UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
UNDERGRADUATE SCHOOL

Research and Development

BACHELOR THESIS

by

LE Gia Anh Quy

USTHBI7-133

Information and Communication Technology

Title:

Vietnamese Speech Recognition based Hidden Markov Model using CMU Sphinx

Supervisors: Dr. TRAN Giang Son

Lab name: ICT Lab

Hanoi. July 2019

Declaration

I declare that the thesis is entirely my own under the guidance of Dr. Tran Giang Son. I certify that the work and result are totally honest and unprecedented previously published in the same or any similar form. In addition, all assessments, comments and statistics from other authors and organizations are indicated and have been cited accordingly. If any fraud is found, I will take full responsibility for the content of my thesis.

Lời cam đoan

Tôi cam đoan rằng luận án này hoàn toàn thuộc về tôi dưới sự hướng dẫn của tiến sĩ Trần Giang Sơn. Tôi xác nhận rằng công việc và kết quả là hoàn toàn trung thực và chưa từng có trước đây được xuất bản dưới cùng hoặc bất kỳ hình thức tương tự nào. Ngoài ra, tất cả các đánh giá, nhận xét và thống kê từ các tác giả và tổ chức khác đều được chỉ định và đã được trích dẫn tương ứng. Nếu phát hiện bất kỳ gian lận nào, tôi sẽ chịu trách nhiệm hoàn toàn về nội dung của luận án của mình.

(LE Gia Anh Quy)
Hanoi, July 2019

Acknowledgement

Firstly, I would like to express my thanks to my research supervisor Dr. Tran Giang Son for giving me the opportunity to do research and providing invaluable guidance during this internship. Secondly, my sincere thanks also goes to Dr. Nghiem Thi Phuong for her encouragement, insightful comments and hard questions during the research. Thirdly, I thank my fellow labmates, friends in ICTLab: Kieu Tran, Linh Loc, An Luu, Duc Pham, Trung Bui for the stimulating discussions and for all the fun we have had in the last three years. I also very appreciate to have been a student at USTH where all professors and staffs are always willing to support their students. Besides, I would like to thank the CMU Sphinx research community for answering and providing many valuable knowledge I asked during the research. Last but not least, I express my deep and sincere gratitude to my parents who raise me up throughout my life by their patient and their love.

Lời cảm ơn

Trước tiên, tôi xin gửi lời cảm ơn sâu sắc tới Tiến sĩ Trần Giang Sơn đã hướng dẫn và tạo cơ hội vô giá để tôi nghiên cứu trong thời gian thực tập này. Thứ hai, lời cảm ơn chân thành của tôi cũng gửi đến Tiến sĩ Nghiêm Thị Phương vì sự khích lệ, những hiểu biết sâu sắc và các câu hỏi giá trị trong quá trình tôi nghiên cứu về đề tài của mình. Thứ ba, tôi cảm ơn các bạn đồng nghiệp, bạn bè của tôi ở ICTLab: Kiều Trần, Linh Lộc, An Lưu, Đức Phạm, Trung Bùi vì những cuộc thảo luận tích cực và vì tất cả những niềm vui chúng tôi có trong ba năm qua. Tôi cũng rất trân trọng khi là một sinh viên ở trường USTH, nơi tôi đã nhận được rất nhiều sự hỗ trợ từ các giảng viên và các anh/chị nhân viên. Bên cạnh đó, tôi muốn cảm ơn cộng đồng nghiên cứu CMU Sphinx đã trả lời và cung cấp nhiều kiến thức có giá trị mà tôi đã hỏi trong quá trình nghiên cứu. Cuối cùng nhưng không kém phần quan trọng, tôi bày tỏ lòng biết ơn sâu sắc và chân thành đến cha mẹ, những người nuôi dạy tôi suốt cuộc đời bằng sự kiên nhẫn và tình yêu của họ.

(LE Gia Anh Quy)
Hanoi, July 2019

Contents

List of Acronyms

AM Acoustic Model. 9

ASR Automatic Speech Recognition. 1

BW Baum-Welch. 9

CD Context-Dependent. 18

CI Context-Independent. 18

CMU Carnegie Mellon University. 15

CMUCLMTK CMU-Cambridge Statistical Language Modeling Toolkit. 15

DCT Discrete Cosine Transform. 8

DFT Discrete Fourier Transform. 6

DNN Deep Neural Network. 1

EM Expectation-Maximization. 9

FFT Fast Fourier Transform. 7

FPT The Corporation for Financing Promoting Technology. 2

GMM Gaussian Mixture Model. 9

HMM Hidden Markov Model. 2

HTK Hidden Markov Model Toolkit. v

ISIP Institute for Signal and Information Processing. v

LM Language Model. 11

LPC Linear Prediction Coefficients. 6

MFCC Mel-Frequency Cepstral Coefficients. 6

MLLR Maximum Likelihood Linear Regression. 24

OOV Out-Of-Vocabulary. 19

PNCC Power-Normalized Cepstral Coefficients. 6

SyER Syllable Error Rate. 4

VASR Vietnamese Automatic Speech Recognition. v

VNUHCM University of Science Ho Chi Minh City, Vietnam. 4

VSLP Vietnamese Speech and Language Processing. 5

WER Word Error Rate. 11

List of Figures

List of Tables

Abstract

Speech Recognition research has arisen significantly since the 1970s when Leonard Baum of Princeton University invented Hidden Markov Model (HMM) which provided a statistical-based approach to generate text from speech. Up to now, many research groups are developing open source toolkit for ASR, such as CMU Sphinx, Kaldi, HTK, Julius and ISIP, based on the combination of traditional HMM and n-gram language model.

For ASR application developers, it is better to start with CMU Sphinx because of its intelligible recipe. However, Sphinx does not consist any trained models in Vietnamese for them to develop an application. Therefore, I am trying to build and deploy a Vietnamese Automatic Speech Recognition (VASR) model, tune its performance using Sphinx's packages. Moreover, I hope my work helps new researchers who have difficulty in understanding the concept of Speech Recognition, are able to build their own models without needing too much specialist knowledge.

The purpose of the thesis is to propose a practical method of recognizing Vietnamese speech to text based HMM using CMU Sphinx system. The 3 different datasets VIVOS, FPT and VoiceViet are used to train and test separately in order to analyze ASR system's performance on different characteristics: recording environment, variant speakers, corpus quality. Besides, the trained model can be embedded in Android mobile platform.

Keywords: *Vietnamese Automatic Speech Recognition, Hidden Markov Model, CMU Sphinx*

Chapter 1

Introduction

1.1 Definition

Automatic Speech Recognition (ASR) is the process of converting the speech signal to a string of words automatically by means of algorithms. It is the ability of a machine to identify words or phrases in spoken language and convert them to a machine readable format *(Mehmood et al. 2014)*.

An ASR system is categorized depending on its application mainly on 3 aspects : speech mode, speaker model and size of vocabulary. Firstly, speech mode determines the ability to recognize the speech signal of the system, e.g: isolated words, connected words, continuous speech and spontaneous speech. Secondly, speaker model, which tells whether the system depends on specific speakers, focuses on the properties of speaker voice. Thirdly, the accuracy, complexity and processing requirements of the system depend on the size of vocabulary including: small (tens of words), medium (hundreds of words), large (thousands of words) and very large (tens of thousands of words). In this project, I try to build a continuous independent large vocabulary ASR system for Vietnamese, in which the system can recognize natural spoken language of a wide-range of people across 3 dialects of Vietnam (North, South and Central) with a number of vocabulary (about 5 thousands).

1.2 Scientific Background

In early days, the first recognizer, Audrey, created by Bell Laboratories in 1952 was able to recognize spoken numerical digits by looking for acoustic fingerprints, but it was not exploited. Then, a key turning point coming out in the early 1970s when Jim Baker first applied HMM which had been described by Leonard E. Baum in the '60s, to speech recognition. It was the first time that ASR community had been able to see a lighting day to measure and compare success. And recently, DNN has changed the landscape in ASR research because of the computational power of the modern era and big data. It is faster, more natural and more friendly to human than other means of communication. As a result, research and development of a state-of-the-art speech recognition system attracts more attention of both researchers and developers.

1.3 Motivation

Over the last few years, a number of commercial continuous ASR systems show pretty well-performance, such as Google Speech-To-Text, Microsoft Cognitive Services or FPT Speech Recognition in Vietnam. They are light, fast and quick to load. However, there are many situation where you cannot use API and need to develop an ASR from scratch. Nevertheless, it is impossible for new researchers to implement all the system. This leads to a release of a good number of open-source ASR systems which provide free sufficient toolkits to build a custom ASR engine. Some of them are recently implemented DNN to achieve a higher level of accuracy but in this project, I only take care of an ASR system based on traditional HMM since it is still a knowledge base to learn every aspects of ASR in order to build a state-of-the-art system.

1.4 Objectives

My thesis is about building a continuous speech, independent-speaker and large vocabulary Vietnamese ASR system based on the combination between Hidden Markov Model (HMM) and N-gram Language Model, using speech recognition toolkit CMU Sphinx. In this thesis, I use 3 open pre-built speech corpus Vivos[2], FPT Open Speech Data[3] and VoiceViet[4] for training 3 different models and testing. Purposely, the trained model can be embedded in mobile applications with the support of CMU Sphinx. Moreover, I can gain experiment and knowledge in multimedia information processing, especially in speech recognition.

1.5 Thesis Structures

In this section, I will summarize throughly the content of each chapter:

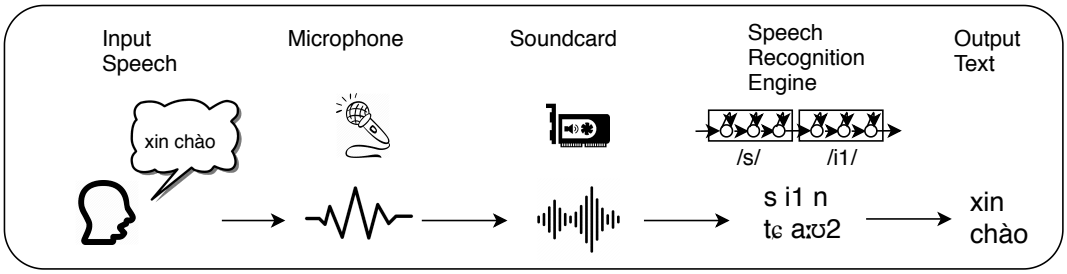
- Chapter 1: Introduce briefly the definition, history of ASR and the aim of my thesis.
- Chapter 2: Fundamental theory of ASR related to the project, brief overview of several ASR toolkits, and some recently researches, achievements, challenges in Vietnam.
- Chapter 3: Recipe to adapt Vietnamese ASR in CMU Sphinx framework.
- Chapter 4: Experimental setups in order to get the best accuracy result.
- Chapter 5: Summary of results, discussion and future work.

[2]https://ailab.hcmus.edu.vn/vivos
[3]https://fpt.ai/fpt-open-speech-data/
[4]https://voiceviet.itrithuc.vn/

Chapter 2

Automatic Speech Recognition

2.1 The general ASR concept



Step	Process Name	Description
1	Speech Input	Microphone gets human's voice in the form of analog signal
2	Digitization	Sound card converts analog signal to digital signal
3	Recognition	Speech recognition engine converts digital signal to phonemes, then words
4	Application	System processes recognized words for application: voice search, voice control,...

Figure 2.1 – Basic steps of ASR Concept.

Figure 2.1 illustrates roughly the basic steps of an ASR system *(Liao. 2003)*. The process can be divided into 4 steps. It involves capturing the voice of user, converting analog signals into digital signals, breaking them into smallest unit of utterance (phonemes) to predict the most probable sequence of relative words, finally the recognized word will be used as input for other application purpose.

2.2 Open-Source ASR Toolkits

Currently, according to *Thompson (2017)*, CMU Sphinx, Kaldi, HTK, Julius and ISIP are the most 5 popular open-source toolkits. Table 2.1 shows the rating of each toolkit on 3 characteristics: development activity, tutorials/examples and community exclude their recognition accuracy. It also lists some existing trained models in those systems. It shows that CMU Sphinx has the highest rating scores along with trained sample models for over 10 languages, not including Vietnamese.

In term of accuracy, Gaida et al. (2014) ran a large-scale of evaluation on HTK, Julius, Sphinx and Kaldi. They shown that Kaldi outperformed all the other recognition toolkits but has the highest computational cost. Sphinx also provided a good results shortly but requiring the development of advanced parts by the user and extensive tuning. HTK had the similar accuracy result to Sphinx but it is nearly impossible to use without being an expert. Finally, Julius performed the poorest result for both usability and accuracy. Overall, each toolkit has their own pros and cons but for developing application purpose, CMU Sphinx is the most appropriate because of its flexibility, readable documentations and well-balance between accuracy and computational cost.

Table 2.1 – Comparison of open source speech recognition toolkits[5]

Toolkit	Programming languages	Development activity	Tutorials and examples	Community	Trained models
CMU Sphinx	Java, C, Python, others	+++	+++	+++	English plus 10 other languages
Kaldi	C++, Python	+++	++	+++	Subset of English
HTK	C, Python	++	+++	++	None
Julius	C, Python	++	++	+	Japanese
ISIP	C++	++	++	+	Digits only

2.3 Related Work

In Vietnam, the ASR research community has developed for more than 20 years. There are many researches reaching their remarkable achievements with the support of open-source ASR toolkits. Now, I want to give my review on some of their recently researches.

A Non-expert Kaldi recipe for Vietnamese Speech Recognition System *(Luong et al. 2016)*
Luong et al. at AILab of VNUHCM built a VASR system using Kaldi toolkit on their own speech corpus VIVOS. For acoustic model, they trained 4 different models: a triphone GMM-based baseline, one discriminative trained, one speaker adaptation for GMM and a hybrid HMM-DNN with speaker adapted feature, along with 3 different recipes: grapheme-based pronunciation dictionary with standard MFCC feature, augmenting of pitch into the acoustic feature and incorporation of tones to the phonetic decision tree. The language model they used, was collected from online news and forum, size up to 500MB. In result, the SyER varies from 9% to 19%. From my point of view, they provided very good recipes in setting up Kaldi to be suitable for tonal speech Vietnamese to gain a very high accuracy for medium-scale ASR in comparing between traditional HMM and DNN based model. However, it has a heavy knowledge for beginners to reproduce their experiment despite of the non-expert recipe.

Automatic Speech Recognition for Vietnamese using HTK system *(Nguyen et al. 2010)*
In this paper, *Nguyen et al.* research on VASR using HTK system. They built text and speech corpus on their own: about 6 hours of Northern Vietnamese speech to train acoustic model and 500 MB of text to train language model. They used several methods of representing Vietnamese phoneme in training acoustic model in which, the combination of two diphones in each syllable had the highest accuracy for independent-speaker, 71.37% . MLLR method was used for speaker adaptation to increase performance for identified speaker, up to 75.96%. To train language model, CMUCLMTK was used and the bigram LM was applied to HTK. Overall, they made great effort in researching Vietnamese phonology and applied to build phonetic dictionary. They also have

[5]https://www.svds.com/open-source-toolkits-speech-recognition/

a very detail and condense content from preprocessing data to training models. It is extremely helpful for new researchers who want to use HTK toolkit for ASR.

Development Of High-performance And Large-scale Vietnamese Automatic Speech Recognition Systems *(Do et al. 2018)*
Do et al. develop an ASR system with the implementation of DNN instead of traditional HMM in training acoustic model for large-scale speech corpus ($\approx$ 1000 hours) with large variation of speakers (North, South, Central) and speaking conditions (quiet, noisy). For language model, they collect text data from various online websites and movie subtitles, and use the 4-gram model (99 MB) and its pruned version (18 MB) in order to reduce the memory usage in online decoding. The lowest WER they can achieve just about 6% on VSLP 2018 challenges, 9%, and 17.39% on their own clean/noisy test sets respectively. Besides, they introduce their approach to collect a large amount of data in a short period of time but still maintain the phonetic balance property. Generally, Do et al. has made the new state-of-the-art system for VASR by implementing DNN for a practical large-scale data problem.

Overall, Luong et al., Nguyen et al. are describing reasonable methods of training a VASR system based HMM with support of open source frameworks, but none of them using CMU Sphinx. Meanwhile, Do et al. has achieved the best performance for realistic problem in VASR, but based on DNN. DNN is the newest approach in speech recognition, but it is only implemented in Kaldi. However, it is very difficult to use Kaldi framework without prior experience in speech recognition. It is better to use CMU Sphinx to understand every basic aspects of ASR although it only uses the traditional HMM approach.

2.4 Speech Recognition Engine

A speech recognition engine can include generally two components: *front-end* and *decoder* (see Figure 2.2). The front-end block involves in parameterizing speech signal for the decoder to recognize speech to text. The decoder uses acoustic model, language model and lexicon to search for the best sequence of words.

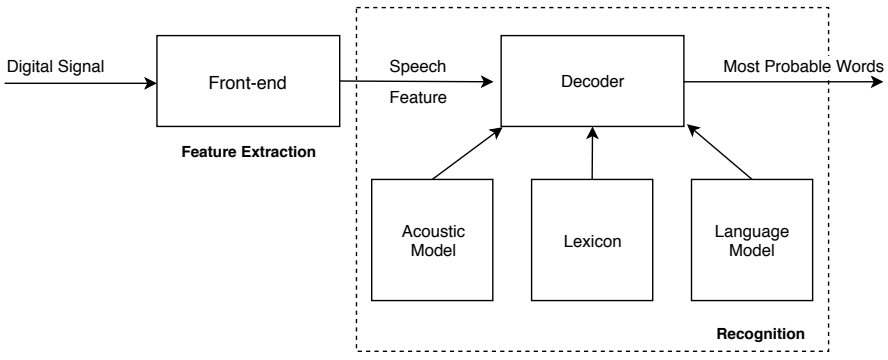


Figure 2.2 – A general speech recognition engine.

2.4.1 Feature Extraction

First and foremost, feature extraction is the heart of an ASR engine. It converts the waveform into a set of feature vectors. There are several features which have been tried for speech recognition,

such as Mel-Frequency Cepstral Coefficients (MFCC), Linear Prediction Coefficients (LPC), Power-Normalized Cepstral Coefficients (PNCC). The main goal is to make the vectors to have the highest discrimination between phonemes of speech.

2.4.1.1 Mel-Frequency Cepstral Coefficient

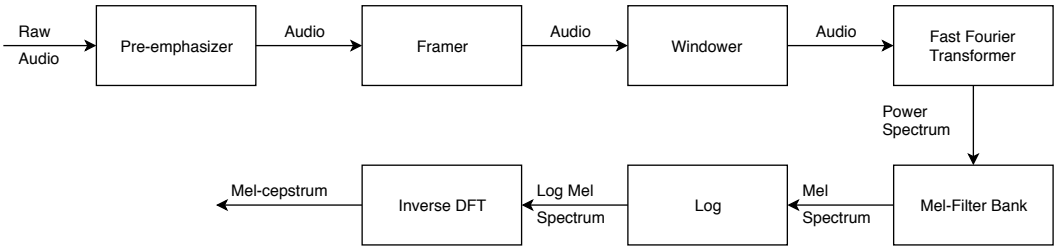


Figure 2.3 – MFCC steps in extracting feature.

Mel-Frequency Cepstral Coefficients is the most popular feature extraction technique in speech recognition because it can mimic cochlea function of our ears. The cochlea, which produces nerve impulses to human's brain in response to sound vibration, perceives the frequencies linearly up to 1 KHz and logarithmically above it *(Stevens et al. 1940)*. Therefore, the idea of MFCC is to convert audio in time domain into frequency domain to see all speech's information, then mimics cochlea's filters by using Mel filters. Figure 2.3 describes all steps to extract MFCCs.

Pre-emphasis
Pre-emphasis boosts the amplitudes of the high frequencies. Therefore, it exposes more phonetic information which is diluted in natural speech, then improves phone-based recognition performance. In simple form, it can be implemented as:

$$y_t = x_t + \alpha x_{t-1} \tag{2.1}$$

where $\alpha < 0$. The reason is that low frequency essentially means slow variation in time and so the numerical values of a low frequency signal tend to change slowly or smoothly from sample to sample. By subtracting, it removes the part of the samples that did not change in relation to its adjacent samples and the remains is the part of the signal that changes rapidly (high frequency signals).

Framing
Generally, the speech signal is approximately constant in short period of time (usually 20-30 ms) *(Huang et al. 2001, p.275)*, but changes over a longer time. Therefore, when extracting features, it is necessary to segment the waveform into small frames with overlapping, then process each of them separately. By overlapping, the framing process can minimize the lost signal information.

Windowing
The window function is used to smooth the signal for the computation of the Discrete Fourier Transform (DFT). The DFT computation makes an assumption that the input signal repeats continuously. Therefore, if there is a discontinuity between the first point and last point of adjacent frames, we can not compute DFT. So, every sample in a frame will be multiplied by window function to smoothly attenuate both ends of the frame towards zero.
The Hamming window is usually used in speech signal spectral analysis, because its spectrum

falls off rather quickly so the resulting frequency resolution is better, which is calculated as:

$$\begin{aligned} &Y(n) = X(n) * W(n), \\ &W[n] = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{N}, & 0 \le n < N \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2.2}$$

where W(n) is Hamming window function, N = number of samples in each frame, Y(n) = output signal, X(n) = input signal.

Fast Fourier Transform

Fast Fourier Transform (FFT) is simply a fast algorithm that computes the DFT of a signal. DFT is a discrete version of a Fourier Transform which converts each window of N samples from time domain to its frequency domain. The k^{th} sample of a Fourier Transform is computed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{j2\pi kn}{N}} \tag{2.3}$$

where x[n] is the n^{th} sample in the time domain, X[k] is the value of the k^{th} sample in its Fourier spectrum, N is the total no. of samples in the window. DFT coefficients are generally complex where $e^{j\theta}$ has a real part $\cos\theta$ and an imaginary part $\sin\theta$:

$$e^{j\theta} = \cos\theta + j\sin\theta \tag{2.4}$$

As a result, the power spectrum is computed by squaring magnitude spectrum as following equations:

$$\begin{aligned} X[k] &= X_{real}[k] + jX_{imag}[k] \\ X_{magnitude}[k] &= sqrt(X_{real}[k]^2 + X_{imag}[k]^2) \\ X_{power}[k] &= X_{magnitude}[k]^2 \end{aligned} \tag{2.5}$$

Mel-filter bank

This process modifies the spectrum to model the non-linear frequency resolution of the human ear because human discriminates sounds better at low frequencies. At first, linear frequency spectrums are warped to Mel-scale frequency spectrums for better resolution at low frequencies and less at higher frequencies, the conversion between Hertz(f) and Mel(m) using following equations:

$$m = 2595 log_{10}(1 + \frac{f}{100}) \tag{2.6}$$

$$f = 700(10^{\frac{m}{2595}} - 1) \tag{2.7}$$

Moreover, human ear actually responds to a band of frequencies with a peak response at a particular frequency. Then, a bank of triangular filters are applied to mimic this, typically 24-40 filters (*Huang et al. 2001, p.314-316*). Each triangular filter has a response of 1 at the center frequency and decrease linearly toward 0 till it reaches the center frequencies of the two adjacent

filters where the response is 0. A triangular filter is calculated by the following equation :

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)}, & f(m-1) \le k < f(m) \\ 1, & k = f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)}, & f(m) < k \le f(m+1) \\ 0, & k > f(m+1) \end{cases} \tag{2.8}$$

which satisfies $\sum_{m=0}^{M-1} H_m[k] = 1$ where M is number of filters, f(m) is a Mel-scaled frequency at filter m.

To calculate filter-bank energies, we multiply each filter-bank with the power spectrum, then add up coefficients:

$$S[m] = \sum_{k=0}^{N-1} X_{power}[k] H_m[k],\ 0 \le m < M \tag{2.9}$$

Log energy computation

The logarithm is used to compress dynamic range of filter-bank energies, $\log S[m]$.

Inverse DFT

The final process is to convert the Log Mel-spectrum into time domain by inverse DFT. The most common technique to inverse DFT is Discrete Cosine Transform (DCT), specifically DCT-II for speech recognition (*Huang et al. 2001, p.228*), because it decorrelates the filter-bank coefficients, which can be modelled efficiently as a Gaussian distribution:

$$y[k] = \sum_{m=0}^{M-1} \log S[m] \cos\left(k(m+\frac{1}{2})\frac{\pi}{M}\right),\ \text{for } 0 \le k < M \tag{2.10}$$

2.4.2 Recognition

Speech recognition problem is essentially a probabilistic problem: finding the most probable word sequence, $\hat{W}$, given the acoustic observations, A.

$$\hat{W} = \underset{W}{argmax} P(W|A) \tag{2.11}$$

(Maximum a posteriori)

$$= \underset{W}{argmax} \frac{P(A|W)P(W)}{P(A)} \tag{2.12}$$

(Bayes' rule)

Since $P(A)$ is a constant for any single observation sequence, Equation (2.12) can be written as:

$$\hat{W} = \underset{W}{argmax} P(A|W)P(W) \tag{2.13}$$

The probability of $P(A|W)$ and $P(W)$ can be estimated from the predefined acoustic model and language model.

2.4.2.1 Acoustic Model

Acoustic Model (AM) contains statistical representations of each distinct phonemes that makes up a word. These statistical representations are called Hidden Markov Models (HMMs). Each phoneme has its own HMM. The speech decoder will search the most similar HMM to each distinct phoneme, then look for a word which is combined by those matching HMMs.

Gaussian Mixture Model - Hidden Markov Model

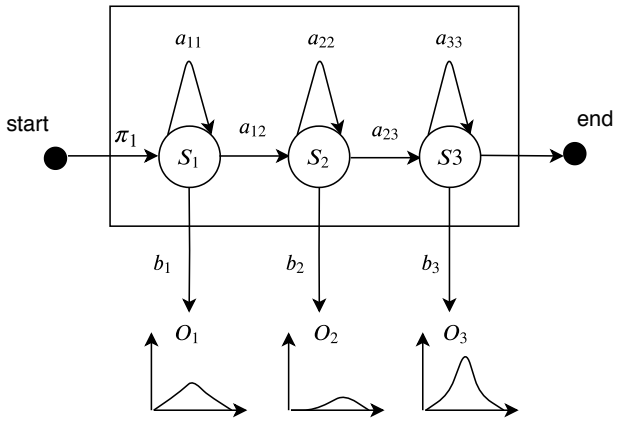


Figure 2.4 – 3-state HMM.

A HMM is a probabilistic model of the joint probability of a collection of random variable $O = \{O_1, ..., O_T\}$ and $S = \{S_1, ..., S_T\}$ in T-segments. In other words, HMM served as statistical model to estimate the probability of a set of observations O based on the sequence of hidden transition states S. Applying in ASR, the goal of AM is to find the most probable sequence of phonetic representations (hidden states) given a sequence of speech signal segments (observations). To train a HMM, it is required 3 principle learning parameters: *initial state distribution* $\pi = \pi_i = P(s_1 = S_i)$, *transition probability matrix* $A = a_{ij} = P(s_{t+1} = S_j | s_t = S_i)$ and *observation distribution* $B = b_i(O_t = P(O_t | s_t = S_i)$. Hence, the likelihood $P(A|W)$ can be written as:

$$\begin{aligned} P(A|W) &= \sum_{allS} P(O|S,\lambda)P(S|\lambda) \\ &= \sum_{s_1,s_2,...,s_T} \pi_{s_1} b_{s_1}(O_1) \prod_{n=2}^{T} a_{s_{n-1}s_n} b_{s_n}(O_n) \end{aligned} \tag{2.14}$$

These parameters $\lambda = (\pi, A, B)$, can be estimated using the Baum-Welch (BW) algorithm, which is special case of the Expectation-Maximization (EM) algorithm (*A. Bilmes. 2000*). The Gaussian Mixture Model (GMM) is used to approximate the *observation distribution B* in EM. In other word, each state is modeled by a GMM to determine the likelihood of the observation in only that state. Overall, the GMM-HMM model is designed to exploit the temporal in context-dependent units.

Triphone

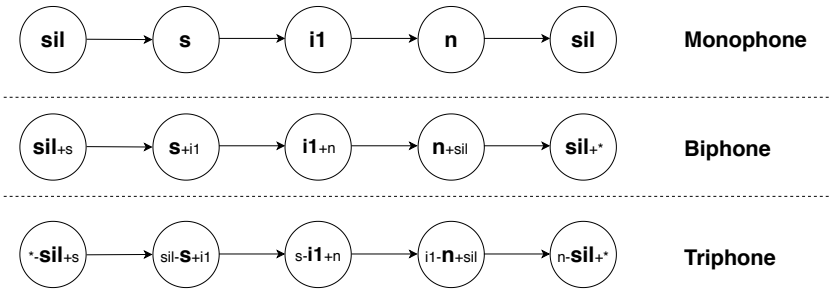


Figure 2.5 – Monophone, Biphone, Triphone acoustic models of word "xin".

Typically, most common GMM-HMM model uses tied-state Triphones, also known as "Triphones", to improve the recognition accuracy *(Huang et al. 2001, p.428)*. HMM is usually trained on different units of phone instead of word because there are much fewer phones to compute than words. Neither a fraction nor a sequence of phones is a triphone. A triphone is just a whole phone, in which the first part depends on the preceding phone, the second part is stable and the next part depends on the subsequent phone. In other word, a triphone depends on left and right context. Hence, depending on the amount of context, we use different terms to describe a model of a phone (see Figure 2.5), e.g: monophone - context independent, biphone - depends on either left or right context, quinphone - depends on 2 phones to the left or 2 phones to the right, etc.

State Tying - Senone

Moreover, among all triphone models, there are many states that are similar to each other. Therefore, they can share the same set of parameters. This sharing process is called 'tying', and those groups of tied states are called "Senones" *(Huang et al. 2001, p.430)*. In such a way, when we re-estimate these senones, the untied data is pooled so that a better estimation is obtained.

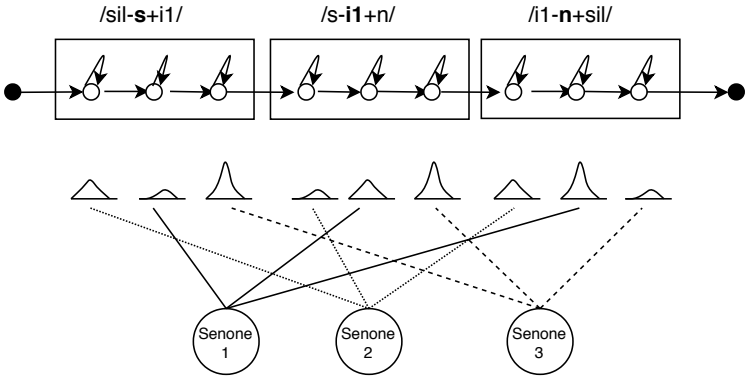


Figure 2.6 – Tied-state triphone HMM.

Decision tree

The most common technique in tying state is decision tree. Decision trees categorize triphone states into a linguistic questions. Optimal questions at any level of the tree are determined from data. All triphone states which end up at the same leaf of tree will be tied, which means a state has one decision tree. The distribution at any state is corresponding to its composed decision tree. The total number of tied states or senones or decision trees' leaves is fixed, depending on the

amount of training data . The number of Gaussian per tied state also depends on the amount of training data. Both increasing the number of Gaussian components and senones needs more sufficient data.

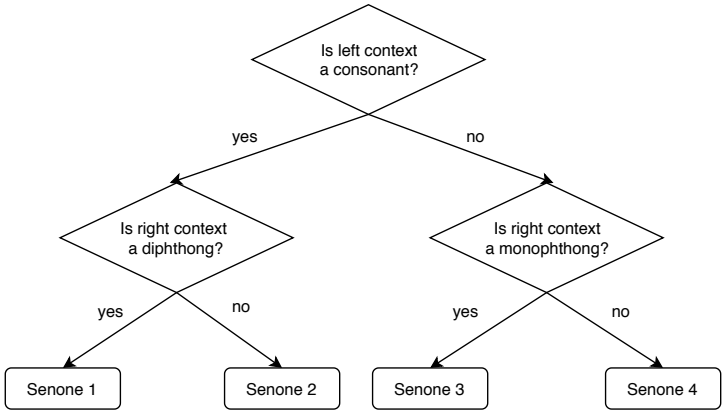


Figure 2.7 – Decision tree of a state.

Word Error Rate

Word Error Rate (WER) is widely used as one of the most important measurement to evaluate speech recognition performance. To compute the minimum error rate, we need to align two word sequences and compute the number of substitutions (Subs), deletions (Dels) and insertions (Ins). This problem can be solved by using Dynamic Programing algorithm. The WER is then defined as:

$$\text{WER} = 100\% * \frac{\text{Subs+Dels+Ins}}{\text{No. of word in the correct sentence}} \tag{2.15}$$

2.4.2.2 Language Model

Language Model (LM) contains a large list of word with their probability of occurrence based on a set of documents. In ASR, LM helps discriminate ambiguous phonemes and limits the number of possible words that need to be considered at any point in search space. As a result, the decoding process is faster and higher accuracy. For example, it is very difficult to distinguish between these two utterance using acoustic properties, "xôi lạc bánh khúc" and "tôi lạc bánh khúc". Obviously, with prior knowledge in Vietnamese, the first utterance is more likely to be hear than the other one in real life.

N-gram LM

Back to Equation (2.13), the LM is used to calculate the prior probability $P(W)$ of the word sequence W, can be written as:

$$\begin{aligned} P(W) &= P(w_1, w_2, ..., w_n) \\ &= P(w_1)P(w_2|w_1)P(w_n|w_1, w_2, ..., w_{n-1}) \\ &= P(w_1)\prod_{i=2}^{n} P(w_i|w_1, w_2, ..., w_{i-1}) \end{aligned} \tag{2.16}$$

$w_1, w_2, ..., w_n$ are all the words making a observing sentence. In practical, it is impossible to cover all scenarios to calculate the probability $P(w_i|w_1, ..., w_{i-1})$ *(Huang et al. 2001, p.553)*. Instead, it is assumed that w_i only depends on some n words before, called n-gram language model. For the

sake of simplicity, n is a small number, such as unigram $P(w_i)$, bigram $P(w_i|w_{i-1})$ and trigram $P(w_i|w_{i-1}, w_{i-2})$.

N-gram Probability Estimation

To build a N-gram LM, it is required a large scale of sentences in which, we need to label the starting with <s> and ending with </s> of a sentence. It is because:

> *"We need the end-symbol to make the bigram grammar a true probability distribution. Without an end-symbol, the sentence probabilities for all sentences of a given length would sum to one. This model would define an infinite set of probability distributions, with one distribution per sentence length." Jurafsky et al. 2018*

$P(w_i)$ can be estimated simply by counting words in training text. The ending symbol "</s>" is treated exactly as a word, so the $P(</s>)$ is counted like other words. For example, the 3-gram probability of the ending word </s> given context "<s>$w_1 w_2 w_3 w_4$</s>" is calculated as:

$$P(</s>|w_3, w_4) = \frac{Count(w_3, w_4, </s>)}{Count(w_3, w_4)} \tag{2.17}$$

However, direct estimation by counting is not a possible approach in all cases. If we have 500 words in our vocabulary, it could exist $500^2 = 250000$ bigrams and even much larger for trigram LM, in which most N-grams (words) will have 0 count. However, this does not mean that unseen bigrams or trigrams will never occur during recognition since a person could say something silly which doesn't exist in the training text. Therefore, it should be another approach to deal with the problem, called "Smoothing".

Smoothing

Smoothing is the process of modifying probability estimation to be more general. In other words, it takes a bit from the ones who have, and distribute to the ones who don't. There are various smoothing techniques such as Good-Turing Smoothing, Jelinek-Mercer Smoothing, Kneser-Ney Smoothing or Absolute Discount, etc. All of them follow the same basic principle: *Modify the counts of seen N-grams (e.g: Count-ϵ, ϵ=1, 0.5 or 0.1)*. The probability of seen words are calculated from the modified counts, while the left-over probability is reassigned to unseen words. For unigram, the left-over probability is uniformly distributed over all unseen words after discounting. However, for N-gram (N>1), the process is based on lower-order N-gram probability, e.g: trigram based on bi-gram, bigram based on unigram, which is referred to as "Back-off" (also known as Katz Smoothing).

Back-off N-gram LM

Back-off is a smoothing technique, used to estimate recursively probabilities for unseen high-order N-grams based on (N-1)-grams. Let $W_0 = \{w_1, ..., w_K\}$ are all words in the vocabulary (include </s>). Supposing that W_{N-1} is the context in the training data we are trying to estimate N-gram probabilities $P(w_i|W_{N-1})$. Let $w_1, ..., w_L$ are all words being seen following context W_{N-1}. We compute the N-gram probability for these words after discounting. Then, we have a left-over probability mass:

$$P_{left over}(W_{N-1}) = 1 - \sum_{i=1}^{L} P(w_i|W_{N-1}) \tag{2.18}$$

Next, we need to assign $P_{leftover}(W_{N-1})$ to all unseen words $w_{L+1}, w_{L+2}, ...$ in context $W(N-1)$. According to back-off assumption for unseen words, the probabilities of **unseen N-grams** must follow the same pattern as **seen (N-1)-grams**, but need to be scaled by a constant β, such that:

$$P_{leftover}(W_{N-1}) = \beta(W_{N-1}) \sum_{i \in unseen} P(w_i|W_{N-2}) \tag{2.19}$$

In which, the scaling constant $\beta(W_{N-1})$, also called as the "Back-off Weight", is specific to the context of the (N-1)-gram. Then, the Equation (2.19) is derived to find the back-off weight β:

$$\beta(W_{N-1}) = \frac{P_{leftover}(W_{N-1})}{\sum_{i \in unseen} P(w_i|W_{N-2})} \tag{2.20}$$

Now, to find back-off weight β for unseen N-grams in context W_{N-1}, we should know the probabilities of corresponding (N-1)-grams. Even if, (N-1)-grams are unseen, then we must know the probabilities of corresponding (N-2)-grams. This process is "back-off" recursively until reaching the 1-grams (unigrams) where we are always able to calculate its unseen probabilities after discounting:

$$\sum_{i \in unseen} P(w_i) = 1 - \sum_{i \in seen} P(w_i) \tag{2.21}$$

All computed back-off weight β will be stored for all words existing in training data in LM, then will be retrieved to calculate for unseen N-grams if required. Figure 2.8 describes every steps in retrieving a trigram probability $P(w_3|w_1, w_2)$ from LM.

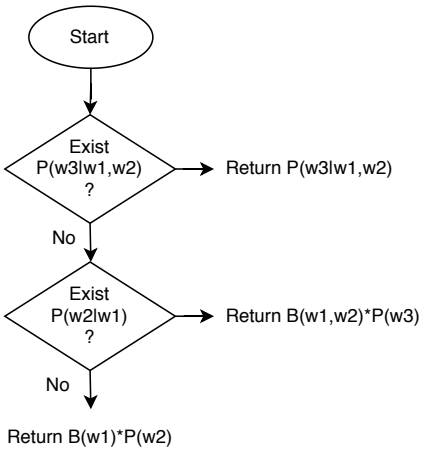


Figure 2.8 – Back-off to find probability for a trigram in language model.

Perplexity

Perplexity is a metric to evaluate the performance of language model by applying the measure of cross-entropy without participating in ASR system. Assuming we have a test corpus $C = s_1, s_2, ..., s_m$

with m sentences and N words for total, the perplexity on C of a trigram LM will be computed as:

$$p(s) = \prod_{i=1}^{length(s)} p(w_i|w_{i-1}, w_{i-2}) \tag{2.22}$$

$$H(C) = -\frac{1}{N}\sum_{i=1}^{m} \log_2 p(s_i) \tag{2.23}$$

$$PP(C) = 2^{H(C)} \tag{2.24}$$

The perplexity on test corpus evaluates the generalization capability of the language model. The lower the perplexity the lower complexity of language model, which will be better for recognition. However, a low perplexity does not guarantee to have good recognition performance if acoustic recognition is highly confusable *(Huang et al. 2001, p.556)*.

2.4.2.3 Lexicon

Lexicon is also known as "Phonetic Dictionary". It contains all vocabularies with corresponding pronunciation that an ASR system could recognize. Moreover, lexicon links the gap between acoustic model and language model for the decoder to search for the best hypothesis. It means the decoder will know which acoustic models are used for a certain word.
The size of a dictionary is an important factor for an ASR system because the bigger size of dictionary, the bigger complexity for both AM and LM, which will affect the accuracy and speed. Vocabulary's size usually depends on the specific domain of an ASR system, e.g: number of vocabulary for voice control system is much fewer than voice search system.
Every language has its own pronunciation rule, so it is not an easy task to create such a good coverage dictionary without expert knowledge in linguistics. It is prefer to use existing dictionaries then extend with their written rule-based.

2.4.2.4 Decoder

Decoder incorporates information from acoustic model, language model and lexicon to decode the given spoken input. So far, we have known that acoustic model, language model are just containing large lists of statistic representations and lexicon is used to link between them. Then, decoding process is referred to the *search algorithm* for the most possible word sequence through all available sequences. As stated in Equation 2.13, speech recognition problem is related to find the word sequence $\hat{W}$ which maximizes its prior probability $P(W)$ (modeled by LM) and the likelihood of spoken input A, $P(A|W)$ (modeled by AM) among all available word sequences $W = \{W_1, W_2, ...\}$. Therefore, speech recognition can be represented as Word HMMs, in which individual phonemes are trained by HMMs, then are concatenated to make larger HMMs for words *(Huang et al. 2001, p.439)*. To find $\hat{W}$ in a real-time system for optimal accuracy and speed, especially when the size of vocabulary increases, there are several complex search strategies designed to deal with the problem such as Viterbi Beam Search, Lexicon Tree Search, etc.

Chapter 3

Vietnamese Adaptation in CMU Sphinx

3.1 Speech Recognition Framework CMU Sphinx

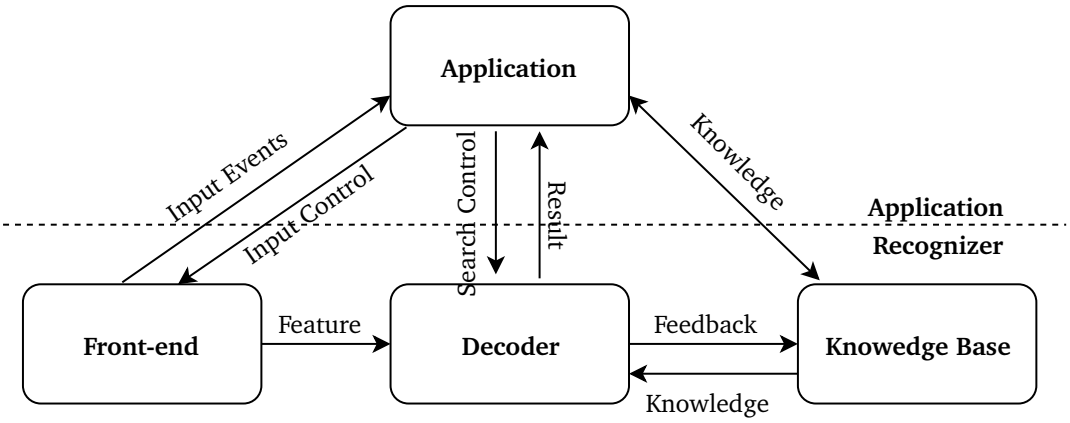


Figure 3.1 – CMU Sphinx Design Architecture (Liao, 2003).

CMU Sphinx[6], or Sphinx in short, is a project developed at Carnegie Mellon University (CMU) used for both research and commercial purpose of ASR. *Liao (2003)* presented a high level architecture of Sphinx (see Figure 3.1). The architecture consists of the front end block, the decoder block and the knowledge base block. The speech input firstly goes through the front-end to be converted to speech features, then the decoder uses linguists learned from knowledge base which includes acoustic model, language model and phonetic dictionary (lexicon) to search for the best matching word sequence. My work focuses on the adaptation in the knowledge base in order to provide speech features and linguistics of Vietnamese for the system to learn. The work includes building phonetic dictionary, building language model and training acoustic model with existing packages provided by CMU Sphinx and some external tools:

CMU Sphinx's packages: *Sphinxtrain* - Acoustic model trainer, *CMU-Cambridge Statistical Language Modeling Toolkit (CMUCLMTK)* - Language model trainer in ARPA format, *Pocketsphinx* - Sphinx's fastest decoder with the most accuracy at realtime speed (*Huggins-Daines et al.2006*).

[6]https://cmusphinx.github.io/

Other tools: *Phonemizer* - A lightweight text-to-phoneme converter based on Espeak (supporting IPA), *Audacity* - Record and modify audio, *Python* - Convert audio sampling rate, bit depth and modify text.

3.2 Vietnamese Pre-built Data Collection

3.2.1 Speech Corpora

Speech corpus is an important requirement for developing any ASR system. Speech corpus consists of audio utterances and their text transcription. A good speech corpus need to be large, high quality and cover large range of speaker styles on specific word. Making a speech corpus requires a lot of time, therefore, to save time, I use some free pre-built speech corpora of other groups. I train 3 different models on 3 different speech corpora which are VIVOS, FPT Open Speech Data and VoiceViet. The characteristics of 3 datasets are summarized in the following table:

Table 3.1 – Properties of Vietnamese pre-built speech corpora VIVOS, FPT, VoiceViet

Dataset	Dialect	Environment	Training	Testing
VIVOS	North, South, Mid	Quiet	13h	0.7h
FPT	North, South, Mid	Noisy	17h	0.7h
VoiceViet	North	Quiet	15h	3.5h

According to VIVOS and VoiceViet, two dataset were recorded in a quiet environment with high quality microphone, the training and testing set were carefully built by variant of speakers (only North speakers for VoiceViet, South + North + Central speakers for VIVOS). Meanwhile, the FPT dataset was recorded in a noisy environment and does not contain any detail description of the recording process.

3.2.2 Text Dataset

Normally, the text corpus used to train a language model is very large (up to million sentences) and be collected from various online sources such as newspaper, forums, stories, etc. However in my work, for the simplicity, I initially use transcription from speech corpus to train language model. Although the size of text transcription from the speech corpus is small, it's still cover enough information in the specific domain of each dataset. Latter, more data will be crawled to train a larger language model to see whether performance is improved (see Experiment 4.3). The following table shows the size of each text dataset:

Table 3.2 – Size of text data VIVOS, FPT, VoiceViet

Dataset	Size	Sentence	
		Training	Testing
VIVOS	1.2MB	11660	760
FPT	1.3MB	17393	525
VoiceViet	1.1MB	17945	4487

3.3 Vietnamese Adaptation Recipe

3.3.1 MFCC Feature

Before training and testing, a set of feature vectors are computed from the audio data, one each for every recording in the speech corpus. The MFCC technique is applied to extract a 39-dimensional feature vector which is composed of first 13 MFCCs and their corresponding 13 delta (velocity features) and 13 delta-delta (acceleration features). The MFCC parameters are described in the following table:

Table 3.3 – MFCC feature parameters

Parameter	**Value**
Sampling rate	16000
Pre-emphasis coefficient	0.97
Hamming Window	25ms size 10ms shift
FFT size	512
No. of Mel filters	25
Dimension of MFCC	13
MFCC + Delta + Double Delta	39

3.3.2 IPA/Vietnamese Lexicon

A lexicon constrains the vocabulary of an ASR system and also provides a mapping between vocabulary and its pronunciation. Then, there are two factors in a lexicon: *vocabulary* and *pronunciation*. First, I need to extract every unique words from transcription of the speech corpus. Then, each unique word will be divided into subwords and be mapped to corresponding phonetic symbol as the following example:

tôm t͡ɔ7 m
ngon ŋ ɔ7 n
hơn h əː7 n
mì m i2
quện w e6 n
một m o6 t͡

Figure 3.2 – An example of a Vietnamese phonetic dictionary.

Table 3.4 – Vietnamese phonology based Espeak with potentially 209 phonemes in total

Consonant	b c d* đ g* h k l m n p s x t v ch kh ng* nh ph qu th tr
Vowel**	a ă â e ê o ô ơ u ư y ai ay ao au ây âu eo êu ia* iê iu oa* oe oi ôi ơi ơu ui ua* ưa*
Tone	a (none), à (grave), á (acute), ả (hook), ã (tilde), ạ (underdot)

* d = gi = r, g = gh, ng = ngh, ia = yê, oa=oo, ua=uô, ưa=ươ

** a vowel could have different tones

The phonetic mapping rule is following Espeak's rule[7], whose output can be represented as Vietnamese International Phonetic Alphabet (IPA/Vietnamese) (see Appendix A), which is such a standard phonetic symbol. In Vietnam, there are 3 main dialects: Northern dialect, Central dialect and Southern dialect. Three dialects have their own distinction, so there are different rules between them. Northern dialect phoneset consists of 23 consonants, 31 vowels, and 6 different tones, which can contain a total number of 209 unique phoneme for Vietnamese (see Table 3.4). The total number of phoneme in each dataset, which is described in the Table 3.5, may exceed 209 because there could exist non-Vietnamese words.

Table 3.5 – Number of phonemes in each dataset

Dataset	Training	Testing (OOP)
VIVOS	219	12%
FPT	253	11%
VoiceViet	229	6%

* OOP: Out-of-phonelist phones

3.3.3 Context-Dependent Tied State AM Training

The acoustic model provides HMMs of every phonemes existing in the speech corpus. Training tied-state triphone model in Sphinx requires successfully passing through 3 phases as shown in Figure 3.3

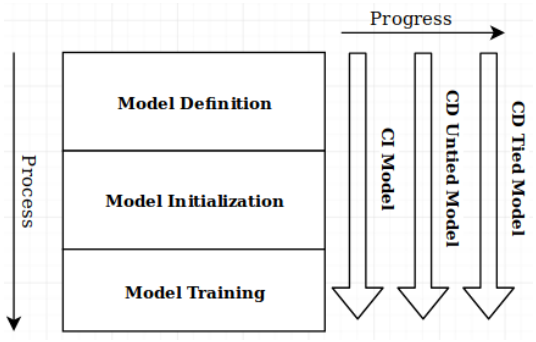


Figure 3.3 – Acoustic Model Training Progress.

In process, the first step is to prepare a model definition file which defines a unique numerical identity to every state of every HMM. Next step, 4 parameters of every state including Gaussian mixture weights, transition matrices, means and variances of all Gaussians are initialized before training. Last step, all parameters are re-estimated until converging by Baum-Welch algorithm. In progress, firstly, the acoustic model parameters are trained based on the Context-Independent (CI) monophones (extract directly from lexicon). Secondly, the acoustic model evolves into Context-Dependent (CD) untied triphones based on the CI monophone parameters. Lastly, the CD tied triphones is trained by reduce the number of state distributions through combining similar states. Two main parameters are required to identify while training acoustic model: number of **Gaussian Mixture components** and number of **Senones**, since they depend on the scale of training corpus. Later, I will describe in detail several setups to find the best combination of them. Besides, the linguistic questions that are needed for classifying phonemes are automatically designed by Sphinx. All state distributions of CI phones are clustered to obtain questions for

[7] https://github.com/espeak-ng/espeak-ng/blob/master/dictsource/vi_rules

tri-phonetic contexts *(Singh et al. 1999)*. Then, I do not have to manually specify linguistic questions for Vietnamese phonology. A trained acoustic model contains following files:

mdef *# model definition*
feat.params *# feature parameters*
mixture_weights *# GMM weights*
means *# global mean of Gaussian distributions*
noisedict *# a set of words made up of noise phones, e.g: <s> SIL*
transition_matrices *# HMM transition matrices*
variances *# global variance of Gaussian distributions*

Figure 3.4 – All files in an AM.

mdef file contains all senone-ids of every state of every phone (baseline monophone + triphone). *feat.params* file has all parameters of the acoustic feature. *mixture_weights* file stores GMM weights for all senones. *means* and *variances* files contain global mean and variance of gaussian components in each senone. *transition_matrices* file contains transitional probabilities in baseline HMM (monophone). Last, *noisedict* file contains non-speech sound units. The detail in each file can be seen in Appendix B.

3.3.4 Trigram LM Training

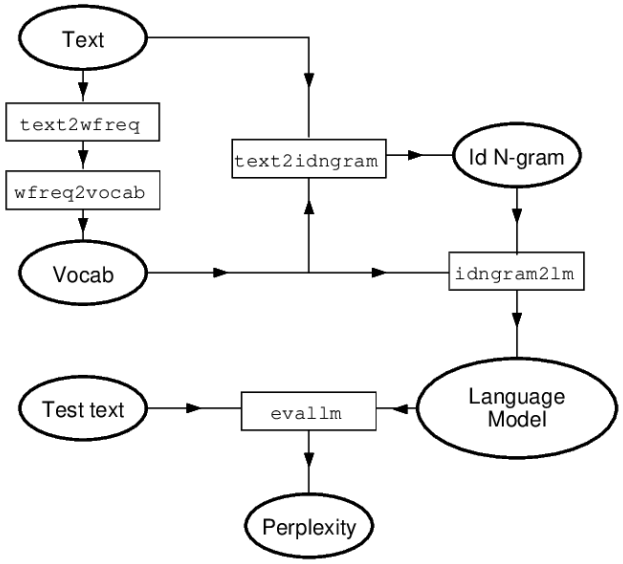


Figure 3.5 – Build LM Process.

The trigram language model provides the occurence probability of trigrams and also can back-off to find bigrams and unigrams. The process to build a trigram LM is illustrated in the Figure 3.5. Firstly, all unique words (unigrams) are counted and used as vocabulary. Next, bigrams and trigrams are counted and discounting using Good Turing Smoothing technique. Then every discounted n-gram is assigned with an ID and finally are converted into a binary format language model of standard ARPA format. Moreover, the language model can be evaluated by computing its perplexity with respect the the test text transcription of each dataset. There are 2 type of vocabulary for language model provided by Sphinx, which each handle Out-Of-Vocabulary (OOV) words in different ways. *A closed vocabulary* model does not make any computing for OOVs, which is suitable for a command/control application where the vocabulary is restricted to the number of commands that the system could understand. And, *an open vocabulary* model allows for OOVs to

occur, in which OOVs are treated as any other word in the vocabulary so it might be used in a spontaneous speech application such as voice search. In this project, I use an open vocabulary LM for my system as following example:

```
\data\
ngram 1=6001
ngram 2=175542
ngram 3=368859

\1-grams:
-2.7726 <UNK>     -0.2449
-1.1142 </s>      -4.6373
-1.1142 <s>       -0.9281
-3.7863 a          -0.4164
-4.7449 abc        -0.3707
-5.7863 ac         -0.2513
-5.3092 ada        -0.4355
....
\2-grams:
-3.3704 <UNK> ai -0.1150
-3.5262 <s> a     -0.2260
....
```

Figure 3.6 – An example of an open vocabulary LM in ARPA format.

In ARPA format, all probabilities and back-off weights are given in log10 form. All OOVs are mapped to UNK, referred to unknown words. The following table describes properties of each language model built on the 3 datasets:

Table 3.6 – Properties of 3 LM from VIVOS, FPT, VoiceViet

Dataset	Training			Testing	
	1-gram	2-gram	3-gram	OOV	Perplexity
VIVOS	4863	74899	129027	88 (2%)	98.43
FPT	5445	78878	138976	468 (9%)	220.36
VoiceViet	4691	79407	142804	331 (7%)	180.53

3.3.5 Pocketsphinx decoder

The decoder uses a three-pass search strategy consisting of the following stage: *fwdtree* - Viterbi search using a static lexicon tree, *fwdflat* - Viterbi search using a flat lexicon and *bestpath* - Word graph search. These search algorithms are implemented in Pocketsphinx (Huggins-Daines et al. 2006). Design elements of Pocketsphinx's decoder can be seen in Figure 3.7.

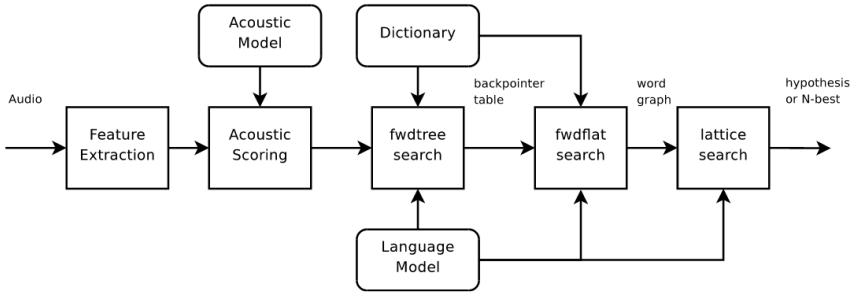


Figure 3.7 – Pocketsphinx Design Architecture (Huggins Daines, 2011).

Chapter 4

Experiments & Results

The experimental setups are divided into 3 stages. During first stage, one of the 3 datasets is used to improve speech corpus quality in order to compare the accuracy of low quality data versus high quality data. At second stage, the acoustic model is trained with different number of senone and gaussians to identify the best combination of them. In the last stage, I train a generic LM with more data, then constrain the LM by vocabulary of each datasets.

4.1 Improve Speech Corpus Quality

Table 4.1 – WER before and after realignment of VIVOS with 4000 senones and 16 gaussian components

Dataset	No-realignment	Realignment
VIVOS	78.48%	41.09%

"Quality" in speech corpus roughly means the alignment between audio and its transcription. During recording, people could easily misread the transcription, e.g: one says: "Xin chào các bạn", but its transcription is: "Xin chào chào các bạn". Or during modifying the record, they could easily cut some segments which make the audio and transcription mismatch. These mismatch data could affect the performance of the acoustic model because they will be ignored while training, then lower the accuracy result if there is too much mismatch data (or not enough sufficient data). Therefore, it is prior to check mismatch data and re-align them. To check mismatch data, there could be several errors during training acoustic model, such as: "*ERROR: "backward.c", line 430: Failed to align audio to transcript: final state of the search is not reached... VIVOSSPK27_089 ignored*", which are logged into a file. Table 4.2 show the number of mismatch data of 3 datasets.

Table 4.2 – Data quality of VIVOS, FPT, VoiceViet

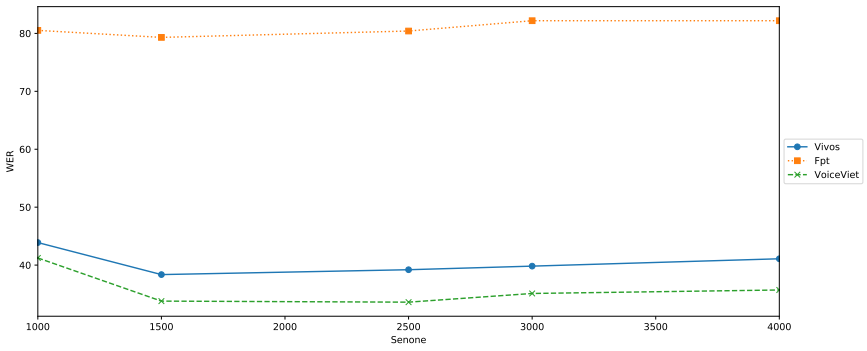
Dataset	No. of mismatch data
VIVOS	292 (2.5%)
FPT	1212 (6.96%)
VoiceViet	24 (0.13%)

The realignment process could be record another audio for a transcription or edit the transcription to match with its audio. So, it requires a lot of time if the dataset has too much mismatch data.

Therefore, I only choose one dataset VIVOS to do this experiment. The number of senones and gaussians in this experiment is chosen randomly with 4000 senones and 16 gaussians. The accuracy result on VIVOS dataset (after realigning) is shown in Table 4.1.

4.2 Identify best combination of GMM component and Senone

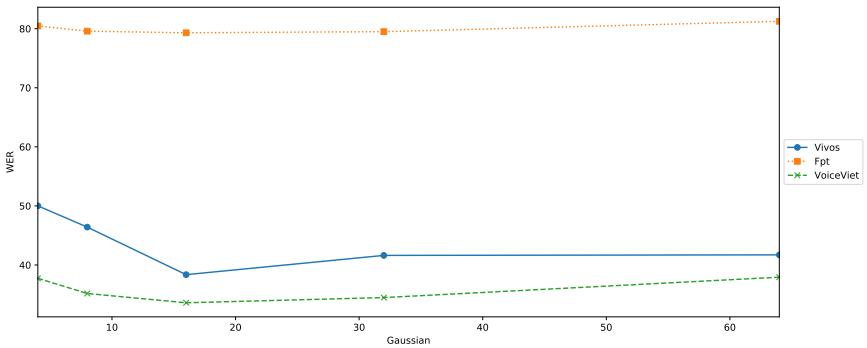
Dataset	Senone					Gaussian
	500	1500	2500	3000	4000	
VIVOS	43.89%	38.36%	39.20%	39.82%	41.09%	16
FPT	80.52%	79.31%	80.41%	82.19%	82.19%	16
VoiceViet	41.23%	33.78%	33.60%	35.10%	35.70%	16



*NOTE: Only Vivos dataset was re-aligned.

Figure 4.1 – WER in reference to number of Senone.

Dataset	Gaussian					Senone
	4	8	16	32	64	
VIVOS	50.01%	46.41%	38.36%	41.62%	41.71%	1500
FPT	80.47%	79.58%	79.31%	79.50%	81.25%	1500
VoiceViet	37.74%	35.17%	33.60%	34.48%	37.93%	2500



*NOTE: Only Vivos dataset was re-aligned.

Figure 4.2 – WER in reference to number of GMM component.

In this experiment, all datasets are used to identify the best combination of Gaussian mixture components and number of Senones. Firstly, I trained acoustic model on different number of senones, which are 1000, 1500, 2500, 3000 and 4000 (with fixed 16 gaussians) as shown in Figure 4.1. Then, the best number of senones will be used to identify the best number of gaussians. The number of gaussians varies in 4, 8, 16, 32 and 64 as shown in Figure 4.2.
In this work, the trained model on VoiceViet speech corpus obtains the lowest WER 33.6% with 2500 senones, followed by VIVOS's model with 38.36% and 1500 senones, finally FPT with 79.31% and 1500 senones. It is obvious that the VoiceViet's model show pretty well performance because it has the highest speech corpus quality. The FPT's model shows very poor performance because it has the lowest corpus quality, e.g: during training its acoustic model, there are thousands of audio having mismatch transcription. The VIVOS's model performs normally with the re-aligned data from experiment 4.1. After identifying the best number of GMM component, all datasets still have the lowest WER at 16 gaussians.

4.3 Language Model Improvement

At this stage, I increase the training data size of language model by merging the training text of 3 corpora, addition to the data which is crawled from VoiceViet API[8]. Once data are collected it must be cleaned: special characters removed, numbers/acronyms expanded to full text representation, sentences split on lines. The final size of data is approximately 200,000 sentences ($\approx$ 30 MB). Because this generic language model can contain a lot of OOVs corresponding to each specific dataset, it is better to limit generic LM by each dataset's vocabulary in order to improve the decoding speed. The Table 4.3 is showing the evaluation of generic LM on each testing text of each dataset, while the Figure 4.3 is displaying the corresponding decoding speed, e.g: with 0.1 xRT, decoding a 2 second audio's length takes 0.1*2 = 0.2 sec.

Table 4.3 – Evaluation of generic LM on each dataset with vocabulary constraint

(a) Perplexity

Dataset	Perplexity		
	Baseline	No Limit	Limit
VIVOS	98.43	-23%	-27%
FPT	220.36	-10%	-16%
VoiceViet	180.53	-76%	-76%

(b) Recognition Accuracy

Dataset	WER		
	Baseline	No Limit	Limit
VIVOS	38.36%	-9.91%	-9.85%
FPT	79.31%	+3.79%	+3.33%
VoiceViet	33.6%	-44.52%	-44.67%

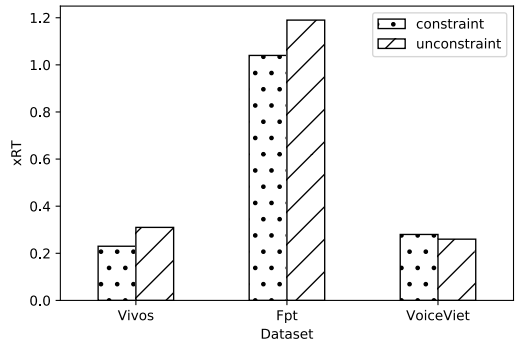


Figure 4.3 – Real Time Ratio (xRT) between constrained and unconstrained vocabulary LMs.

[8]https://voiceviet.itrithuc.vn/docs/

Chapter 5

Conclusion and Future work

In this thesis, I use CMU Sphinx toolkit to build a VASR engine based on the combination of context-dependent tied state triphone acoustic model (GMM-HMM) and trigram language model. To build Vietnamese phonetic dictionary, I use Espeak's mapping rule whose output is represented as IPA symbols.

5.1 Summary of Results

During the research, I try to execute 3 experiments to tune recognition performance: manually realigning speech corpus, benchmark the best combination of senones and gaussian components, crawling more text data to train language model. Firstly, it is proved that speech corpus quality affects numerously recognition accuracy. Secondly, the number of senones and gaussian components depends upon the amount of training data which covers enough phonetic properties. Thirdly, the good prior probability of words in language model also improve recognition performance, practically when training for a specific domain. Besides, the training data for language model is usually very large which can contain a lot of OOVs, so it is better to constrain its vocabulary. This reduces the decoding speed, and may increase the final accuracy. Finally, the best WERs are 34.56%, 79.31%, 18.59% for VIVOS, FPT and VoiceViet datasets respectively. It turns out that HMM perform badly on noisy environment rather than clean or quiet one.

5.2 Future Works

First and foremost, I stated that the trained model can be deployed easily to develop application in mobile platform (a simple demo is shown in Appendix C). Next, considering to build my own speech corpus is practical although it requires a lot of resources. Lastly, using experience and knowledge gained from this project, I can start to learn and use DNN instead of HMM using Kaldi for research purpose. For developing application purpose, using Kaldi is harder than CMU Sphinx. Therefore, there are still several methods to improve accuracy using traditional HMM in Sphinx, such as: hand-writing linguistic questions for Vietnamese, feature transformation (e.g: Linear Discriminant Analysis), other estimating parameter algorithm (e.g: Maximum Mutual Information Estimation), adapt acoustic model for noisy environment or particular speakers using MLLR, etc.

Bibliography

A. Bilmes, Jeff (2000). "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models." In: *Technical Report ICSI-TR-97-021, University of Berkeley* 4.

Do, Truong, Phuong Pham, Hoang Tung Tran, and Chi Luong (2018). "DEVELOPMENT OF HIGH-PERFORMANCE AND LARGE-SCALE VIETNAMESE AUTOMATIC SPEECH RECOGNITION SYSTEMS." In: *Journal of Computer Science and Cybernetics* V.34, pp. 335–348. DOI: 10.15625/1813-9663/34/4/13165.

Gaida, Christian et al. (2014). *Comparing Open-Source Speech Recognition Toolkits **.

Huang, Xuedong, Alex Acero, and Hsiao-Wuen Hon (2001). *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*.

Huggins-Daines, D. et al. (2006). "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices." In: 1, pp. I–I. ISSN: 1520-6149. DOI: 10.1109/ICASSP.2006.1659988.

Huggins Daines, David (2011). "An Architecture for Scalable, Universal Speech Recognition." AAI3528181. PhD thesis. Pittsburgh, PA, USA. ISBN: 978-1-267-58227-0.

Jurafsky, Daniel and James Martin (2018). "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition." In: chap. 3, p. 40.

Liao, Chun-Feng (2003). "Understanding the CMU Sphinx Speech Recognition System." In: pp. 3–4.

Luong, Hieu-Thi and Hai-Quan Vu (2016). "A non-expert Kaldi recipe for Vietnamese Speech Recognition System." In: *WLSI/OIAF4HLT2016*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 51–55. URL: https://www.aclweb.org/anthology/W16-5207.

Mehmood, Faizan and Saqib Khan (2014). *An Overview on Speech Recognition System and Comparative Study of its Approaches*.

Nguyen, H. Q., V. L. Trinh, and T. D. Le (2010). "Automatic Speech Recognition for Vietnamese Using HTK System." In: *2010 IEEE RIVF I. C. C. C. Technologies*, pp. 1–4. DOI: 10.1109/RIVF.2010.5633587.

Singh, R., B. Raj, and R. M. Stern (1999). "Automatic clustering and generation of contextual questions for tied states in hidden Markov models." In: *ICASSP99 (Cat. No.99CH36258)*. Vol. 1, 117–120 vol.1. DOI: 10.1109/ICASSP.1999.758076.

Stevens, S. S. and J. Volkmann (1940). "The relation of pitch to frequency: a revised scale." In: *The American Journal of Psychology* 53, pp. 329–353.

Thompson, C. (2017). "Open Source Toolkits for Speech Recognition." In: URL: https://www.svds.com/open-source-toolkits-speech-recognition/.

Appendix A

Vietnamese International Phonetic Alphabet

Initial consonants		Final consonants		Monophthongs		Diphthongs	
ɓ	**b**a	m	thê**m**	a	**ă**n	iə	b**ia, yê**u
ɗ	**đi**	n	ba**n**	aː	b**a**	yə	x**ươ**ng, ch**ưa**
f	**ph**ở	ŋ	trứ**ng**	e	v**ề**	uə	**uố**ng, m**ua**
ɣ	**g**a; **gh**ế	n	mi**n** (*after* /i, e/)	ɛ	x**e**	aːɪ	m**ai**
h	**h**àng	ŋ͡m	bì**nh**	ə	**â**n	aɪ	m**ay**
x	**kh**ô	p	tiế**p**	əː	b**ơ**	aːʊ	c**ao**
l	**l**à	t͡n	xuấ**t**	y	t**ư**	aʊ	c**au**
m	**m**ai	kh	á**c**	o	c**ô**	oɪ	h**ôi**
n	**n**am	t	chí**t** (*after* /i, e/)	ɔ	c**o**	əːɪ	h**ơi**
ɲ	**nh**à	c	cá**ch**	u	r**u**	əːʊ	qu**ơu**
ŋ	**ng**âm; **ngh**e	**Medial glide**		i	**i**m, **y**	əɪ	c**ây**
p	**p**in	ʷ	qu**ố**c	**Tones**		əʊ	c**âu**
s	**x**a			aː**7**	**a**	ɛʊ	l**eo**
t͡n	**t**ây			aː**6**	**ạ**	eʊ	k**êu**
t	**th**ầy			aː**5**	**ã**	iɛ	v**iê**n
tɕ	**ch**è			aː**4**	**ả**	iʊ	m**iu**
v	**v**ề			aː**3**	**á**	oə	l**oa**
z	**gi**à; **d**a; **r**a			aː**2**	**à**	oæ	h**oe**
g	**c**á					ɔɪ	t**oi**
ʂ	**s**en					uɪ	m**ui**
tʃ	**tr**e						

Figure A.1 – Hanoi Dialect IPA Symbols based on Espeak.

Appendix B

Acoustic Model Files Format

mdef

```
229 n_base # no. of base phones
2383528 n_tri # no. of triphones
9535028 n_state_map # no. of triphone states
3187 n_tied_state # no. of states of triphones after
                    state - sharing is done
687 n_tied_ci_state # no. of states of base phones
                      after state-sharing is done
229 n_tied_tmat # no. of transition matrices for
                  HMM of base phone
#
# Columns definitions
#base lft  rt p attrib tmat      ... state id's ...
  SIL     -   -   -  filler   0          0      1      2 N
   a2     -   -   -     n/a   1          3      4      5 N
    ...
   a2    SIL  ŋ  b     n/a    1    687    690    692 N
    ....
# base: name of each phone
# lft: left-context of the phone (- if none)
# rt: right-context of the phone (- if none)
# p: position of a triphone
      b = word beginning , e = word ending
      i = word internal , s = single word
# attrib:attribute of non-speech phone
# tmat: id of transition matrix associated with phone
# state id's: ids of HMM states associated with any
              phone. "N" stands for a non-emitting
             state
```

mixture_weights

```
mixw 3187 1 4 # 3187 senones, 1 feature stream
                4 gaussians per state

# Mixture weight for senone no. 0, feature-stream no. 0
mixw [0 0] 2.511435e+05 1.893e-01 3.142e-01 2.196e-01 2.769e-
01

# Mixture weight for senone no. 1, feature-stream no. 0
mixw [1 0] 2.663632e+05 1.980e-01 2.851e-01 3.251e-01 1.919e-
01
....
```

transition_matrices

```
tmat 229 4 # 229 HMMs, 4 states
( 3 emitting states +1 non emitting state)

tmat [0] # HMM no. 1
 8.572e-01 1.428e-01
         8.654e-01 1.346e-01
                  8.318e-01 1.682e-01

tmat [1] # HMM no. 2
 7.330e-01 2.670e-01
         5.854e-01 4.146e-01
                  7.114e-01 2.886e-01
...
```

means or variances (same format)

```
param 3187 1 4 # 3178 senones, 1 feature stream, 4 Gaussians per state
# Note: Each senone is a mixture of 4 gaussians.
# Means (variances) for senone no. 0, feature-stream no.0, Gaussian density no.0, followed by its 39-
dimensional mean vector (13 cepstra + 13 delta + 13 delta-delta).
mgau 0
feat 0
density    0 -1.700e+01 -8.064e+00 -6.620e-02 -7.559e+00 1.789e+00 -2.715e+00 -1.842e+00
-2.640e+00 2.254e+00 4.426e+00 1.480e+00 5.035e+00 2.139e+00 1.864e-01 8.999e-01 1.145e-01
7.230e-01 5.930e-01 3.790e-01 -3.428e-01 1.279e+00 2.279e+00 2.186e+00 2.756e+00 2.956e+00
1.358e+00 1.215e-01 5.859e-01 7.637e-01 2.166e+00 1.459e+00 5.257e-01 8.955e-01 2.938e+00
6.649e-01 -1.943e+00 -8.849e-01 -2.683e+00 -2.896e+00

#Gaussian density no. 1, followed by its 39-dimensional feature vector
...
#Gaussian density no. 3, followed by its 39-dimensional feature vector

# Means (variances) for senone no. 1, feature-stream no.0
# Gaussian density no. 0, followed by its 39-dimensional feature vector
# Gaussian density no. 1, followed by its 39-dimensional feature vector
....
```

feat.params

```
-lowerf 130
-upperf 6800
-nfilt 25
-transform dct
-lifter 22
-feat 1s_c_d_dd
-agc none
-cmn batch
-varnorm no
```

noisedic

```
<s> SIL
</s> SIL
<sil> SIL
```

Figure B.1 – An example of AM files format of VoiceViet dataset.

Appendix C

Android Application Demo

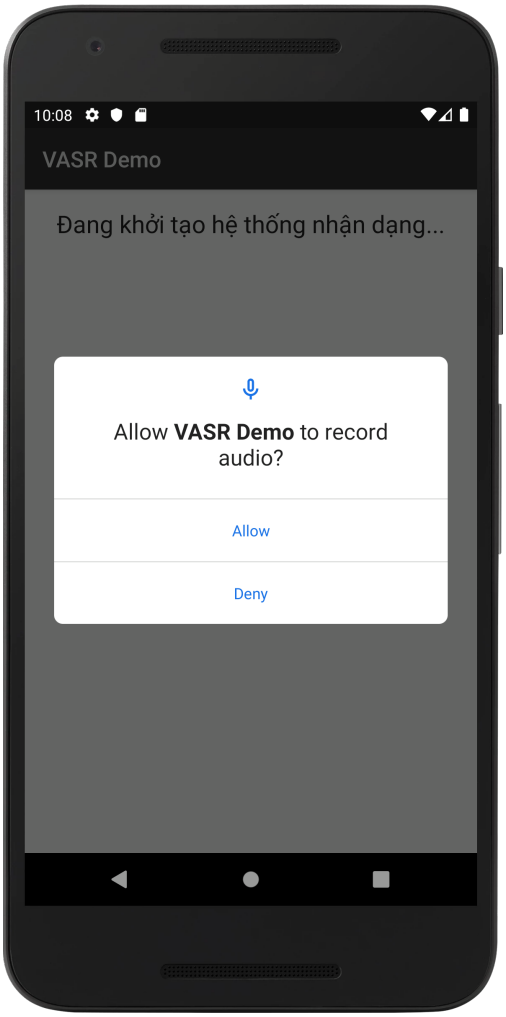


Figure C.1 – Require record audio permission

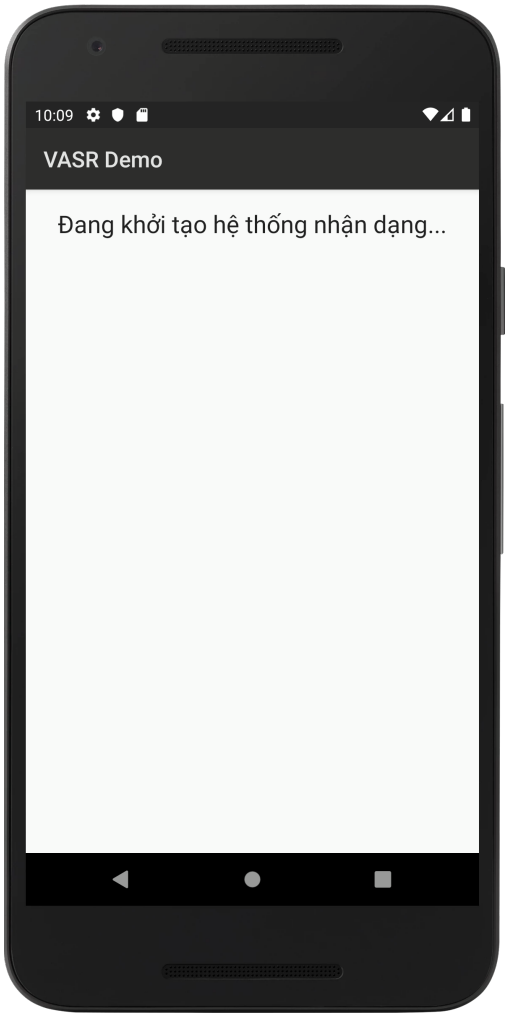


Figure C.2 – Initialize resources for decoding

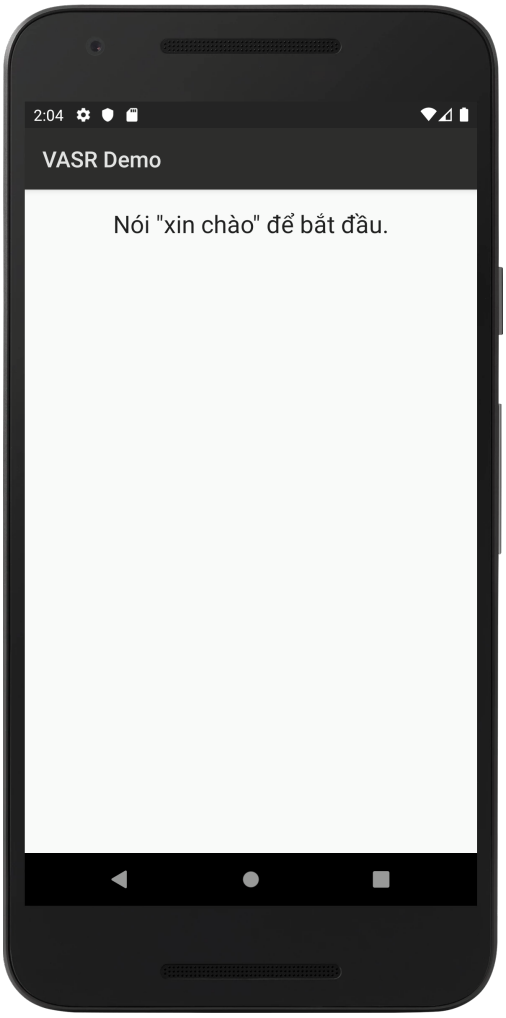


Figure C.3 – Use keyword to activate continuous recognition

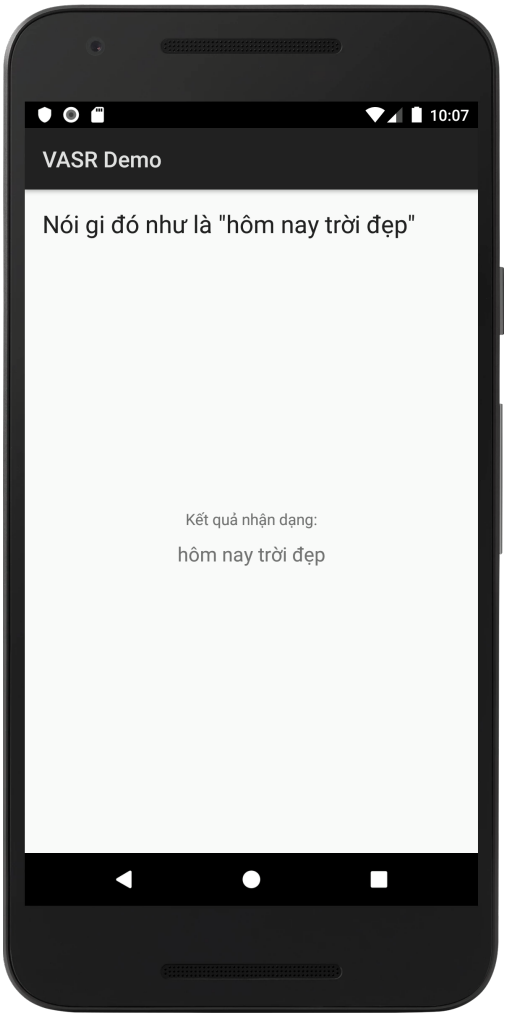


Figure C.4 – Recognized text of speech