속하는지 분류하려면, 미리 훈련된 AM 필요
→ ERROR: Circular reference (이는 딥러닝 기반 system에서도 동일)

How to train **Acoustic Model**?

- 고전적 system의 필요성
: **Monophone** GMM-HMM AM은 처음부터 (from scratch) 학습 가능하다!

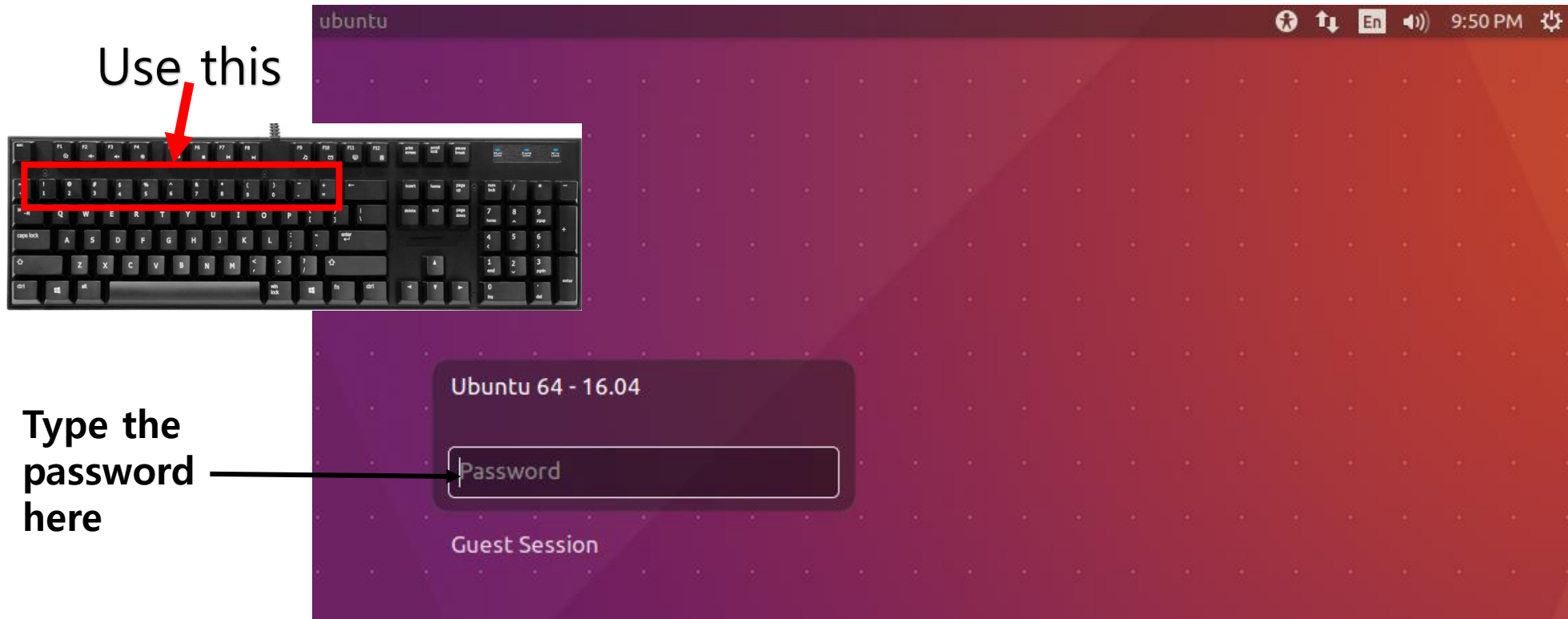


실습. 가상 Ubuntu 실행

- *가상 시스템 가져오기 / 실행*
- 우분투 로그인
- 터미널 사용법 학습

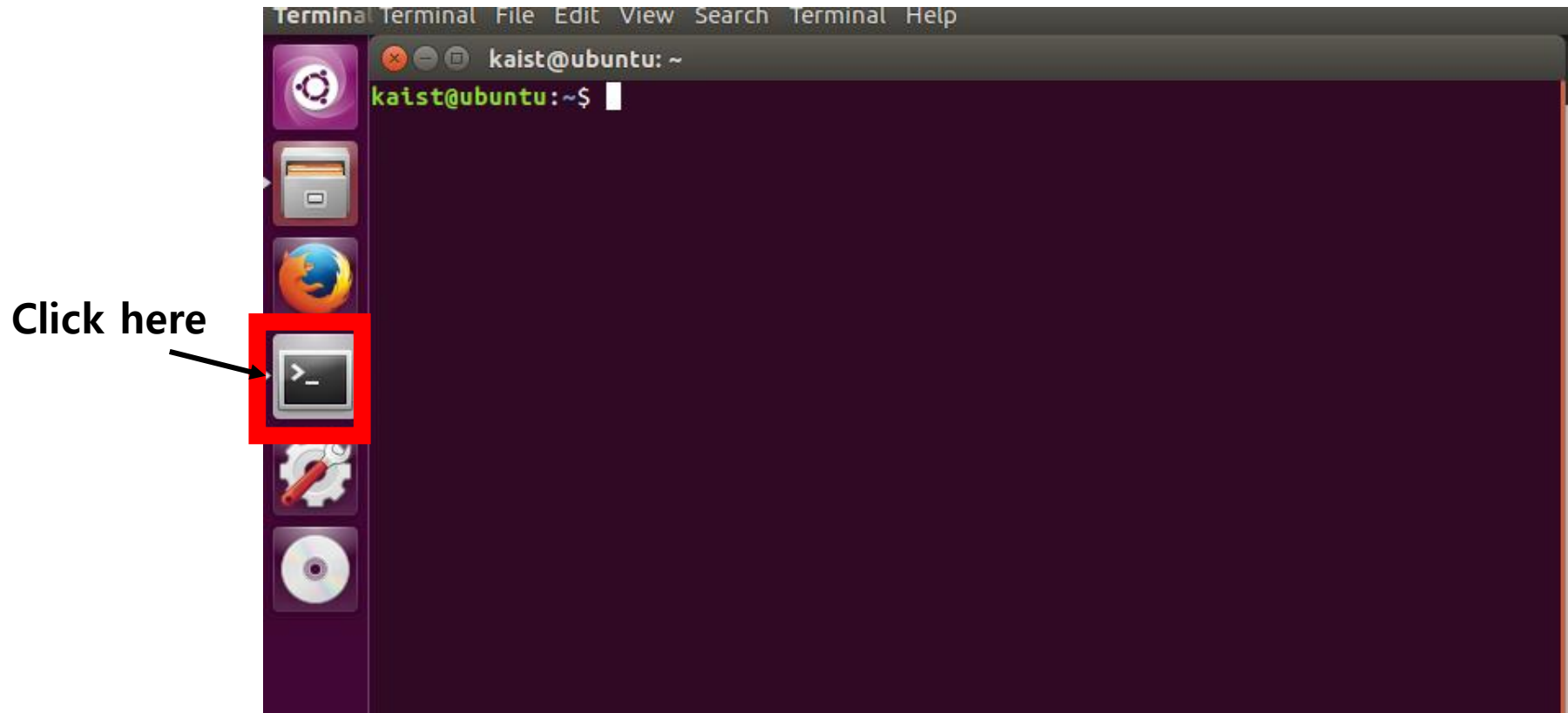
Ubuntu 로그인

- 가상 이미지 로드 후 "123450"으로 접속
 - 주의: 키보드 왼쪽 위 숫자키를 통해 입력



Ubuntu 로그인

- 왼쪽 사이드바 터미널 아이콘을 통해 터미널 열기
 - 리눅스 터미널에 익숙하지 않은 경우, 다음 장 명령어 참조하여 여러 폴더 오가기 연습



Ubuntu Terminal 명령어 일람

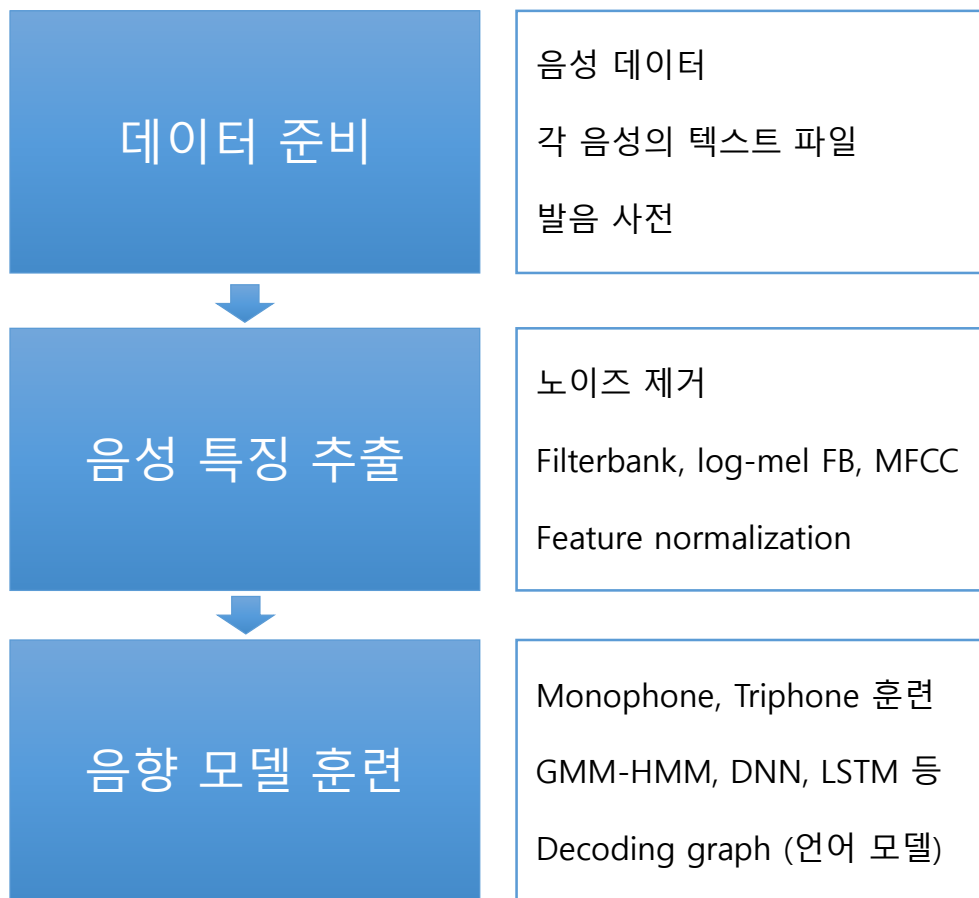
- 파일 관련
 - ls (list) – 현재 디렉토리 파일 리스트 출력
 - cd (change directory) – 디렉토리 이동
 - mv (move) – 파일 옮기기
 - cp (copy) – 파일 복사
 - rm (remove) – 파일 삭제
 - pwd (print working directory) – 현재 위치 출력
(TIP : tab키를 통해 자동 완성 가능)
- 해보기
 - 터미널 실행하자마자 나오는 위치는 어디인가?
 - 터미널에서 kaldi/egs/fsdd로 이동하여 파일 목록을 출력하라

실습. Digit recognition

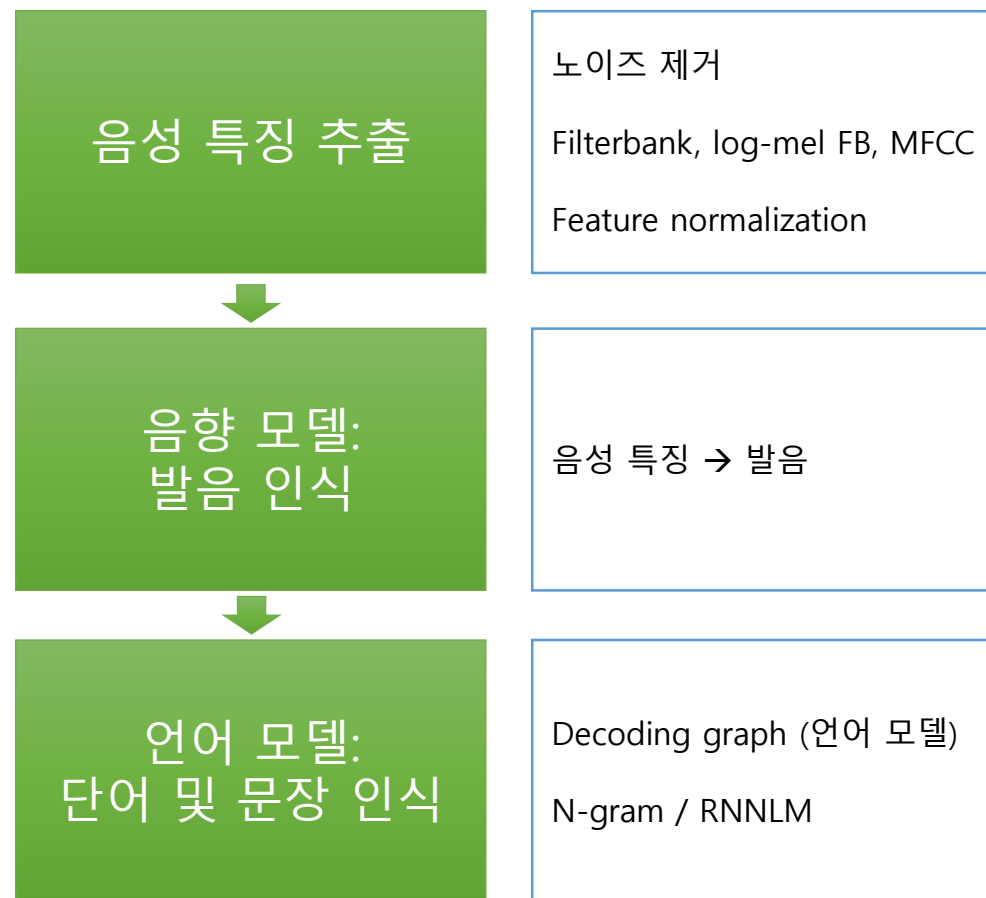
- FSDD용 기본 Kaldi recipe 이해
- 실행 및 실험 결과 분석
- 더 나은 결과를 만드는 법

음성인식 모델의 구조

음성인식 모델 훈련

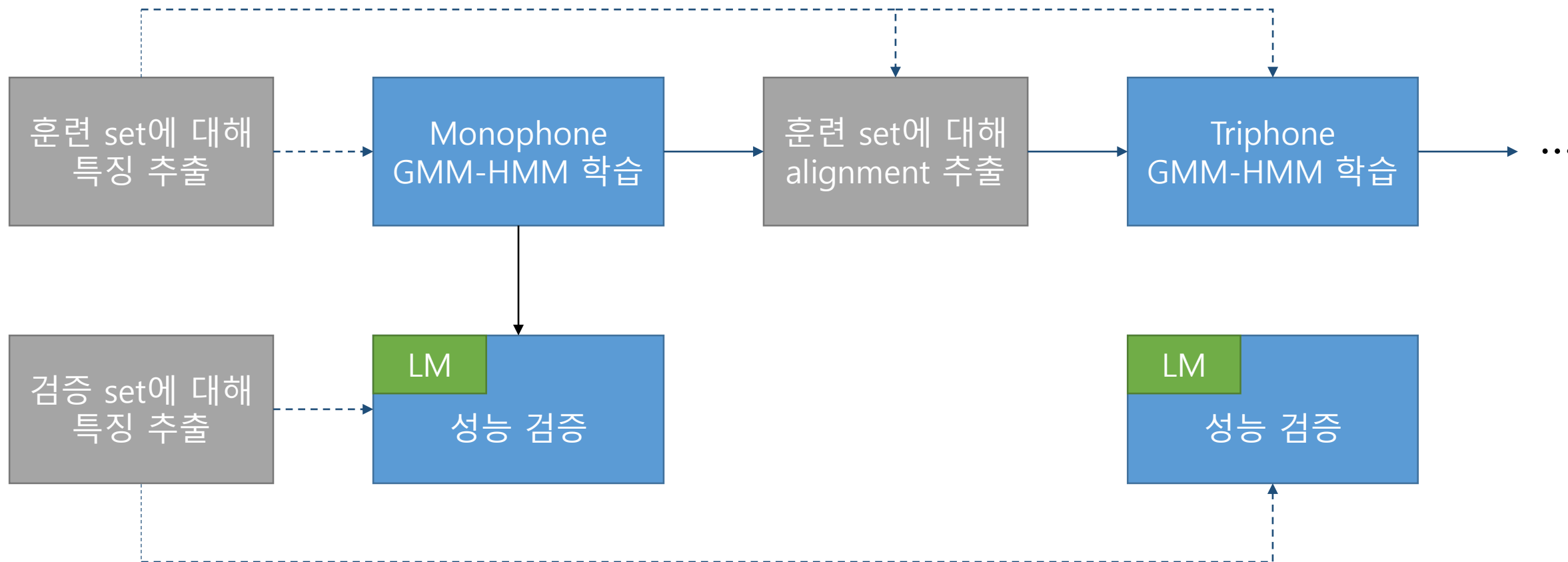


음성인식 모델을 통한 인식



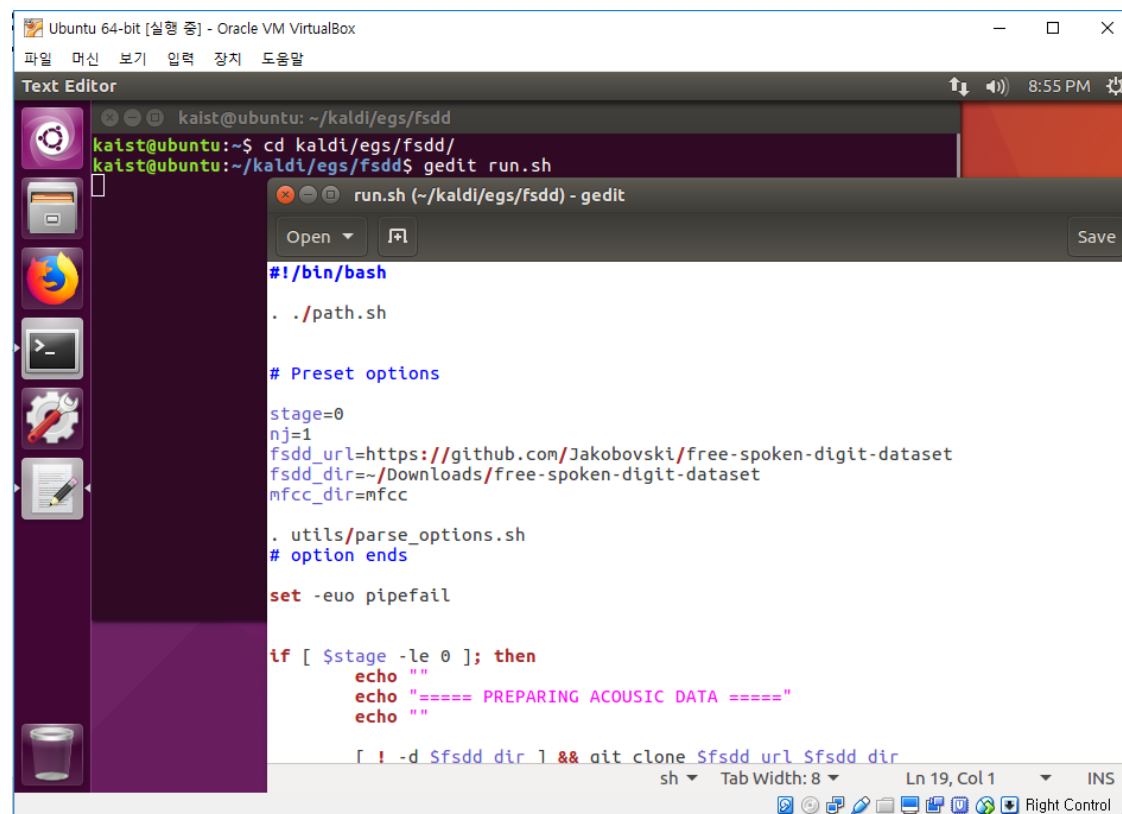
음성인식 모델의 구조

- 본 실습에서는 고전적 음성인식 모델에서, 아래 부분을 학습
 - Kaldi에서는 웬만한 음성인식용 데이터에 대하여 아래를 포함하는 recipe를 제공



훈련 스크립트 확인

- 내장된 GUI text editor를 통해 훈련 스크립트 확인
 - 터미널 실행 후 `kaldi/egs/fsdd`로 이동 (`$ cd kaldi/egs/fsdd`)
 - 내장 유틸인 `gedit`을 통해 스크립트 체크 (`$ gedit run.sh`)



```
kaist@ubuntu: ~/kaldi/egs/fsdd
kaist@ubuntu:~$ cd kaldi/egs/fsdd/
kaist@ubuntu:~/kaldi/egs/fsdd$ gedit run.sh

run.sh (~/.kaldi/egs/fsdd) - gedit

#!/bin/bash

. ./path.sh

# Preset options
stage=0
nj=1
fsdd_url=https://github.com/Jakobovski/free-spoken-digit-dataset
fsdd_dir=~/.Downloads/free-spoken-digit-dataset
mfcc_dir=mfcc

. utils/parse_options.sh
# option ends

set -euo pipefail

if [ $stage -le 0 ]; then
    echo ""
    echo "===== PREPARING ACOUSIC DATA ====="
    echo ""

    [ ! -d $fsdd_dir ] && git clone $fsdd_url $fsdd_dir
```

훈련 스크립트 확인

- 매 stage 별 역할

- Stage 0 : PREPARING ACOUSIC DATA
 - FSDD 데이터셋 다운로드
 - 데이터 목록을 Kaldi style로 전처리 (훈련셋과 검증셋 분리 과정 포함)
- Stage 1 : FEATURE EXTRACTION
 - MFCC 특징 추출
- Stage 2 : PREPARING LINGUISTIC DATA
 - 발음사전, 단어 목록, 음소 목록 준비
 - 언어정보 목록을 Kaldi style로 전처리
- Stage 3 : PREPARING LANGUAGE MODEL
 - "Uniform 1-gram" LM을 ARPA format으로 생성
 - ARPA format의 LM을 WFST graph로 변환

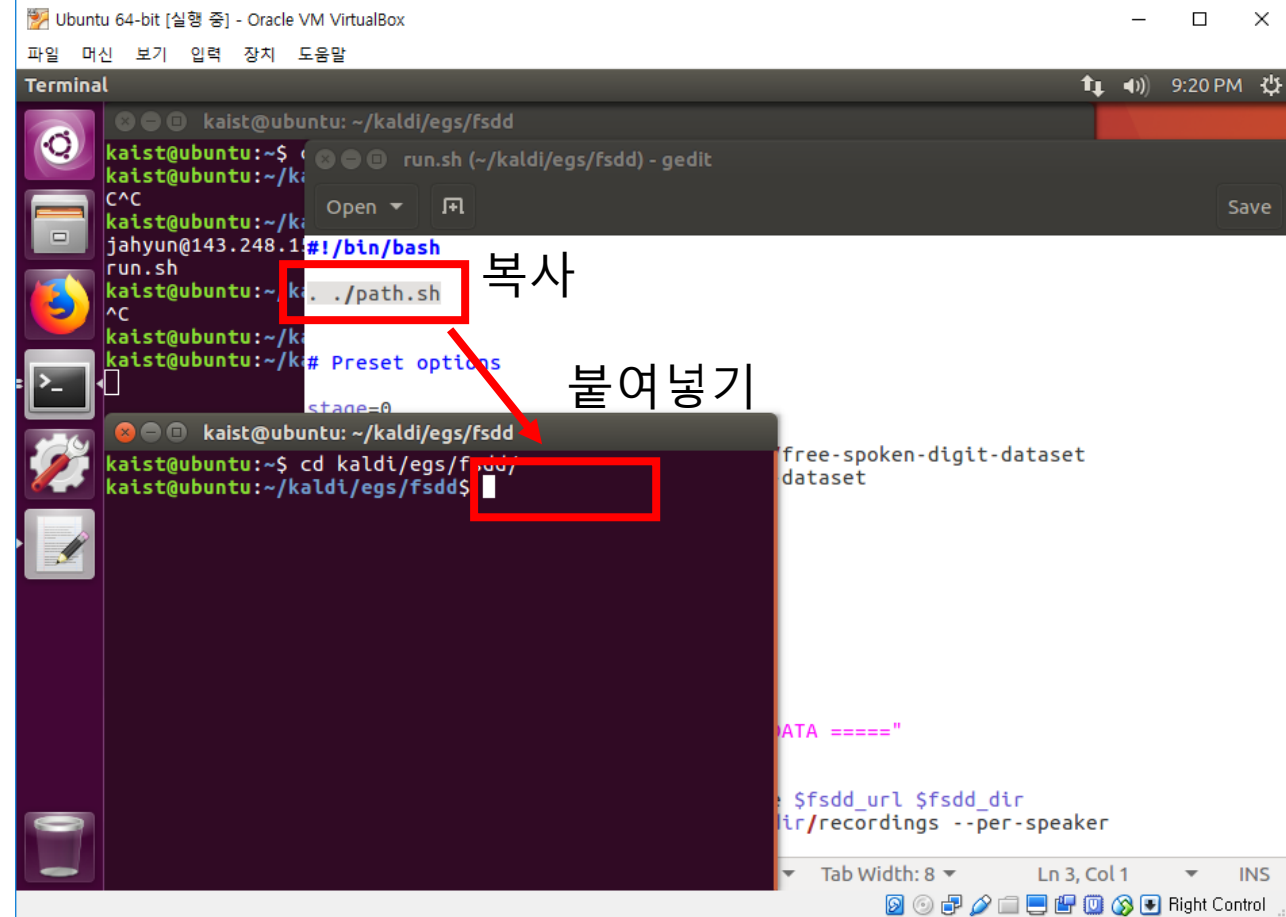
- Stage 4 : MONOPHONE MODEL TRAINING
- Stage 5 : MONOPHONE MODEL DECODING
- Stage 6 : MONOPHONE MODEL ALIGNMENT
 - Monophone GMM-HMM AM 훈련
 - 훈련한 AM과 사전에 준비한 발음사전 및 LM을 가지고 검증 데이터셋을 디코딩
 - Triphone GMM-HMM AM 훈련을 위해 훈련한 Monophone AM으로 훈련 데이터셋을 정렬
- Stage 7 : TRIPHONE MODEL TRAINING
- Stage 8 : TRIPHONE MODEL DECODING
 - Triphone GMM-HMM AM 훈련 ("senone" 500개, Gaussian 2000개)
 - 검증 데이터셋을 디코딩

훈련 스크립트 실행

- run.sh 실행

- 스크립트는 한번에 돌아가도록 설계되었으나, 처음에는 stage별로 직접 터미널에 입력하며 실행할 것을 권장
 - 확실하지 않을 때는 조교를 불러주세요

- Shift + 클릭을 통해 새 터미널 실행
- #주석과 (stage를 가리키는) if 절은 제외하고 매 줄을 직접 입력하여 실행
 - 터미널에서는 "붙여넣기" 단축키가 Shift+Insert임에 유의
 - For 절과, 맨 뒤에 \ (backslash) 입력 후 들여쓰여진 절은 한꺼번에 입력할 것
- 매 stage를 실행 후 파일 목록을 확인해가며 어떤 파일이 생기는지 체크



```
ngram-count -order 1 \  
-write-vocab data/local/tmp/vocab-full.txt \  
-wbdiscount -text data/local/dict/words.txt \  
-lm data/local/tmp/lm.arpa
```

예컨대 위는 실제로는 아래와 같이 한 줄로 입력하여야 함.

```
ngram-count -order 1 -write-vocab data/local/tmp/vocab-  
full.txt -wbdiscount -text data/local/dict/words.txt -lm  
data/local/tmp/lm.arpa
```

실습. Digit recognition

- *FSDD용 기본 Kaldi recipe 이해*
- *실행 및 실험 결과 분석*
- 더 나은 결과를 만드는 법

결과 분석

- 두 모델에 대해 디코딩할 때마다 인식 결과가 WER로 나온다
 - Stage 5 : MONOPHONE MODEL DECODING
 - Stage 8 : TRIPHONE MODEL DECODING
- AM과 LM의 결합 비율에 따라 총 11개의 WER 확인 가능
 - Kaldi의 '빠른' decoder 구현에는 random access가 포함되므로 실행할 때마다 성능이 달라질 수 있음

- WER?
 - Word Error Rate의 약자로, 음성인식 시스템의 성능을 확인하는 표준 지표

$$WER = \frac{ins + sub + del}{words} \times 100(\%)$$

- 일반적으로는 monophone model이 triphone model보다는 성능이 떨어짐
 - 그렇지 않은 경우가 종종 나오는데, 이는 구현의 부적절함[†]으로 인해 발생

```
kaist@ubuntu:~/kaldi/egs/fsdd$ for x in exp/*/decode*; do [ -d $x ] && grep WER $x/wer_* | utils/best_wer.sh; done
%WER 17.20 [ 86 / 500, 1 ins, 43 del, 42 sub ] exp/mono/decode/wer_9
%WER 26.00 [ 130 / 500, 1 ins, 67 del, 62 sub ] exp/tri1/decode/wer_10
```

[†]구현의 부적절함

- (1) fsdd_lang_prep.sh에서 단어나 음소군 등 언어 정보를 정의할 때, 사용되는 음성 데이터 특성을 이미 안다고 가정했다. 이와 달리 일반적으로 원래 음성인식에서는 언어 정보를 음성 데이터와 무관하게 정의하고 사용한다. (따라서 디코딩시 out-of-vocabulary 문제가 나오기도 한다.)
- (2) 언어 정보를 fsdd_lang_prep.sh처럼 직접 정의한 이유를 본 시스템의 작동 범위가 digit recognition에 한정되기 때문이라고 하면, digit recognition 특성상 훈련에 필요한 언어 정보도 제한된다. 하지만 이 경우는 task 특성상 굳이 triphone과 같은 복잡한 모델링을 사용할 필요가 없다.
- (3) 따라서 여기서 triphone 모델이 (monophone 모델보다) 성능이 낮다면, 이는 task의 복잡도에 비해 모델 복잡도가 높아 overfitting (과적합)이 발생해 성능 열화가 일어났다고 볼 수 있다. 이 경우 과적합 점검 및 정도 조절은 triphone 모델 훈련 hyper-parameter인 senone 수 및 Gaussian 수를 줄여서 확인해볼 수 있다.

더 나은 결과를 만드는 법

- 성공적인 기계학습 시스템을 만드는 것은 결국 데이터
 - FSDD 전체 데이터셋은 3명의 화자로 이루어짐
 - 현재 구현에서는 이를 훈련-검증용으로 나눌 때 단순히 2명을 훈련에, 1명을 검증에 사용
 - 훈련에 사용되지 않은 화자의 발음 정보는 훈련된 모델에 들어가지 않게 된다!

[run.sh 중]

```
if [ $stage -le 0 ]; then
    echo ""
    echo "==== PREPARING ACOUSIC DATA ====="
    echo ""

    [ ! -d $fsdd_dir ] && git clone $fsdd_url $fsdd_dir
    local/fsdd_data_prep.sh $fsdd_dir/recordings --per-speaker
fi
```

- Stage 0의 위 부분에서 "--per-speaker" 부분을 지우면, 화자를 고르게 분배한다.
 - 자세한 구현사항은 직접 local/fsdd_data_prep.sh의 구현 부분을 확인할 것!
 - 위를 통해 전체 스크립트 실행시 성능이 대폭 향상되는 것을 확인 가능

더 나은 결과를 만드는 법

- 더 정확한 alignment를 가지고 모델을 훈련하면 더 정확한 모델이 나온다
 - 동일한 환경인 경우, 훈련된 tri1 모델을 가지고 앞서 monophone alignment를 수행한 것과 같이 alignment를 만들어 훈련하면 보다 정확한 모델이 나온다
- run.sh의 기존 부분을 사용하여, 오른쪽과 같이 두 번째 triphone 모델을 구현해볼 수 있다
 - 다만 이 경우에는, task의 단순성으로 인해 성능 향상을 보장하지 못함
- 일반적인 음성인식 task에서는 이를 응용하여, monophone GMM-HMM부터 시작하여 점차 정확성을 높여 가며 최종적으로는 신경망 기반의 음성인식 모델을 만든다

```
if [ $stage -le 9 ]; then
    echo ""
    echo "==== TRIPHONE MODEL ALIGNMENT ====="
    echo ""

    steps/align_si.sh --nj $nj \
        data/train data/lang exp/tri1 exp/tri1_al

fi

if [ $stage -le 10 ]; then
    echo ""
    echo "==== TRIPHONE MODEL (2nd pass) TRAINING ====="
    echo ""

    steps/train_deltas.sh 500 2000 \
        data/train data/lang exp/tri1_al exp/tri2

fi

if [ $stage -le 11 ]; then
    echo ""
    echo "==== TRIPHONE MODEL (2nd pass) DECODING ====="
    echo ""

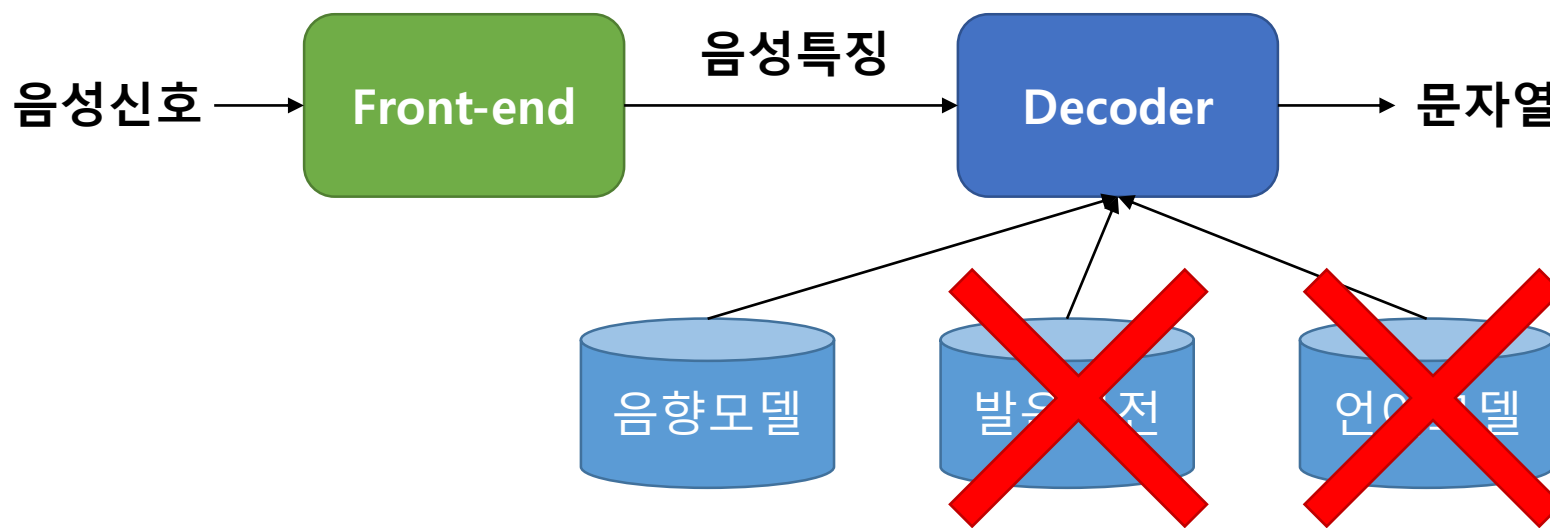
    utils/mkgraph.sh data/lang exp/tri2 exp/tri2/graph
    steps/decode.sh --nj $nj \
        exp/tri2/graph data/test exp/tri2/decode

fi
```

End-to-end ASR system

- 종단간 음성인식 시스템

- 현재 가장 고성능의 신경망 기반 음성인식 시스템은 결국 고전적 시스템 기반 (GMM-HMM 음향모델에서 Gaussian을 신경망 모델로 대체)
- 음소셋과 발음사전, 언어모델 등 여러 전문가적 정보 필요
- 음성신호에서 곧바로 문자열을 만들어내는 (End-to-end) 단 하나의 신경망 모델로, 음성인식을 구성하는 모든 모듈을 대체하려는 움직임



Resources

- LibriSpeech

- 대규모 (약 1,000시간) 공개 영어 음성 데이터로, 최근 많은 음성인식 연구에서 활용
- Kaldi의 egs/librispeech 디렉토리에 recipe 존재
 - <http://www.openslr.org/12> - 음성-문장 데이터
 - <http://www.openslr.org/11> - 언어모델, 발음사전, 단어목록

- Mini-LibriSpeech

- 위 LibriSpeech 데이터셋을 음성인식 시스템 입문자들이 활용하기 좋게 줄인 데이터
- Kaldi의 egs/mini_librispeech 디렉토리에 recipe 존재
 - <http://www.openslr.org/31> - 음성-문장 데이터

- Project Zeroth

- 한국어 음성인식을 위한 공개 한국어 음성 데이터 및 Kaldi 기본 recipe
 - <https://github.com/goodatlas/zeroth> - Kaldi recipe (data download 포함)
 - Kaldi 공식 배포판에도 Project Zeroth를 위한 기본 recipe 수록
- Kaldi의 첫 설치 및 사용이 어려울 경우 **참고할 만한 한국어 리소스**