BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC LẠC HỒNG
*** * ***

HUỲNH THANH GIÀU

NGHIÊN CỨU VỀ NHẬN DẠNG TIẾNG NÓI TIẾNG VIỆT VÀ ỨNG DỤNG THỬ NGHIỆM TRONG ĐIỀU KHIỂN MÁY TÍNH

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Đồng Nai, năm 2012

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC LẠC HỒNG
*** * ***

HUỲNH THANH GIÀU

NGHIÊN CỨU VỀ NHẬN DẠNG TIẾNG NÓI TIẾNG VIỆT VÀ ỨNG DỤNG THỬ NGHIỆM TRONG ĐIỀU KHIỂN MÁY TÍNH

Chuyên ngành: Công nghệ Thông tin

Mã số: 60.48.02.01

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. VŨ ĐỨC LUNG

Đồng Nai, năm 2012

LỜI CÁM ƠN

Đầu tiên, em xin bày tỏ lòng biết ơn chân thành đến thầy Vũ Đức Lung, người đã tận tình hướng dẫn, tạo mọi điều kiện thuận lợi để em hoàn thành tốt luận văn tốt nghiệp này.

Em cũng xin cảm ơn sự dạy dỗ và giúp đỡ tận tình của tất cả quí thầy cô tại trường Đại học Lạc Hồng. Tất cả các kiến thức mà em được truyền đạt sẽ là hành trang quí giá trên con đường học tập, làm việc và nghiên cứu sau này.

Em xin được tri ơn tất cả.

Đồng Nai, tháng 8 năm 2012

Học viên

Huỳnh Thanh Giàu

TÓM TẮT LUẬN VĂN

Nghiên cứu nhận dạng tiếng nói đã được các nước trên thế giới thực hiện rất nhiều năm qua và cũng đã có những thành công nhất định. Ở Việt Nam cũng có nhiều công trình nghiên cứu và thử nghiệm, tuy nhiên, các kết quả vẫn còn hạn chế và cần có nhiều nghiên cứu nữa trong vấn đề này.

Nhằm tìm hiểu những phương pháp nhận dạng tiếng nói tiếng Việt để đóng góp một phần nhỏ vào những công trình nghiên cứu đó, luận văn muốn nghiên cứu về nhận dạng tiếng nói tiếng Việt và ứng dụng thử nghiệm trong giao tiếp với máy tính để có thể nhận dạng tiếng nói tiếng Việt bằng việc sử dụng mô hình Markov ẩn dựa trên nền tảng CMUSphinx của đại học Carnegie Mellon.

Luận văn chủ yếu nghiên cứu về tiếng nói, các phương pháp xử lý tiếng nói, rút trích đặc trưng tiếng nói bằng MFCC (Mel-scale Frequency Cepstral Coefficient) và LPC (Linear Predictive Coding), mô hình Markov ẩn, mô hình âm học, âm vị áp dụng cho tiếng Việt. Luận văn cũng tìm hiểu về kiến trúc hệ thống nhận dạng tiếng nói qua công cụ Sphinx và sử dụng công cụ đó để thử nghiệm cho việc nhận dạng tiếng nói tiếng Việt.

Qua nghiên cứu, luận văn đã nắm được cách xử lý tiếng nói, mô hình, phương pháp nào là tương đối tốt nhất cho việc nhận dạng tiếng nói tiếng Việt. Bên cạnh đó, luận văn cũng xây dựng được một chương trình demo để minh họa cho những hiểu biết của mình về nhận dạng tiếng nói tiếng Việt.

Trong thời gian hạn chế với mức độ phức tạp của vấn đề nhận dạng tiếng nói tiếng Việt, luận văn này chỉ là bước nghiên cứu ban đầu cho nhận dạng tiếng nói tiếng Việt.

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH VẼ

MỞ ĐẦU

Tiếng nói là phương tiện giao tiếp cơ bản nhất của con người, sử dụng lời nói là một cách diễn đạt đơn giản và hiệu quả nhất. Đã từ lâu, con người luôn mơ ước đến các hệ thống máy điều khiển tự động có thể giao tiếp bằng tiếng nói tự nhiên của con người. Ngày nay, cùng với sự phát triển của khoa học kỹ thuật và công nghệ, đặc biệt trong lĩnh vực tin học. Các hệ thống máy tự động đã dần thay thế con người trong nhiều công việc. Nhu cầu giao tiếp với thiết bị máy bằng tiếng nói là rất cần thiết, đó là phương thức giao tiếp văn minh và tự nhiên nhất.

Nhận dạng tiếng nói là một vấn đề không mới. Trên thế giới đã và đang có có rất nhiều công trình nghiên cứu về vấn đề này với rất nhiều phương pháp nhận dạng tiếng nói khác nhau. Và những nghiên cứu đó cũng có những thành công đáng kể. Có thể kể đến như: hệ thống nhận dạng tiếng nói tiếng Anh Via Voice của IBM, Spoken Toolkit của CSLU (Central of Spoken Laguage Under-standing), Speech Recognition Engine của Microsoft, Hidden Markov Model toolkit của đại học Cambridge, CMU Sphinx của đại học Carnegie Mellon,… ngoài ra, một số hệ thống nhận dạng tiến nói tiếng Pháp, Đức, Trung Quốc,... cũng khá phát triển. Tiếng Việt thì cũng có một số công trình của các nhóm như: AILab, Vietvoice, Vspeech… Nhưng đối với nước ta, nhận dạng tiếng nói vẫn là một lĩnh vực khá mới mẻ. Đến nay tuy đã có nhiều nghiên cứu về nhận dạng tiếng nói tiếng Việt và đã đạt được một số thành tựu, nhưng nhìn chung vẫn chưa đạt được kết quả cần thiết để có thể tạo ra các sản phẩm mang tính ứng dụng cao.

Với mong muốn có thể hiểu được cách giao tiếp giữa người và máy tính, luận văn này nghiên cứu các phương pháp nhận dạng tiếng nói, từ đó xây dựng một chương trình demo nhận dạng tiếng nói tiếng Việt mà khi con người nói máy tính có thể hiểu được.

CHƯƠNG 1: TỔNG QUAN

1.1. TỔNG QUAN TÌNH HÌNH TRONG VÀ NGOÀI NƯỚC

Vấn đề nghiên cứu các phương pháp nhận dạng tiếng nói đã và đang thu hút rất nhiều sự đầu tư và nghiên cứu của các nhà khoa học trên khắp thế giới. Ý tưởng về xây dựng các hệ thống nhận dạng tiếng nói đã có từ những năm 50 của thế kỷ 20 và đến nay đã đạt được nhiều kết quả đáng kể.

Trên thế giới đã có rất nhiều hệ thống nhận dạng tiếng nói tiếng Anh đã và đang được ứng dụng rất hiệu quả như: Via Voice của IBM, Spoken Toolkit của CSLU (Central of Spoken Laguage Under-standing), Speech Recognition Engine của Microsoft, Hidden Markov Model toolkit của đại học Cambridge, CMU Sphinx của đại học Carnegie Mellon,… ngoài ra, một số hệ thống nhận dạng tiến nói tiếng Pháp, Đức, Trung Quốc,… cũng khá phát triển.

Đối với nước ta, nhận dạng tiếng nói vẫn là một lĩnh vực khá mới mẻ. Đến nay tuy đã có nhiều nghiên cứu về nhận dạng tiếng nói tiếng Việt và đã đạt được một số thành tựu, nhưng nhìn chung vẫn chưa đạt được kết quả cần thiết để có thể tạo ra các sản phẩm mang tính ứng dụng cao. Có thể kể đến các công trình sau:

- AILab: Đây là công trình được phòng thí nghiệm Trí tuệ Nhân tạo - AILab thuộc Đại học Khoa học Tự nhiên tạo ra dựa trên các công nghệ tiên tiến nhất về nhận dạng và tổng hợp tiếng nói để đáp ứng nhu cầu của người dùng. Dựa trên công nghệ xử lí tiếng nói tiếng Việt, AILab đã xây dựng phần mềm iSago chuyên hỗ trợ tìm kiếm thông tin qua tiếng nói.

Thông qua ứng dụng phần mềm người sử dụng có khả năng hỗ trợ giao tiếp với điện thoại di động trực tiếp bằng lời nói. Từ đó người sử dụng tìm kiếm thông tin nhà hàng, quán Bar, Café trên địa bàn TP. HCM.

Khi người dùng đặt câu hỏi bằng tiếng nói, iSago sẽ truyền nội dung truy vấn này về server để xử lý và gửi lại kết quả tìm kiếm, dạng một danh sách: tên nhà hàng, địa chỉ.

Phần mềm này cũng cho phép người dùng hiển thị địa chỉ tìm được dạng bản đồ hoặc nghe đọc địa chỉ trực tiếp bằng công nghệ tổng hợp giọng nói. Phần

mềm được cung cấp miễn phí tại địa chỉ www.ailab.hcmus.edu.vn (http://www.ailab.hcmus.edu.vn)

- Vietvoice: Đây là phần mềm của một người dân Việt Nam ngụ tại Canada. Phần mềm có khả năng nói tiếng Việt từ các tập tin. Để chạy được chương trình, cần cài đặt Microsoft Visual C++ 2005 Redistributable Package (x86). Đối với người khiếm thị, phần mềm này cho phép sử dụng cách gõ tắt (nhấn nút Ctrl và một chữ) để chọn lựa một trong các tính năng hiển thị trên màn hình. Người dùng có thể cập nhật từ điển các chữ viết tắt và các từ ngữ tiếng nước ngoài.

- Vspeech: Đây là một phần mềm điều khiển máy tính bằng giọng nói do một nhóm sinh viên Đại học Bách Khoa TP. HCM viết. Phần mềm sử dụng thư viện Microsoft Speech SDK để nhận dạng tiếng Anh nhưng được chuyển thành tiếng Việt. Nhóm đã khá thành công với ý tưởng này, do sử dụng lại thư viện nhận dạng engine nên thời gian thiết kế rút ngắn lại mà hiệu quả nhận dạng khá tốt. Phần mềm Vspeech có các lệnh gọi hệ thống đơn giản như gọi thư mục My Computer, nút Start,… Phiên bản mới nhất có tương tác với MS Word 2003, lướt web với trình duyệt Internet Explorer. Không có các chức năng tùy chỉnh lệnh và gọi tắt các ứng dụng. Phần mềm chạy trên nền Windows XP, microphone và card âm thanh sử dụng tiêu chuẩn thông thường.

Tuy nhiên việc ứng dụng nhận dạng giọng nói vào điều khiển máy tính còn nhiều hạn chế. Ở Việt Nam thì hầu như chỉ mới có bộ phần mềm Vspeech của nhóm sinh viên trường Đại học Bách Khoa TP. HCM, các phần mềm khác chỉ thử nghiệm trong phòng thí nghiệm, chưa được sử dụng thực tế vì chưa đạt trên 100 từ. Phần mềm Vspeech được phát triển từ mã nguồn mở Microsoft Speech SDK nhận dạng tiếng Anh, thông qua dữ liệu, phương thức trung gian, việc nhận dạng được chuyển trong Vspeech để nhận biết tiếng Việt.

1.2. MỤC ĐÍCH ĐỀ TÀI

Luận văn nghiên cứu những ý tưởng cơ bản và các phương pháp được sử dụng trong nhận dạng tiếng nói từ đó xây dựng một chương trình demo nhận dạng khoảng 20 từ tiếng Việt dùng để điều khiển máy tính bằng giọng nói.

Luận văn gồm 05 chương:

Chương 1: Tổng quan về tình hình trong và ngoài nước liên quan đến việc nhận dạng tiếng nói, mục tiêu đề tài và giới hạn của đề tài.

Chương 2: Trình bày một số kiến thức cơ bản về xử lý tín hiệu số, biểu diễn tiếng nói trên ảnh phổ và phương pháp rút trích đặc trưng tiếng nói bằng phương pháp MFCC (Mel-scale Frequency Cepstral Coefficient) và LPC (Linear Predictive Coding).

Chương 3: Tiếp cận phương pháp nhận dạng tiếng nói dựa trên mô hình Markov ẩn bao gồm khái niệm, sử dụng thực tế và một số hạn chế của nó. Bên cạnh đó cũng đề cập đến 2 mô hình quan trọng xây dựng nên bộ ngôn ngữ cho hệ thống nhận dạng là mô hình âm học và mô hình ngôn ngữ.

Chương 4: Giới thiệu về công cụ hỗ trợ nhận dạng tiếng nói CMUSphinx của đại học Carnegie Mellon, các thành phần trong kiến trúc của nó để có được cái nhìn tổng quan về một hệ thống nhận dạng tiếng nói, đồng thời hỗ trợ cho việc xây dựng chương trình demo nhận dạng tiếng nói.

Chương 5: Xây dựng chương trình demo nhận dạng tiếng nói tiếng Việt sử dụng công cụ Sphinx, trong đó mô tả quá trình xây dựng mô hình ngôn ngữ và huấn luyện mô hình âm học cho chương trình nhận dạng.

Phụ lục: Bảng phiên âm phiên âm tiếng Việt mức âm vị theo dạng ASCII dựa trên bảng mẫu tự phiên âm quốc tế IPA (International Phonetic Alphabet) được sử dụng trong chương trình.

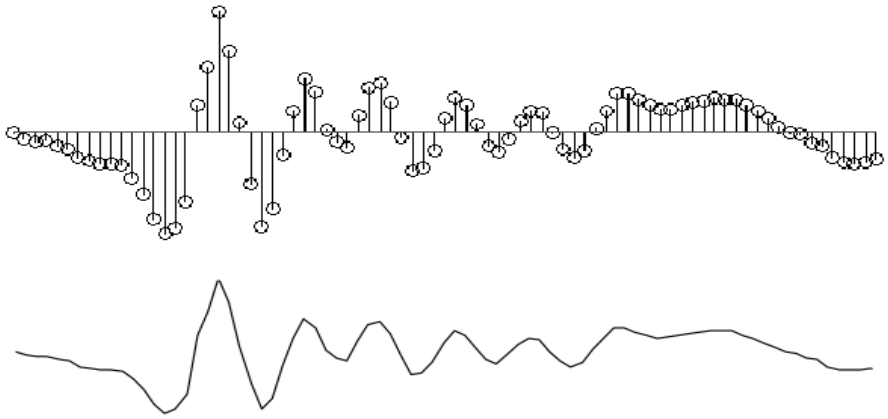
1.3. GIỚI HẠN ĐỀ TÀI

Luận văn chỉ giới hạn trong việc tìm hiểu về tiếng nói, các phương pháp xử lý tiếng nói, rút trích đặc trưng tiếng nói; mô hình Markov ẩn, mô hình âm học, âm vị áp dụng cho tiếng Việt; kiến trúc hệ thống nhận dạng tiếng nói qua công cụ Sphinx. Chương trình demo chỉ dừng ở mức nhận dạng được khoảng 100 câu lệnh cơ bản điều khiển máy tính (được liệt kê ở chương 5). Khi một người đọc lệnh điều khiển, máy tính sẽ hiểu và xuất hiện dòng lệnh đó trên màn hình của chương trình.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT XỬ LÝ TIẾNG NÓI

2.1. CƠ SỞ XỬ LÝ TÍN HIỆU SỐ

Định nghĩa một tín hiệu analog $x_a(t)$ dưới dạng hàm biến đổi liên tục theo thời gian: Nếu ta lấy mẫu tín hiệu x với một khoảng thời gian lấy mẫu T (tức là t = nT), ta có thể xác định một tín hiệu thời gian rời rạc $x(n) = x_a(nt)$. Hơn nữa ta có thể xác định tần số F_s như $F_s = 1/T$, nghịch đảo của khoảng thời gian lấy mẫu T.



Hình 2.1. Tín hiệu analog và tín hiệu số tương ứng

2.1.1. Các hệ thống và tín hiệu số:

2.1.1.1. Các tín hiệu dạng *sin*:

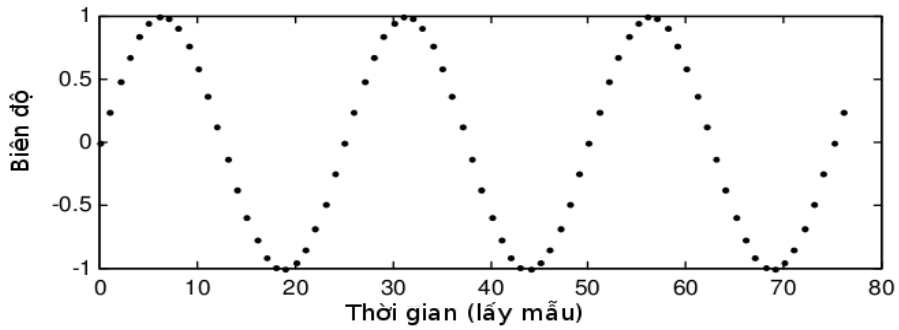
Một trong những tín hiệu quan trọng là sóng dạng *sin* hay đường hình *sin*:

$$x_0(n) = A_0\cos(\omega_0 n + \varphi_0) \quad (2.1)$$

Trong đó A_0 là biên độ của đường *sin*, ω_0 là tần số góc, và φ_0 là pha.

Đơn vị đo góc là *radian*, do đó tần số góc ω_0 liên hệ với tần số f_0 bởi công thức $\omega_0 = 2\pi f_0$ và $0 \leq f_0 \leq 1$

Tín hiệu này tuần hoàn với chu kỳ $T_0 = 1/f_0$.



Hình 2.2. Đường hình sin với chu kỳ 25 mẫu

Đường hình *sin* đóng vai trò quan trọng vì các tín hiệu tiếng nói có thể được phân tách thành các tổng của các đường *sin*. Tổng của 2 đường *sin* $x_0[n]$ và $x_1[n]$ có cùng tần số góc ω_0 nhưng khác biên độ A_0, A_1, và pha φ_0, φ_1 là một đường *sin* khác có cùng tần số nhưng khác biên độ A và pha φ.

Đường hình *sin* ở công thức (2.1) được diễn đạt theo phần thực của hàm số mũ phức tương ứng như sau:

$$x_0[n] = A_0\cos(\omega_0 n + \phi_0) = \mathrm{Re}A_0 e^{j(\omega_0 n+\phi_0)} \quad (2.2)$$

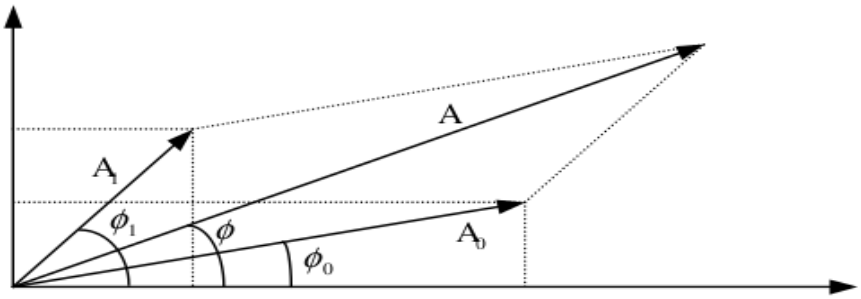
Trong đó: $j = \sqrt{-1}$

Do đó tổng của hai tín hiệu hàm mũ phức là:

$$A_0 e^{j(\omega_0 n+\phi_0)} + A_1 e^{j(\omega_0 n+\phi_1)} = e^{j\omega_0 n}(A_0 e^{j\phi_0} + A_1 e^{j\phi_1}) = e^{j\omega_0 n} A e^{j\phi} = A e^{j(\omega_0 n+\phi)} \quad (2.3)$$

Lấy phần thực của 2 vế ta có:

$$A_0\cos(\omega_0 n + \phi_0) + A_1\cos(\omega_0 n + \phi_1) = A\cos(\omega_0 n + \phi) \quad (2.4)$$



Hình 2.3. Biểu diễn tổng của hai đường sin cùng tần số

Để tính A và φ , ta có các công thức:

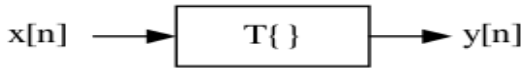
$$A^2 = {A_0}^2 + {A_1}^2 + 2A_0A_1\cos(\phi_0 - \phi_1) \quad (2.5)$$

$$\tan\phi = \frac{A_0\sin(\phi_0)+A_1\sin(\phi_1)}{A_0\cos(\phi_0)+A_1\sin(\phi_1)} \quad (2.6)$$

2.1.1.2. Hệ thống số:

Một hệ thống số là một hệ thống mà cho một tín hiệu đầu vào x(n), phát sinh một tín hiệu đầu ra y(n):

$$y(n) = T\{x(n)\} \quad (2.7)$$



Hình 2.4. Sơ đồ khối của một hệ thống kỹ thuật số

Nói chung, một hệ thống số được định nghĩa tuyến tính nếu và chỉ nếu:

$$T\{a_1x_1(n) + a_2x_2(n)\} = a_1T\{x_1(n)\} + a_2T\{x_2(n)\} \quad (2.8)$$

Với bất cứ giá trị a_1, a_2 và bất cứ tín hiệu $x_1(n)$, $x_2(n)$.

Hệ thống tuyến tín bất biến *(Linear Time-Invariant - LTI)* được mô tả như sau:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(k) \quad (2.9)$$

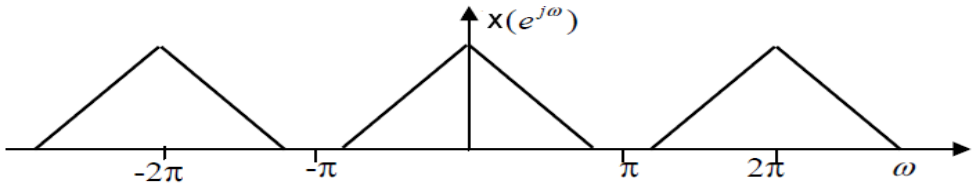
Trong đó * là phép nhân chập. Đây là phép toán quan trọng nhất trong xử lý tín hiệu để xác định đầu ra y(n) hệ thống khi biết đầu vào x(n) và đáp ứng xung h(n). Phép chập có tính chất: giao hoán, phân phối, kết hợp.

2.1.2. Phép biến đổi tần số liên tục:

2.1.2.1. Biến đổi Fourier:

Biến đổi Fourier của một tín hiệu x(n) được định nghĩa như sau:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jwn} \quad (2.10)$$



Hình 2.5. Đồ thị hàm $X(e^{j\omega})$

Vì $e^{j\omega} = cos\omega + jsin\omega$ tuần hoàn với chu kỳ 2π, do vậy khi thể hiện $X(e^{j\omega})$ ta chỉ cần thể hiện với dải từ 0 đến 2π hoặc từ -π đến π rồi lấy tuần hoàn.

Các cách thể hiện $X(e^{j\omega})$:

+ Biểu diễn theo phần thực phần ảo Re, Im

$$X(e^{j\omega}) = \Re[X(e^{j\omega})] + j\Im[X(e^{j\omega})] \quad (2.11)$$

+ Biểu diễn theo Module và Argument :

$$X(e^{j\omega}) = |X(e^{j\omega})|.e^{j\arg[X(e^{j\omega})]} \quad (2.12)$$

+ Biểu diễn theo độ lớn và pha:

Độ lớn có thể lấy giá trị âm và dương.

$$X(e^{j\omega}) = A(e^{j\omega}).e^{j\theta(\omega)} \quad (2.13)$$

Sự tồn tại của biến đổi Fourier: Căn cứ vào các tính chất hội tụ của chuỗi và

sự ánh xạ đầy đủ từ miền thời gian rời rạc n sang miền tần số ω (tức là khi sang miền tần số ω, chỉ tồn tại biến ω chứ không tồn tại biến n), ta có:

Biến đổi Fourier của một dãy x(n) sẽ tồn tại nếu và chỉ nếu:

$\sum_{n=-\infty}^{\infty}|x(n)| < \infty$ (Có nghĩa là chuỗi $\sum_{n=-\infty}^{\infty}|x(n)|$ hội tụ).

Biến đổi Fourier ngược (IFT: Inverse Fourier Transform):

Biến đổi Fourier ngược của phổ tín hiệu X e jω được định nghĩa như sau:

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega \qquad (2.14)$$

Ở đây biến đổi Fourier ngược giúp ta xác định được *x(n)* từ $X(e^{j\omega})$.

Bảng 2.1. Các tính chất của biến đổi Fourier

Miền n	**Miền ω**
$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$	$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jwn}$
$ax_1(n) + bx_2(n)$; (a, b: hằng số)	$aX_1(e^{j\omega}) + bX_2(e^{j\omega})$
$x(n - n_0)$	$e^{-j\omega n_0}X(e^{j\omega})$
x(n) là thực (tính chất đối xứng)	$X^*(e^{j\omega}) = X(e^{-j\omega})$
	Re $[X(e^{j\omega})]$ = Re $[X(e^{-j\omega})]$
	Im $[X(e^{j\omega})]$ = Im $[X(e^{-j\omega})]$
	$\lvert X(e^{j\omega})\rvert = \lvert X(e^{-j\omega})\rvert$
	arg $[X(e^{j\omega})]$ = -arg $[X(e^{-j\omega})]$
$x^*(n)$	$X^*(e^{-j\omega})$
$x(-n)$	$X(e^{-j\omega})$
$x_1(n) * x_2(n)$	$X_1(e^{j\omega}) . X_2(e^{j\omega})$
$x_1(n) . x_2(n)$	$\frac{1}{2\pi}\int_{-\pi}^{\pi} X_1(e^{j(\omega-\omega')}).X_2(e^{j\omega'})d\omega'$
$nx(n)$	$j\frac{dX(e^{j\omega})}{d\omega}$
$e^{j\omega_0 n}x(n)$	$X[e^{j(\omega-\omega_0)}]$
$x(n)\cos\omega_0 n$	$\frac{1}{2}X[e^{j(\omega-\omega_0)}] + \frac{1}{2}X[e^{j(\omega-\omega_0)}]$

$\sum_{n=-\infty}^{\infty} x_1(n).x_2^*(n)$	$\frac{1}{2}\int_{-\pi}^{\pi} X_1(e^{j\omega}).X_2^*(e^{j\omega})d\omega$
Quan hệ Parseval $\sum_{n=-\infty}^{\infty} \lvert x(n)\rvert^2$	$\frac{1}{2}\int_{-\pi}^{\pi} \lvert X(e^{j\omega})\rvert^2 d\omega$

2.1.2.2. Biến đổi Z:

Định nghĩa: Biến đổi z của một dãy x(n) được định nghĩa như sau:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (2.15)$$

Định nghĩa trên còn được gọi là biến đổi z hai phía.

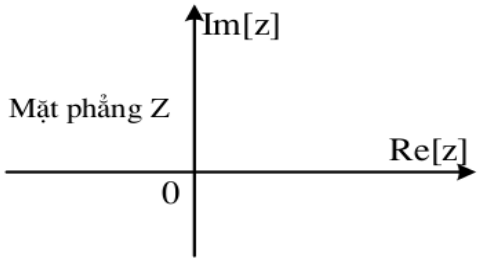
Ta sẽ có biến đổi z một phía nếu thay đổi cận n chạy từ 0 đến +∞:

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n} \quad (2.16)$$

Ở đây ta phải thấy được z là một biến số phức được biểu diễn theo 2 dạng:

+ Biểu diễn theo phần thực, phần ảo Re[z], Im[z]

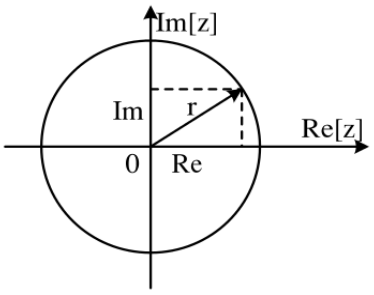
$$z = \text{Re}[z] + j.\text{Im}[z] \quad (2.17)$$



Hình 2.6. Biểu diễn theo phần thực phần ảo

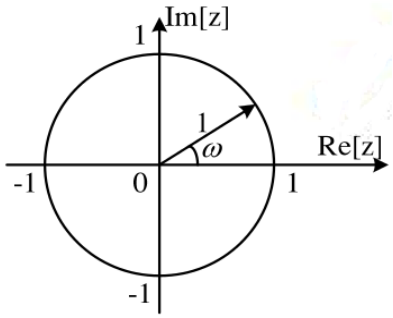
+ Biểu diễn theo tọa độ cực:

$$z = re^{j\omega} = r(\cos\omega + j\sin\omega) = r\cos\omega + j\sin\omega = \Re[z] + \Im[z] \quad (2.18)$$



Hình 2.7. Biểu diễn Z trên mặt phẳng phức

- Trường hợp đặc biệt: z = r = 1 , ta có vòng tròn đơn vị.



Hình 2.8. Vòng tròn đơn vị

Miền hội tụ của biến đổi z: Tập hợp tất cả các giá trị của z mà tại đó chuỗi

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{2.19}$$

hội tụ được gọi là miền hội tụ của biến đổi z. Ký hiệu RC *(Region of Convergence)*.

Biến đổi z ngược (IZT: Inverse Z Transform):

$$IZT[X(z)] = x(n)$$

Biến đổi z ngược được định nghĩa như sau:

$$x(n) = \frac{1}{2\pi j}\oint_C X(z).z^{n-1}dz \tag{2.20}$$

$\oint_C$ - Đường cong kín đi qua gốc tọa độ. Tích phân đường theo chiều dương.

Bảng 2.2. Các tính chất của biến đổi Z

Miền n	**Miền Z**
$x(n) = \frac{1}{2\pi j}\oint_C X(z)z^{-1}$	$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$
$ax_1(n) + bx_2(n)$; a, b là hằng số	$aX_1(z) + bX_2(z)$
$x(n-n_0)$	$z^{-n_0}X(z)$
$a^n x(n)$	$X(a^{-1}z)$
$nx(n)$	$-z\frac{dX(z)}{dz}$
$x^*(n)$; (*: liên hợp phức)	$X^*(z^*)$
$x(-n)$	$X\left(\frac{1}{z}\right)$
$x_1(n) * x_2(n)$	$X_1(z).X_2(z)$
$x_1(n).x_2(n)$	$\frac{1}{2\pi j}\oint_C X_1(v)X_2\left(\frac{z}{v}\right)v^{-1}dv$

2.1.2.3. Quan hệ giữa biến đổi Fourier và biến đổi Z

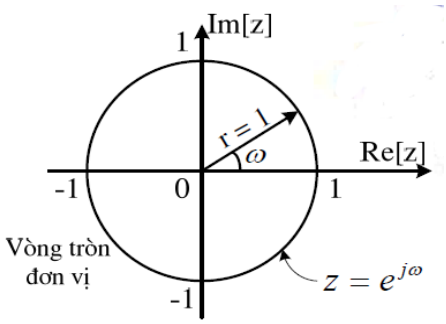
Ta thấy, theo định nghĩa của biến đổi z:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (2.21)$$

Mặt khác z là một biến số phức và được biểu diễn trong mặt phẳng phức theo toạ độ cực như sau: $z=r.e^{j\omega}$

Nếu chúng ta đánh giá biến đổi Z trên vòng tròn đơn vị (r=1), ta có:

$$X(z)\,|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} x(n).e^{-j\omega n} = X(e^{j\omega}) \quad (2.22)$$



Hình 2.9. Thực hiện biến đổi z trên vòng tròn đơn vị

Như vậy, ta rút ra một số nhận xét:

- Biến đổi Fourier chính là biến đổi z được thực hiện trên vòng tròn đơn vị.

- Biến đổi Fourier chỉ là trường hợp riêng của biến đổi z.

- Chúng ta có thể tìm biến đổi Fourier từ biến đổi Z bằng cách đánh giá ZT trên vòng tròn đơn vị với điều kiện vòng tròn đơn vị phải nằm trong miền hội tụ của biến đổi Z.

2.1.3. Phép biến đổi tần số rời rạc:

2.1.3.1. Biến đổi Fourier rời rạc (Discrete Fourier Transform - DFT):

Nếu một tín hiệu $x_N(n)$ tuần hoàn với chu kỳ N thì:

$$x_N(n) = x_N(n+N) \quad (2.23)$$

Biến đổi Fourier rời rạc của một dãy tuần hoàn $x_N(n)$ có chu kỳ N được định nghĩa như sau:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x(n).e^{-j\omega_k n} \quad (2.24)$$

Trong đó: $\omega_k = \frac{2\pi}{N}k$ với $\begin{cases} k = 0 \div N-1 \\ n = 0 \div N-1 \end{cases}$

x(n) là dãy tuần hoàn chu kỳ N nên nó thỏa mãn: x(n) = x(n + lN)

Đặt $W_N{}^{kn} = e^{-j\omega_k n} = e^{-j\frac{2\pi}{N}kn}$

$$W_N{}^{-kn} = e^{j\omega_k n} = e^{j\frac{2\pi}{N}kn}$$

$$W_N = e^{-j\frac{2\pi}{N}}; W_N{}^{-1} = e^{j\frac{2\pi}{N}}; W_N{}^{0} = 1$$

Theo cách đặt như trên thì biến đổi Fourier rời rạc đối với dãy tuần hoàn chu kỳ N được viết lại như sau:

$$X(k) = \sum_{n=0}^{N-1} x(n).W_N{}^{kn} \quad (2.25)$$

Biến đổi Fourier rời rạc ngược (IDFT):

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).e^{j\frac{2\pi}{N}kn} \quad (2.26)$$

Hay viết lại cho gọn:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).W_N{}^{-kn} \quad (2.27)$$

Bảng 2.3. Tính chất của DFT đối với dãy tuần hoàn có chu kỳ N

Miền n	**Miền k**
$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).W_N{}^{-kn}$	$X(k) = \sum_{n=0}^{N-1} x(n).W_N{}^{kn}$
$ax_1(n)_N + bx_2(n)_N$	$aX_1(k)_N + bX_2(k)_N$
$x(n-n_0)_N$	$W_N{}^{kn_0}X(k)$
$W_N{}^{\ln}x(n)$	$X(k+l)$
$x_1(n)_N(*)_N x_2(n)_N$	$X_1(k)_N X_2(k)_N$
$x_1(n)_N x_2(n)_N$	$\frac{1}{N}\sum_{l=0}^{N-1} X_1(l)_N X_2(k-l)_N$ $X_1(k)_N(*)_N X_2(k)_N$
$x(n)$thực	$X(k) = X^*(-k)$
	$\Re[X(k)] = \Re[X(-k)]$
	$\Im[X(k)] = \Im[X(-k)]$
	$\lvert X(k)\rvert = \lvert X(-k)\rvert$
	$\arg[X(k)] = \arg[X(-k)]$

2.1.3.2. Biến đổi Fourier nhanh:

Biến đổi Fourier nhanh - FFT (Fast Fourier Transform) là thuật toán rất hiệu quả để tính DFT của một chuỗi số. Ưu điểm là ở chỗ nhiều tính toán được lặp lại do tính tuần hoàn của số hạng Fourier $e^{-j\frac{2\pi}{N}kn}$. Dạng của DFT là:

$$X(k) = \sum_{n=0}^{N-1} x(n).{W_N}^{kn} \tag{2.28}$$

Ta có $W^{(N+qN)(k+rN)} = W^{nk}$ với mọi q, r nguyên do tính tuần hoàn của số hạng Fourier. Tách DFT thành 2 phần:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2\mathrm{n}){W_N}^{2\mathrm{nk}} + \sum_{n=0}^{\frac{N}{2}-1} x(2\mathrm{n}+1){W_N}^{(2\mathrm{n}+1)k} \tag{2.29}$$

Chỉ số dưới N của số hạng Fourier biểu diễn kích thước của chuỗi. Nếu chúng ta biểu diễn thành phần chẵn của chuỗi số x(n) bằng x_{ev} và thành phần lẻ là x_{od} thì phương trình có thể viết lại:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{ev}{W_{\frac{N}{2}}}^{nk} + {W_{\frac{N}{2}}}^{k} \sum_{n=0}^{\frac{N}{2}-1} x_{od}{W_{\frac{N}{2}}}^{nk} \tag{2.30}$$

Ta có hai biểu thức DFT, do đó có thể viết :

$$X(k) = X_{ev}(k) + {W_{\frac{N}{2}}}^{k} X_{od}(k) \tag{2.31}$$

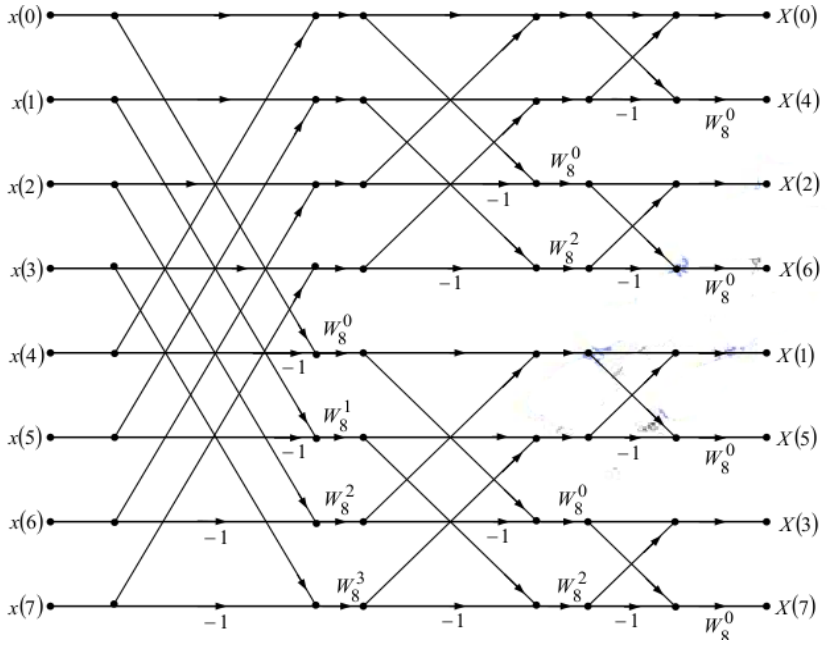
Chỉ số k chạy đến N-1 nhưng do sử dụng tính chu kỳ của hàm chẵn và hàm lẻ nên chỉ cần tính DFT N/2 điểm để có được giá trị của X(k).

$$X_{ev}(k) = X_{ev}\left(k - \frac{N}{2}\right) \frac{N}{2} \leq k < N - 1 \tag{2.32}$$

Tiếp tục chia DFT kết quả thành hai nửa chẵn và lẻ cho đến khi chỉ còn phải tính hai điểm DFT.

$$A(k) = \begin{cases} \lambda(0) + \lambda(1)e^{-j\frac{2\pi}{N}k} & \forall k \\ \lambda(0) + \lambda(1) & k\ even \\ \lambda(0) - \lambda(1) & kodd \end{cases} \tag{2.33}$$

Đối với 2 điểm DFT này chỉ cần phép cộng và trừ mà không cần phép nhân. Để tính toàn bộ DFT, chúng ta nhân 2 điểm DFT với các thừa số W thích hợp từ W^0 tới $W^{N/2}$. Hình dưới1à đồ thị 8 điểm FFT.



Hình 2.10. FFT 8 điểm, cơ số 2, phân chia theo tần số

Chúng ta có thể so sánh trực tiếp DFT và FFT như sau:

Khi tính trực tiếp DFT, mỗi giá trị của k cần N phép nhân phức và N-1 phép cộng phức. Đối với FFT, mỗi hàm đều có dạng $\lambda(0) \pm W^p \lambda(1)$ (gọi là số bướm do đồ thị có hình cánh bướm) yêu cầu một phép nhân và hai phép cộng. Từ đồ thị trên hình 2.5 chúng ta có thể tổng quát hóa số bướm là:

Số bướm $= \frac{N}{2}\log_2 N$

Điều này là do có N/2 hàng bướm (bởi vì mỗi bướm có hai ngõ vào) và $\log_2$N cột bướm.

2.1.3.3. Biến đổi Cosine rời rạc:

Biến đổi Cosine rời rạc DCT *(Discrete Cosine Transform)* được sử dụng rộng rãi trong xử lý tiếng nói. Nó là một phép biến đổi chuyển tín hiệu sang miền tần số.

Phép biến đổi thuận:

$$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n).\cos\left[\frac{\pi(2n+1)k}{2N}\right], k = 0,1,2,\ldots,N-1 \qquad (2.34)$$

Phép biến đổi nghịch:

$$x(n) = \alpha(n) \sum_{k=0}^{N-1} X(k).\cos\left[\frac{\pi(2n+1)k}{2N}\right], n = 0,1,2,\ldots,N-1 \qquad (2.35)$$

2.1.4. Các bộ lọc số và các cửa sổ:

Bộ lọc số là một hệ thống số dùng để làm biến dạng sự phân bố tần số của các thành phần của một tín hiệu theo các chỉ tiêu đã cho.

Lọc số là các thao tác của xử lý dùng để làm biến dạng sự phân bố tần số của các thành phần của một tín hiệu theo các chỉ tiêu đã cho nhờ một hệ thống số.

Phần này mô tả các nguyên tắc cơ bản của thiết kế bộ lọc số, nghiên cứu bộ lọc số có đáp ứng xung chiều dài hữu hạn FIR (Finite-Impulse Response) và bộ lọc số có đáp ứng xung chiều dài vô hạn IIR (Infinite-Impulse Response), là các loại đặc biệt của các bộ lọc số pha tuyến tính.

2.1.4.1. Bộ lọc lý tưởng thông thấp:

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| < \omega_0 \\ 0 & \omega_0 < |\omega| < \pi \end{cases} \quad (2.36)$$

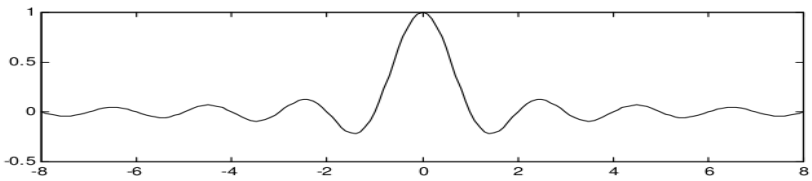
Sử dụng định nghĩa của biến đổi Fourier, ta được:

$$h(n) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega} e^{j\omega n} d\omega = \frac{(e^{j\omega_0 n} - e^{-j\omega_0 n})}{2\pi j n} = \frac{\sin\omega_0 n}{\pi n} = \left(\frac{\omega_0}{\pi}\right) sinc(\omega_{0\mathrm{n}}) \quad (2.37)$$

Ở đây hàm sinc được định nghĩa như sau:

$$sinc(x) = \frac{\sin\pi x}{\pi x} \quad (2.38)$$

có phần thực là hàm chẵn của x được mô tả trong hình dưới.



Hình 2.11. Hàm sinc

2.1.4.2. Các phương pháp cửa sổ:

Các phương pháp cửa sổ là các tín hiệu được tập trung trong một khoảng thời gian giới hạn. Có các phương pháp cửa sổ tam giác như Kaiser, Barlett,... cửa sổ chữ nhật Hanning và Hamming, trong đó Hanning và Hamming được sử dụng rộng rãi trong xử lý tiếng nói.

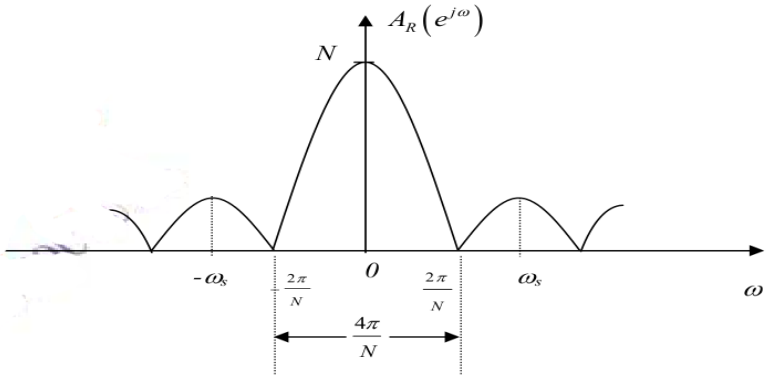
Cửa sổ chữ nhật: Trong miền n, cửa sổ chữ nhật được định nghĩa như sau:

$$w_R(n)_N = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & n \ne \end{cases} \quad (2.39)$$

Xét cửa sổ chữ nhật trong miền tần số ta có:

$$W_R(e^{j\omega})_N = \sum_{n=0}^{N-1} e^{-j\omega n} = \frac{1-e^{-j\omega N}}{1-e^{-j\omega}} = \frac{e^{-j\omega\frac{N}{2}}\left(e^{j\omega\frac{N}{2}}-e^{-j\omega\frac{N}{2}}\right)}{e^{-j\frac{\omega}{2}}\left(e^{j\frac{\omega}{2}}-e^{-j\frac{\omega}{2}}\right)} =$$

$$e^{-j\omega\frac{N-1}{2}}\frac{\sin\omega\frac{N}{2}}{\sin\frac{\omega}{2}} = e^{-j\omega\frac{N-1}{2}}A_R(e^{j\omega}) \qquad (2.40)$$



Hình 2.12. Biểu diễn $A_R(e^{j\omega})$

Có hai tham số đánh giá cửa sổ là:

- Bề rộng đỉnh trung tâm Δω .
- Tỷ số giữa biên độ đỉnh thứ cấp thứ nhất trên biên độ đỉnh trung tâm:

$$\lambda = 20\lg\frac{|W(e^{j\omega_s})|}{|W(e^{j0})|}$$

Cửa sổ Hanning và Hamming: Trong miền n, cửa sổ Hanning và Hamming được định nghĩa như sau:

$$w_H(n)_N = \begin{cases} \alpha - (1-\alpha)\cos\frac{2\pi}{N-1}n & 0 \le n \le N-1 \\ 0 & n \ne \end{cases} \qquad (2.41)$$

Phân loại khác nhau theo hệ số α ta được:

+ α = 0,5 : cửa sổ Hanning

$$w_H(n)_N = \begin{cases} 0{,}5 - 0{,}5\cos\frac{2\pi}{N-1}n & 0 \le n \le N-1 \\ 0 & n \ne \end{cases} \qquad (2.42)$$

+ α = 0,54 : cửa sổ Hamming

$$w_H(n)_N = \begin{cases} 0{,}54 - 0{,}46\cos\frac{2\pi}{N-1}n & 0 \le n \le N-1 \\ 0 & n \ne \end{cases} \qquad (2.43)$$

Ta có các tham số của bộ lọc Hanning:

+ $\Delta\omega_{Han} = 8\pi/N$

+ $\lambda_{Han} \approx -32dB$

Các tham số của bộ lọc Hamming:

+ $\Delta\omega_{Ham} = 8\pi /N$

+ $\lambda_{Ham} \approx -43dB$

Như vậy ta thấy: $\Delta\omega_T = \Delta\omega_{Han} = \Delta\omega_{Ham} = 8\pi /N$; $\lambda_T > \lambda_{Han} > \lambda_{Ham}$ vậy trong 3 cửa sổ bề rộng đỉnh trung tâm là như nhau nhưng biên độ của độ gợn sóng dải thông và dải chắn sẽ nhỏ nhất khi thiết kế bằng cửa sổ Hamming.

2.1.4.3. Bộ lọc FIR và IIR:

Các hệ thống có đặc tính xung có chiều dài hữu hạn được gọi là FIR:

$$h(n) = \begin{cases} \neq 0 & N_1 \leq n \leq N_2 \\ = 0 & -\infty < n < N_1 \vee N_2 < n < \infty \end{cases} \quad (2.44)$$

Giả sử hệ thống FIR:

$$h(n) = \begin{cases} \frac{b_k}{a} & 0 \leq k \leq M \\ 0 & k \neq \end{cases} \quad (2.45)$$

Khi đó đặc tính xung của hệ thống:

$$y(n) = \frac{1}{a_0} \sum_{k=0}^{M} b_k x(n-k) \quad (2.46)$$

Các hệ thống có đặc tính xung có chiều dài vô hạn được gọi là IIR:

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \quad (2.47)$$

Phương trình trên là một phương trình đệ qui:

$$y(n) = -\sum_{k=0}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \quad (2.48)$$

vì vậy IIR còn gọi là lọc đệ qui và FIR là lọc không đệ qui.

Khi đó hệ thống có hàm truyền trong mặt phẳng Z:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} \quad (2.49)$$

2.1.5. Xác suất và quá trình ngẫu nhiên:

Tín hiệu tiếng nói là một quá trình thống kê ở đó sự tương quan của chuỗi số đóng vai trò quan trọng để phân tích, dự báo đặc tính của tiếng nói. Ngoài ra, trong nhận dạng tiếng nói, mô hình xác suất là một trong những mô hình đạt kết quả

tốt nhất. Phần này trình bày khái niệm xác suất và quá trình ngẫu nhiên làm cơ sở cho trích huấn luyện và nhận dạng tiếng nói ở chương sau.

2.1.5.1. Cơ sở xác suất:

Xác suất của sự kiện A được ký hiệu là P(A). Có 3 tính chất của P(A):

- $P(A) \geq 0$

- P(tất cả khả năng có thể xảy ra) = 1

- Cho {Ai}, với (Ai . Aj = 0) thì P(Ai.Aj) = P(Ai) + P(Aj)

Chúng ta định nghĩa thêm xác suất đồng thời và xác suất có điều kiện. Xác suất đồng thời là xác suất để hai hay nhiều sự kiện đồng thời xảy ra trong một phép thử. Kí hiệu xác suất 2 sự kiện A và B xảy ra đồng thời: P(AB)

Xác suất có điều kiện là xác suất để xảy ra sự kiện A trong khi sự kiện B đã xảy ra. Kí hiệu: P(A/B). Công thức tính:

$$P(A \mid B) = \frac{P(AB)}{P(B)} \tag{2.50}$$

Công thức Bayes: 2 biến cố A, B độc lập:

$$P(A \mid B) = \frac{P(B|A).P(A)}{P(B)} \tag{2.51}$$

2.1.5.2. Biến ngẫu nhiên:

Trong xử lý tín hiệu, điều mong muốn là tín hiệu có giá trị hay nằm trong một phạm vi cụ thể. Tín hiệu trong trường hợp này được coi là biến ngẫu nhiên.

Đây là tập sự kiện của biến rời rạc có giá trị cụ thể, ngoài ra còn có tập sự kiện của biến liên tục có giá trị nằm trong một phạm vi nào đó. Chúng ta có thể định nghĩa hàm phân phối xác suất của một biến ngẫu nhiên như sau:

$$F(x) = P(X \leq x)$$

Hàm phân phối xác suất là hàm tăng của biến độc lập x và chỉ đúng cho biến ngẫu nhiên cụ thể X.

Nếu lấy vi phân F(x) theo biến x, ta nhận được hàm mật độ xác suất PDF *(Probablity Destiny Function)* của X:

$$p(x) = \frac{dF(x)}{dx} \tag{2.52}$$

Lấy tích phân p(x), ta có được hàm phân phối xác suất như sau:

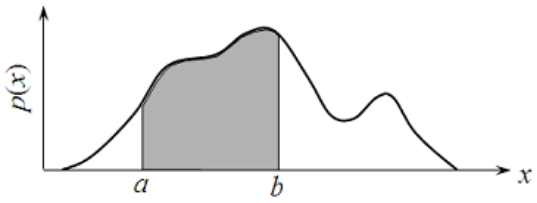
$$F(x) = \int_{-\infty}^{x} p(\lambda)d\lambda \quad (2.53)$$

Ta có thể xác định được xác suất của biến ngẫu nhiên X nằm giữa a và b:

$$P(X \leq b) = P(a < X \leq b) + P(X \leq a)$$

Viết lại phương trình trên theo hàm phân phối:

$$P(a < X \leq b) = F(b) - F(a) = \int_{a}^{b} p(x)dx \quad (2.54)$$



Hình 2.13. Hàm phân phối

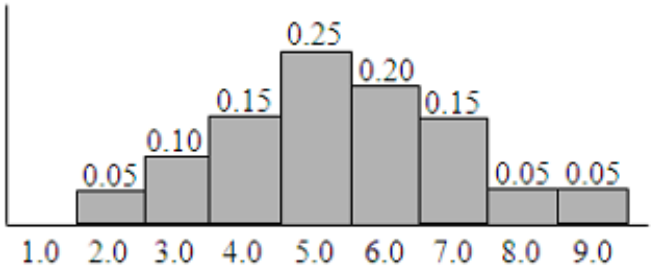
Nghĩa là nếu biết hàm phân phối hay hàm mật độ, chúng ta có thể tính được xác suất của biến ngẫu nhiên X nằm trong phạm vi cho trước.

2.1.5.3. Kỳ vọng, phương sai:

Giá trị kỳ vọng của x ký hiệu E(x) là giá trị có khả năng xảy ra nhiều nhất. E(x) còn được gọi là giá trị trung bình, và được tính từ hàm mật độ như sau:

$$E(x) = \int_{-\infty}^{\infty} xp(x)dx \quad (2.55)$$

Ví dụ:



E(X) = 2·0.05+3·0.10+ … +9·0.05 = 5.35

Phương sai của biến ngẫu nhiên x được định nghĩa:

$$\sigma^2 = \text{Var}(x) = E[(x - E[x])^2] \quad (2.56)$$

σ là căn bậc hai giá trị bình phương trung bình của độ lệch giữa một biến và giá trị trung bình của biến đó.

2.2. BIỂU DIỄN TÍN HIỆU TIẾNG NÓI

Sử dụng pháp biểu diễn tín hiệu tiếng nói trên ảnh phổ (spectrogram). Ảnh phổ rất hữu dụng để phân tích các âm vị và sự chuyển trạng thái của chúng. Một ảnh phổ của một tín hiệu thời gian là một biểu diễn hai chiều đặc biệt, hiển thị thời gian trên trục ngang và tần số trên trục dọc. Một thang màu xám thường được dùng để chỉ mức năng lượng tại mỗi điểm (t, f) với màu trắng chỉ mức năng lượng thấp và màu đen là mức năng lượng cao. Trong phần này sẽ tìm hiểu phương pháp phân tích Fourier thời gian ngắn, công cụ cơ bản để tính toán chúng.

2.2.1. Biến đổi Fourier thời gian ngắn:

Phép biến đổi Fourier không thể áp dụng đối với tín hiệu không dừng, vì các thành phần tần số không ổn định. Tuy nhiên nếu chúng ta chia tín hiệu không dừng thành những đoạn đủ nhỏ theo thời gian thì tín hiệu trong mỗi đoạn có thể xem là tín hiệu dừng và do đó có thể lấy biến đổi Fourier trên từng đoạn tín hiệu này. Như vậy, phép biến đổi Fourier thời gian ngắn STFT *(Short-Time Fourier Transform)* vừa có tính định vị theo tần số do tính chất của biến đổi Fourier, vừa có tính định vị theo thời gian do được tính trong từng khoảng thời gian ngắn. Đây là nguyên lý của STFT hay còn gọi là biến đổi Fourier cửa sổ hóa.

Trong STFT, tín hiệu f(t) đầu tiên được nhân với một hàm cửa sổ w(t-τ) để lấy được tín hiệu trong khoảng thời gian ngắn xung quanh thời điểm τ. Sau đó phép biến đổi Fourier bình thường được tính trên đoạn tín hiệu này. Kết quả chúng ta được một hàm hai biến $STFT_f(w,t)$ xác định bởi:

$$STFT_f(w,t) = \int_{-\infty}^{\infty} f(t).w*(t-\tau)e^{-jwt}dt \qquad (2.57)$$

STFT tại thời điểm τ là biến đổi Fourier của tín hiệu f(t) nhân với phiên bản dịch một khoảng τ theo thời gian w(t-τ) của cửa sổ cơ bản tập trung xung quanh τ. STFT có tính định vị theo thời gian. Cửa sổ càng hẹp thì tính định vị càng tốt.

Để thấy rõ hơn về tính định vị theo tần số, ta áp dụng định lý Parserval để viết lại (2.57) như sau:

$$STFT_f(w,t) = \int_{-\infty}^{\infty}(w(t-\tau)e^{jwt})*f(t)dt$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} F[(w(t-\tau)e^{jwt})^*].F[f(t)]dw$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} [W * (w' - w). e^{j(w\prime - w)\tau}] . F[w'] dw'$$

$$= \frac{e^{-jwt}}{2\pi} \int_{-\infty}^{\infty} W * (w' - w) F(w') e^{jw\prime t} dw' \qquad (2.58)$$

với *W*(w'-w)* và *F(w')* lần lượt là phổ của cửa sổ *w(t-τ)* và tín hiệu *f(t)*.

W(w'-w)* có tác dụng như một bộ lọc dải thông tập trung quanh tần số w có băng thông bằng băng thông *w(t)* làm giới hạn phổ của tín hiệu *F(w')* xung quanh tần số đang phân tích *w*. Như vậy STFT có tính định vị theo tần số. Tính định vị này càng tốt khi băng thông của cửa sổ phân tích càng hẹp.

Ta thấy rằng, STFT chính là số đo độ giống nhau giữa tín hiệu phiên bản dịch và biến điệu của cửa sổ cơ bản vì (2.57) có thể viết lại như sau:

$$STFT_f(w,t) = \int_{-\infty}^{\infty} (w(t-\tau) e^{jwt}) * f(t) dt = \langle g_{w,\tau}(t), f(t) \rangle \qquad (2.59)$$

với $g_{w,\tau}(t) = w(t - \tau)e^{jwt}$ là phiên bản dịch và biến thiên của *w(t)*.

Do việc dịch thời gian một khoảng τ làm cho cửa sổ tịnh tiến một khoảng τ theo trục thời gian và biến điệu cửa sổ với e^{jwt} là cửa sổ tịnh tiến một khoảng w theo trục tần số, nên kích thước của cửa sổ không thay đổi mà chỉ dời đến vị trí mới xung quanh *(τ, w)*. Như vậy, mỗi hàm cửa sổ cơ sở sử dụng trong phép biến đổi này đều có một độ phân giải thời gian - tần số, chỉ khác vị trí trên mặt phẳng thời gian tần số. Do đó, có thể rời rạc hóa dễ dàng STFT trên một lưới chữ nhật $(mw_0, n\tau_0)$.

Nếu hàm cửa sổ là một bộ lọc hạ thông có tần số cắt w_b, hoặc băng thông $2w_b$ thì w_0 được chọn nhỏ hơn w_b và τ_0 nhỏ hơn π/w_0 để việc lấy mẫu không mất thông tin. Các hàm cửa sổ tại tất cả các điểm lấy mẫu sẽ phủ kín mặt phẳng thời gian - tần số của phép biến đổi.

Độ phân giải thời gian - tần số của STFT phụ thuộc vào hàm cửa sổ. Để có độ phân giải tốt thì cửa sổ phân tích phải hẹp (về mặt thời gian). Trong khi đó, để đạt được độ phân giải tần số tốt thì băng thông của cửa sổ phải hẹp. Tuy nhiên, theo nguyên lý bất định thì không thể tồn tại một cửa sổ với khoảng thời gian và băng thông hẹp tùy ý mà có một sự hoán đổi giữa hai thông số này (do tích của chúng bị chặn dưới). Nếu ta chọn cửa sổ có băng thông hẹp để độ phân giải tốt thì khoảng thời gian lại rộng làm cho độ phân giải thời gian lại kém đi và ngược lại, đây chính là nhược điểm của STFT.

2.2.2. Phân tích Fourier thời gian ngắn:

Ý tưởng đằng sau ảnh phổ là tính toán một biến đổi Fourier mỗi 5ms một lần, hay biểu diễn năng lượng tại mỗi điểm thời gian/tần số. Do một vài miền tín hiệu tiếng nói ngắn hơn khoảng 100ms thường xuất hiện định kỳ, ta có sử dụng các kỹ thuật đã đề cập ở phần xử lý tín hiệu số. Tuy nhiên, tín hiệu không còn tuần hoàn khi phân tích các đoạn dài hơn, do đó, việc xác định chính xác của biến đổi Fourier không thể dùng được nữa. Hơn nữa, việc xác định này yêu cầu kiến thức của tín hiệu thời gian vô hạn. Vì hai lý do này, các kỹ thuật mới gọi là phân tích thời gian ngắn (short-time analysis) được đề xuất. Các kỹ thuật này phân tích tín hiệu tiếng nói thành một chuỗi các đoạn ngắn, gọi là các khung (frame) và phân tích mỗi khung này một cách độc lập.

Cho $x_m(n)$ là tín hiệu thời gian ngắn của khung m.

$w_m(n)$ là hàm cửa sổ, bằng 0 tại mọi điểm trừ một vùng nhỏ.

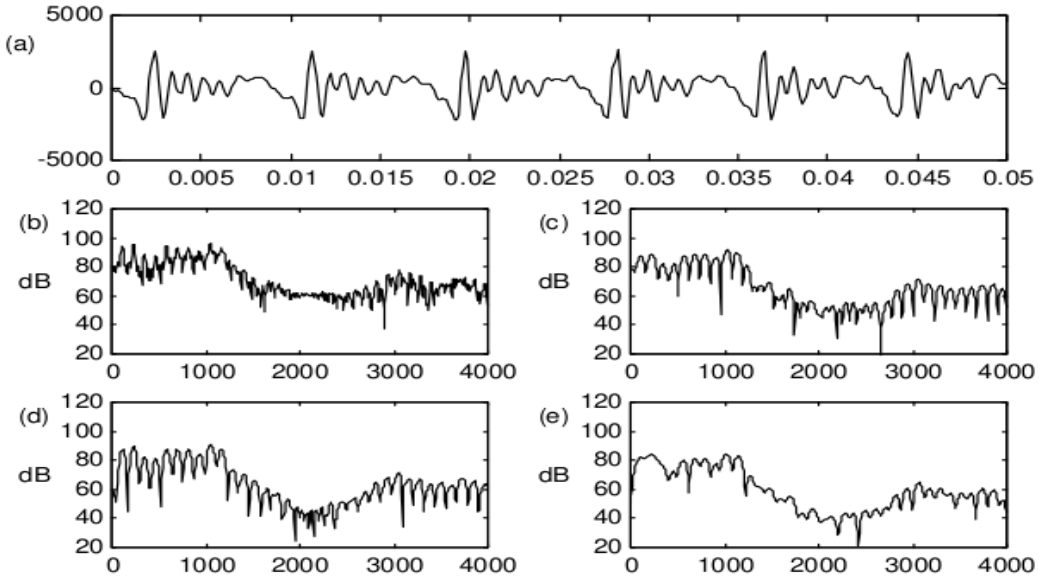
Có $x_m(n) = x(n)w_m(n)$

Do hàm cửa sổ có thể có các giá trị khác nhau đối với mỗi frame m, để giữ giá trị không đổi cho tất cả frame thì:

$w_m(n) = w(m-n)$

Biểu diễn Fourier thời gian ngắn đối với frame m được định nghĩa:

$$X_m(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_m(n)e^{-j\omega n} = \sum_{n=-\infty}^{\infty} w(m-n)x(n)e^{-j\omega n} \quad (2.60)$$

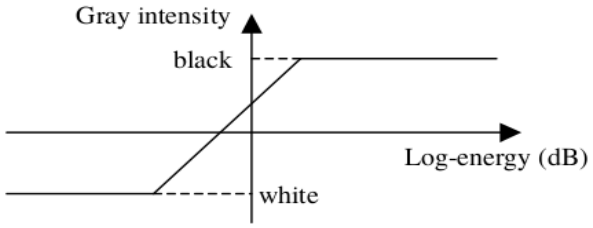


Hình 2.14. Phổ thời gian ngắn của tiếng nói giọng nam

Do ảnh phổ chỉ hiển thị năng lượng và không phải đoạn giới hạn của biến đổi Fourier nên mức năng lượng được tính như sau:

$$\log|X(k)|^2 = \log({X_r}^2(k) + {X_i}^2(k)) \quad (2.61)$$

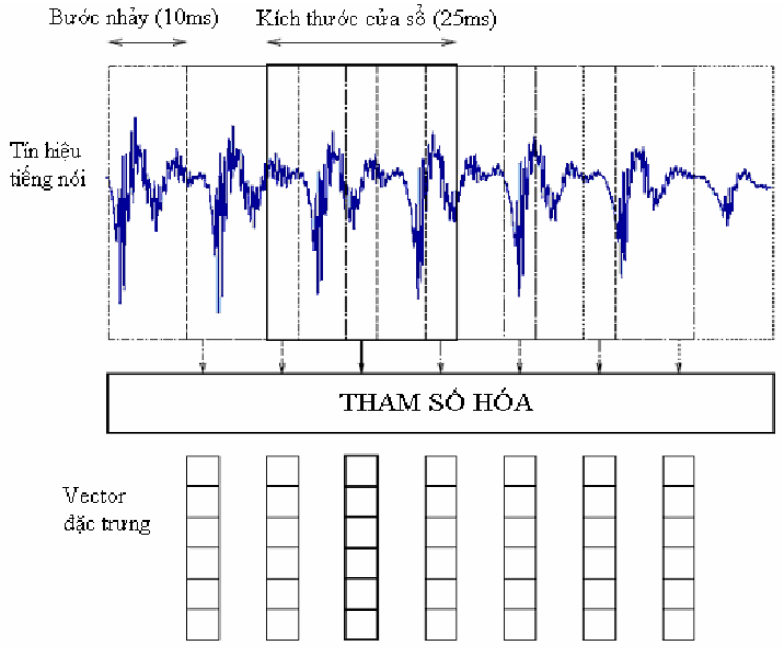
Giá trị này được chuyển sang thang xám như hình (2.16). Các pixel mà giá trị không được tính toán được thêm vào. Đoạn nghiêng điều chỉnh độ tương phản của ảnh phổ, trong khi các điểm bão hòa mà trắng và màu đen điều chỉnh dãy động học.



Hình 2.15. Chuyển đổi giữa giá trị năng lượng log (trên trục x) sang thang xám (trục y)

2.3. RÚT TRÍCH ĐẶC TRƯNG TIẾNG NÓI

Quá trình nhận dạng mẫu (cả ở pha huấn luyện hay pha nhận dạng) đều trải qua giai đoạn trích chọn đặc trưng (feature extraction). Bước này thực hiện các phân tích phổ (spectral analysis) nhằm xác định các thông tin quan trọng, đặc trưng, ổn định của tín hiệu tiếng nói, tối thiểu hóa ảnh hưởng của nhiễu; xúc cảm, trạng thái, cách phát âm của người nói; giảm khối lượng dữ liệu cần xử lý...

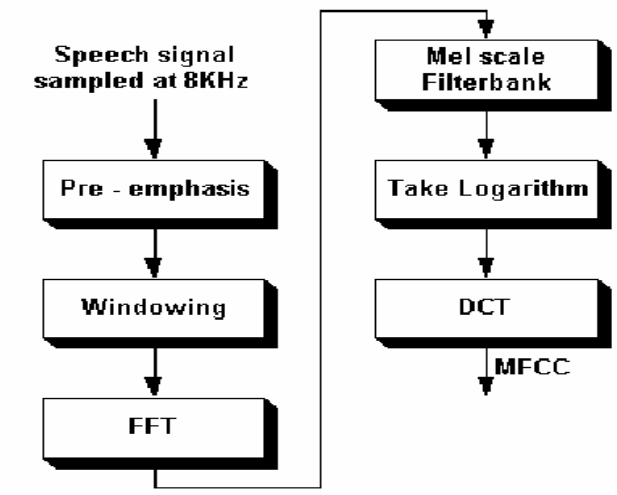


Hình 2.16. Sơ đồ rút trích đặc trưng tổng quát

2.3.1. Trích đặc trưng MFCC (Mel-scale Frequency Cepstral Coefficient)

MFCC là phương pháp trích đặc trưng dựa trên đặc điểm cảm thụ tần số âm của tai người: tuyến tính đối với tần số nhỏ hơn 1kHz và phi tuyến đối với tần số trên 1kHz (theo thang tần số Mel, không phải theo Hz)

Đối với phương pháp MFCC, việc tính đặc trưng có sơ đồ như sau:



Hình 2.17. Các bước tính đặc trưng MFCC

2.3.1.1. Tiền nhấn (Pre-emphasis):

Chúng ta biết rằng phổ tiếng nói hữu thanh có khuynh hướng suy giảm toàn bộ -6 dB/octave khi tần số tăng lên. Điều này là do khuynh hướng suy giảm -12 dB/octave của nguồn kích âm hữu thanh và tăng lên +6 dB/octave do phát âm miệng. Do đó cần phải bù +6 dB/octave trên toàn bộ băng tần. Điều này được gọi là pre-emphasis tín hiệu. Trong xử lý tín hiệu số, chúng ta dùng bộ lọc thông cao có tần số cắt 3 dB ở tần số trong phạm vi từ 100 Hz đến 1k Hz. Phương trình sai phân:

$$y(n) = x(n) - a*x(n) \tag{2.62}$$

Trong đó y(n) là mẫu ra hiện tại của bộ lọc pre-emphasis, x(n) là mẫu vào hiện tại, x(n-1) là mẫu vào trước đó và a là hằng số thường được chọn giữa 0.9 và 1. Lấy biến z của phương trình trên:

$$Y(z)=X(z) - az^{-1}X(z)=(1 - az^{-1})X(z) \tag{2.63}$$

Trong đó z^{-1} là toán tử trễ mẫu đơn vị. Suy ra hàm truyền H(z) của bộ lọc:

$$H(z) = \frac{Y(z)}{X(z)} = 1 - az^{-1} \tag{2.64}$$

2.3.1.2. Cửa sổ hóa (Windowing):

Đầu tiên tín hiệu tiếng nói x(n) sẽ được chia thành từng frame (có thực hiện chồng phủ một phần lên nhau - overlap) để được T frame $x'_t(n)$. Công việc cửa sổ hoá này sẽ được thực hiện bằng cách nhân tín hiệu tiếng nói với một hàm cửa sổ. Gọi phương trình cửa sổ hóa là w(n) ($0 \leq n \leq N-1$; N: số mẫu trong 1 frame tín hiệu), khi đó tín hiệu sau khi được cửa sổ hóa là $X_t(n)$:

$$X_t(n) = x'_t(n).w(n)$$

Hàm cửa sổ thường được dựng là hàm cửa sổ Hamming:

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N}); n = 0..N-1 \quad (2.65)$$

2.3.1.3. Biến đổi Fourier nhanh (Fast Fourier Transform - FFT):

Phổ tín hiệu sau khi nhân với cửa sổ Hamming sẽ sử dụng phép biến đổi Fourier nhanh Ta thu được biên độ phổ chứa các thông tin có ích của tín hiệu tiếng nói. Biến đổi Fourier nhanh - FFT (Fast Fourier Transform) là thuật toán rất hiệu quả để tính DFT của một chuỗi số. Ưu điểm là ở chỗ nhiều tính toán được lặp lại do tính tuần hoàn của số hạng Fourier $e^{-j\frac{2\pi}{N}kn}$. Dạng của DFT là:

$$X(k) = \sum_{n=0}^{N-1} x(n).e^{-j\frac{2\pi}{N}kn} \quad (2.66)$$

2.3.1.4. Lọc qua bộ lọc Mel-scale :

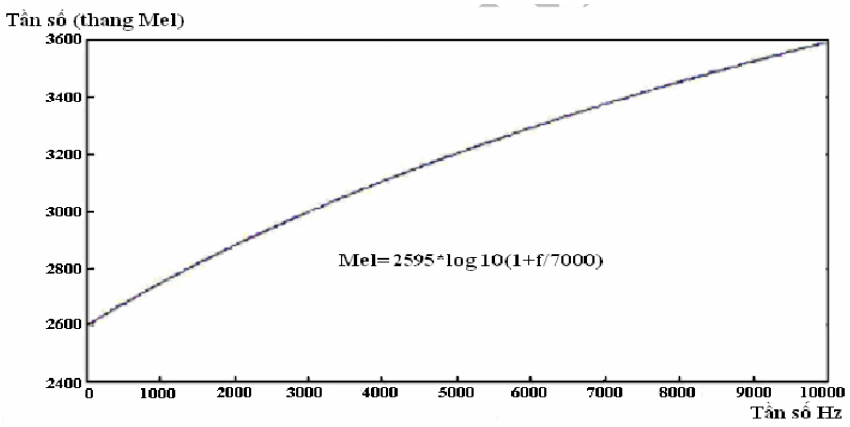
Các nghiên cứu về hệ thống thính giác của con người cho thấy, tai người có cảm nhận đối với độ lớn các tần số không theo thang tuyến tính. Các đặc trưng phổ tần số của tiếng nói được tai người tiếp nhận như ngõ ra của một dãy các bộ lọc. Tần số trung tâm của các bộ lọc này không phân bố tuyến tính dọc theo trục tần số. Thành phần phổ dưới 1 kHz thường được tập trung nhiều bộ lọc hơn vì nó chứa nhiều thông tin về âm thanh hơn. Ở tần số thấp các bộ lọc băng hẹp được sử dụng để tăng độ phân giải tần số để có được tần số cơ bản và họa tần vốn ổn định, còn ở tần số cao các bộ lọc thông băng rộng được sử dụng để thu được các thành phần tần số cao vốn biến động rất nhanh.

Với nỗ lực nhằm mô tả chính xác sự tiếp nhận tần số của tai người, một thang tần số được xây dựng - thang tần số Mel dựa trên cơ sở thực nghiệm cảm nhận nghe của người. Tần số 1 kHz được chọn là 1000 Mel. Mối quan hệ giữa thang

tần số thực (vật lý) và thang tần số Mel (sinh lý) được cho bởi công thức:

$$F_{Mel} = 2595\log_{10}(1 + \frac{F_{Hz}}{700}) \quad (2.67)$$

với F_{Mel} là tần số sinh lý, đơn vị Mel; F_{Hz} là đơn vị tần số thực, đơn vị Hz.



Hình 2.18. Đồ thị biểu diễn mối quan hệ giữa Mel và Hz

Trên hình 2.18 cho thấy, với những tần số nhỏ hơn 1 kHz, thì quan hệ giữa thang mel và tần số thực là gần tuyến tính. Còn các tần số trên 1 kHz thì quan hệ này là logarithm. Như vậy thay vì xây dựng các bộ lọc trên thang tần số thực ta có thể xây dựng các bộ lọc này với tần số trung tâm cách đều tuyến tính trên thang Mel.

Tần số trung tâm của bộ lọc thứ m được xác định bởi:

$$f_m = f_{m-1} + \Delta f_m \quad (2.68)$$

Trong đó: f_m là tần số trung tâm của bộ lọc thứ m

f_{m-1} là tần số trung tâm của bộ lọc thứ m -1

Δf_m là băng thông của bộ lọc thứ m

Δf_m được xác định: Với khoảng tần số dưới 1 kHz, thì fm được chọn sao cho có khoảng 10 bộ lọc phân bố cách đều trong khoảng này. Với khoảng tần số trên 1kHz, f_m thường được tính bởi : $f_m = 1.2 * f_m - 1$.

Kết quả sau khi cho phổ tín hiệu $X_t(k)$ qua bộ lọc ta thu được $Y_t(m)$.

2.3.1.5. Tính log năng lượng phổ:

Sau khi qua bộ lọc Mel, phổ tín hiệu $Y_t(m)$ sẽ được tính Log_{10} theo:

$$log\{|Y_t(m)|^2\} \quad (2.69)$$

2.3.1.6. Biến đổi Cosine rời rạc:

Bước cuối cùng để thu được các hệ số MFCC là lấy biến đổi Cosine rời rạc của kết quả cho bởi (2.65):

$$y_t{}^{(m)}(k) = \sum_{m=1}^{M} \log\{|Y_t(m)|^2\}\cos(k(m-\frac{1}{2})\frac{\pi}{M}) \quad (2.70)$$

Thông thường số điểm rời rạc k của biến đổi ngược này được chọn $1 \leq k \leq 12$. Các hệ số MFCC chính là số điểm rời rạc này, ta có thể có 1-12 hệ số MFCC.

2.3.2. Phương pháp mã hóa dự báo tuyến tính LPC (Linear Predictive Coding)

Ý tưởng cơ bản của phương pháp mã hóa dự báo tuyến tính (LPC) là tại thời điểm n, mẫu tiếng nói s(n) có thể được xấp xỉ bởi một tổ hợp tuyến tính của p mẫu trước đó:

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \quad (2.71)$$

Trong đó giả sử a_1, a_2, ... , a_p là hằng số trên khung dữ liệu (frame) được phân tích. Chúng ta chuyển quan hệ trên thành dạng đẳng thức bằng cách thêm vào số hạng Gu(n) gọi là nguồn kích thích:

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n) \quad (2.72)$$

Trong đó u(n) là nguồn kích thích được chuyển hóa và G gọi là độ lợi của nó. Thực hiện biến đổi z ở hai vế của phương trình trên, ta có:

$$S(z) = \sum_{i=0}^{p} a_i z^{-i} S(z) + GU(z) \quad (2.73)$$

dẫn đến hàm truyền là:

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1-\sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)} \quad (2.74)$$

Ký hiệu š (n) là dự báo tuyến tính của s(n):

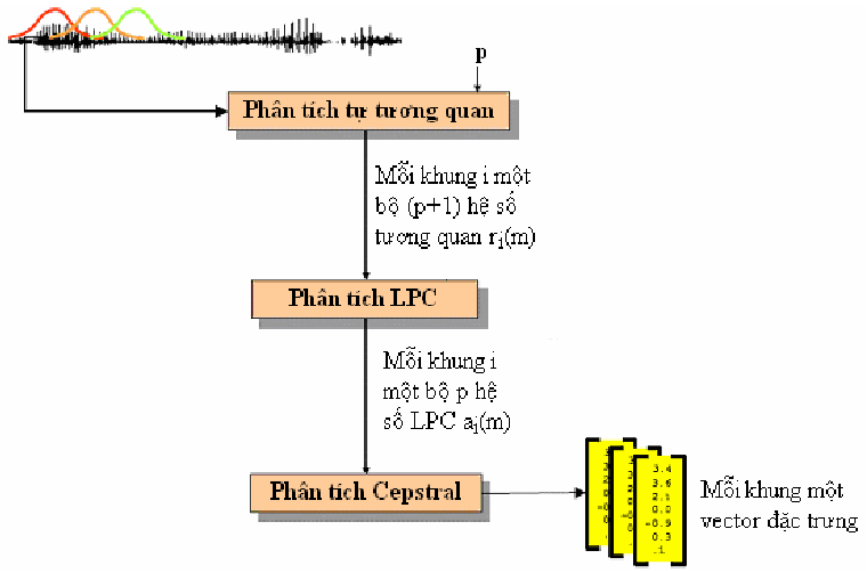
$$\tilde{s} = \sum_{k=1}^{p} a_k s(n-k)$$

Khi đó thiết lập lỗi dự báo e(n) được định nghĩa là:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) = G.u(n) \quad (2.75)$$

Để tìm tập các hệ số a_k, k = 1, 2, ..., p trên khung được phân tích, cách tiếp cận cơ bản là ta cực tiểu hóa sai số bình phương trung bình. Khi đó sẽ dẫn đến việc

ta phải giải một hệ phương trình với p ẩn số. Có nhiều phương pháp để giải hệ phương trình đó, nhưng trong thực tế, phương pháp thường được dùng là phương pháp phân tích tự tương quan.



Hình 2.19. Sơ đồ bộ xử lý LPC rút trích đặc trưng tiếng nói

2.3.2.1. Phân tích tự tương quan:

Mỗi khung sau khi được lấy cửa sổ sẽ được đưa qua bước phân tích tự tương quan và cho ra (p + 1) hệ số tự tương quan:

$$r_i(m) = \sum_{n=0}^{N-m-1} \tilde{x}_i(n)\tilde{x}_i(n+m)\,; m = 0,1,\ldots,p \qquad (2.76)$$

Trong đó giá trị tự tương quan cao nhất, p, được gọi là cấp của phân tích LPC. Thông thường, ta sử dụng các giá trị p trong khoảng từ 8 đến 16.

2.3.2.2. Phân tích LPC:

Bước này, ta sẽ chuyển mỗi khung gồm (p + 1) hệ số tự tương quan thành p hệ số LPC bằng cách dùng thuật toán Levinson - Durbin.

Thuật toán Levinson - Durbin thể hiện qua mã giả sau:

Dữ liệu vào là (p + 1) hệ số tự tương quan chứa trong r; kết quả ra là p hệ số LPC chứa trong a.

Lúc này, ta có thể dùng các hệ số LPC làm vector đặc trưng cho từng khung. Tuy nhiên, có một phép biến đổi tạo ra dạng hệ số khác có độ tập trung cao hơn từ các hệ số LPC, đó là phép phân tích Cepstral.

2.3.2.3. Phân tích cepstral:

Từ p hệ số LPC ở mỗi khung, ta dẫn xuất ra q hệ số cepstral c(m) theo công thức đệ quy sau:

$c_0 = \ln\sigma^2$

$$c_m = a_m + \sum_{k=1}^{m-1} (\frac{k}{m})\, c_k a_{m-k}; l \le m \le p$$

$$c_m = \sum_{k=1}^{m-1} (\frac{k}{m})\, c_k a_{m-k}; p < m < Q$$

Trong đó, σ^2 là độ lợi của mô hình LPC. Thông thường ta chọn Q ≈ (3/2)p.

2.3.2.4. Đặt trọng số cho các hệ số cepstral:

Do độ nhạy của các hệ số cepstral cấp thấp làm cho phổ bị đổ dốc và do độ nhạy của các hệ số cepstral cấp cao gây ra nhiễu nên ta thường sử dụng kỹ thuật đặt trọng số để làm giảm thiểu độ nhạy này:

ĉi (m) = c(m).w(m)

Với w(m) là hàm đặt trọng số. Hàm đặt trọng số thích hợp thường là bộ lọc thông dải:

$$w(m) = [1 + \frac{Q}{2}\sin(\frac{\pi m}{Q})], 1 \le m \le Q \qquad (2.77)$$

Với miền tiếng nói hữu thanh có trạng thái gần ổn định, mô hình tất cả các điểm cực đại của LPC cho ta một xấp xỉ tốt đối với đường bao phổ âm. Với tiếng nói vô thanh, mô hình LPC tỏ ra ít hữu hiệu hơn so với hữu thanh, nhưng nó vẫn là mô hình hữu ích cho các mục đích nhận dạng tiếng nói. Mô hình LPC đơn giản và dễ cài đặt trên phần cứng lẫn phần mềm.

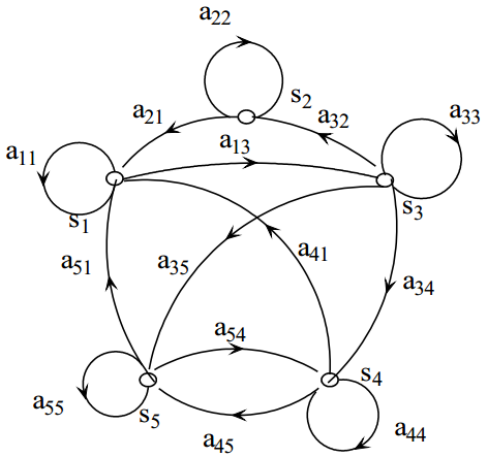
CHƯƠNG 3: NHẬN DẠNG TIẾNG NÓI

3.1. MÔ HÌNH MARKOV ẨN:

Một phương pháp hiệu quả dùng để mô hình hóa cấu trúc động của tiếng nói là mô hình Markov ẩn viết tắt là HMM *(Hidden Markov Models)*. Đây là hướng tiếp cận đối sánh mẫu xác suất, với giả định rằng ở đó các mẫu tiếng nói tuần tự theo thời gian là kết quả của quá trình thống kê hay ngẫu nhiên có tham số, và các tham số này có thể ước lượng. Mỗi từ sau khi qua khâu trích đặc trưng ta thu được một dãy véctơ P chiều (P = số hệ số MFCC), và được kí hiệu là t_1, t_2, …, t_i, …, t_l. Qua khâu lượng tử véctơ, dãy véctơ đặc trưng này được biến đổi thành các quan sát (là các ký hiệu sau khi phân lớp lượng tử véctơ) và được kí hiệu là o_1, o_2, …, o_t, …, o_T. Giả định cơ bản của HMM là mẫu dữ liệu có thể mô tả kỹ như quá trình hình thành một tham số ngẫu nhiên, và các tham số của quá trình phỏng đoán có thể ước tính trong mô hình được định nghĩa rõ ràng và chính xác. Lý thuyết HMM cơ bản được công bố trong một loạt tài liệu của Baum và đồng nghiệp của ông.

3.1.1. Chuỗi Markov rời rạc:

Xét hệ thống có tính chất như sau: ở một thời điểm bất kỳ, hệ thống sẽ ở một trong N trạng thái như hình vẽ dưới đây. Cứ sau một khoảng thời gian đều đặn, hệ thống sẽ chuyển sang trạng thái mới hoặc giữ nguyên trạng thái trước đó. Ta ký hiệu các khoảng thời gian chuyển trạng thái là t=1, 2, … và trạng thái tại thời điểm t của hệ thống là q_t, q_t sẽ có các giá trị 1, 2, …, N. 1 trạng thái tương ứng với 1 sự kiện. Quá trình trên được gọi là quá trình Markov.



Hình 3.1. Minh họa mô hình Markov

Trong hình vẽ, a_{ij} là xác suất chuyển từ trạng thái i sang trạng thái j, ta có các quan hệ:

$$a_{ij} \geq 0, \quad \forall\, i,j$$
$$\sum_{j=1}^{N} a_{ij} \;\forall\, i \qquad (3.1)$$

Ta chỉ xét chuỗi Markov bậc nhất là những hệ thống mà trạng thái hiện tại chỉ phụ thuộc vào trạng thái ngay trước đó, nghĩa là:

$$a_{ij} = P[q_t = j \mid q_{t-1} = i]\ ,\ 1 \leq i, j \leq N \qquad (3.2)$$

Các thành phần trong mô hình Markov:

- N trạng thái của mô hình. Ký hiệu trạng thái ở thời điểm t là q_t.

- N sự kiện: $E = \{e_1, e_2, e_3, ..., e_N\}$. Mỗi sự kiện tương ứng với 1 trạng thái. Tại mỗi thời điểm t, trạng thái phát sinh ra sự kiện tương ứng với nó.

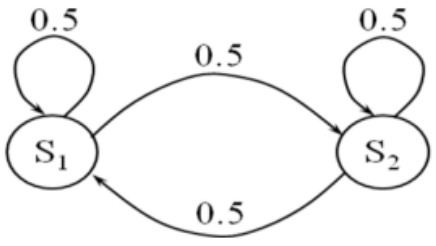
- $A=\{a_{ij}\}$ - là ma trận phân phối xác suất chuyển trạng thái, trong đó a_{ij} là xác suất chuyển từ trạng thái i ở thời điểm t sang trạng thái j ở thời điểm t+1:

$$a_{ij} = P[q_t = j \mid q_{t-1} = i] \qquad 1 \leq i, j \leq N$$

- $\pi = \{\pi_i\}$ - ma trận phân phối trạng thái ban đầu trong đó π_i là xác suất mô hình ở trạng thái i tại thời điểm ban đầu $t = 1$;

$$\pi_i = P[q_1 = i] \qquad 1 \leq i \leq N$$

Ví dụ 1: Tung đồng xu.



Ở đây có 2 trạng thái: S_1 tương ứng với sự kiện e_1 = Xấp; và S_2 tương ứng với sự kiện e_2 = Ngửa.

Ta có các phần tử của ma trận A:

$a_{11} = 0.5$ $a_{12} = 0.5$

$a_{21} = 0.5$ $a_{22} = 0.5$

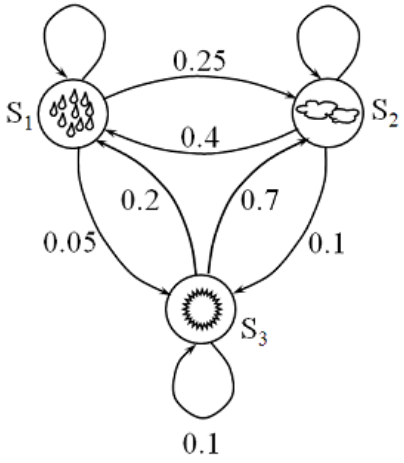
Các sự kiện:

Xấp Xấp Ngửa Ngửa Xấp Ngửa Xấp

tương ứng với các trạng thái:

S_1 S_1 S_2 S_2 S_1 S_2 S_1

Ví dụ 2: Thời tiết của một vùng với mô hình xác suất như sau.



S_1 = rain
S_2 = clouds ; $A = \{a_{ij}\} = \begin{bmatrix} .70 & .25 & .05 \\ .40 & .50 & .10 \\ .20 & .70 & .10 \end{bmatrix}$; $\pi_1 = 0.5$, $\pi_2 = 0.4$, $\pi_3 = 0.1$
S_3 = sun

- Xác suất của chuỗi quan sát {rain, rain, rain, clouds, sun, clouds, rain} ứng với mô hình Markov trên là:

Quan sát = { r, r, r, c, s, c, r}

S = $\{S_1, S_1, S_1, S_2, S_3, S_2, S_1\}$

Time = {1, 2, 3, 4, 5, 6, 7} (days)

= $P[S_1]\ P[S_1|S_1]P[S_1|S_1]P[S_2|S_1]P[S_3|S_2]P[S_2|S_3]P[S_1|S_2]$

= 0.5*0.7*0.7*0.25*0.1*0.7*0.4

= 0.001715

- Xác suất của chuỗi {sun, sun, sun, rain, clouds, sun, sun) ứng với mô hình Markov trên là:

Quan sát = { s, s, s, r, c, s, s}

S = $\{S_3, S_3, S_3, S_1, S_2, S_3, S_3\}$

Time = {1, 2, 3, 4 , 5 , 6 , 7} (days)

= P[S3]P[S3|S3]P[S3|S3]P[S1|S3]P[S2|S1]P[S3|S2]P[S3|S3]

$$= \quad 0.1*0.1*0.1*0.2*0.25*0.1*0.1$$

$$= \quad 5.0*10^{-7} \ .$$

3.1.2. Định nghĩa mô hình Markov ẩn:

Khác với chuỗi Markov như trình bày ở trên, mô hình Markov ẩn có những đặc điểm sau:

- Từ 1 trạng thái có thể phát sinh hơn 1 sự kiện (hay còn được gọi là 1 quan sát).

- Chuỗi quan sát là hàm xác suất của trạng thái.

- Chúng ta có thể tính toán xác suất của các chuỗi trạng thái khác nhau từ một chuỗi quan sát.

Như vậy HMM vẫn phát sinh ra các quan sát. Số lượng trạng thái thông thường khác số lượng quan sát. Khi ở trạng thái S_i, có xác suất $p(o_1)$ để phát sinh sự kiện 1, xác suất $p(o_2)$ để phát sinh sự kiện 2...

Các thành phần của mô hình Markov ẩn:

- N là số lượng trạng thái của mô hình. {1,2,...,N} là các trạng thái. Ký hiệu trạng thái ở thời điểm t là q_t.

- M là số lượng quan sát phân biệt. Các ký hiệu quan sát tương ứng với tín hiệu vật lý mà hệ thống đang mô tả. Ta ký hiệu tập quan sát là $V=\{v_1, v_2, ..., v_M\}$. Đối với tín hiệu tiếng nói, M là kích thước codebook. v_i là mã của từng vector.

- $A = \{a_{ij}\}$ - là ma trận phân phối xác suất chuyển trạng thái, trong đó a_{ij} là xác suất chuyển từ trạng thái i ở thời điểm t sang trạng thái j ở thời điểm t+1

$$a_{ij} = P[q_t = j \mid q_{t-1} = i] \qquad 1 \le i, j \le N$$

- $B = \{b_j(k)\}$ - ma trận phân phối xác suất các ký hiệu quan sát, trong đó $b_j(k)$ là xác suất nhận được ký hiệu quan sát vk ở trạng thái j:

$$b_j(k) = P[o_t = v_k | q_t = j] \qquad 1 \le k \le M \qquad j=1, 2, \ldots N$$

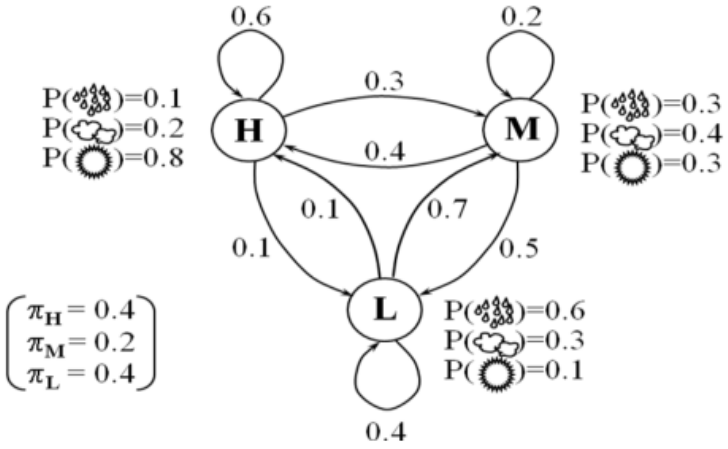
- $\pi = \{\pi_i\}$ - ma trận phân phối trạng thái ban đầu trong đó π_i là xác suất của mô hình ở trạng thái i tại thời điểm ban đầu t=1:

$$\pi_i = P[q_1=i] \qquad 1 \le i \le N$$

Như vậy để đặc tả đầy đủ một HMM cần phải có số trạng thái N của mô hình, tập V gồm M ký hiệu quan sát, ma trận xác suất chuyển trạng thái A, ma trận xác suất các ký hiệu quan sát được B và ma trận xác suất trạng thái ban đầu π.

Ví dụ: Thời tiết và độ ẩm không khí.

Cho mô hình Markov như hình sau:



Mô hình này có: Số trạng thái N=3 (gồm ***High, Medium, Low***)

Số ký hiệu quan sát M=3 (gồm ***Rain, Cloud, Sun***)

Các giá trị các phần tử của ma trận A, B, π như trên hình trên.

- Nếu cho chuỗi quan sát O = {sun, sun, cloud, rain, cloud, sun} và mô hình Markov ẩn như hình vẽ trên, thì xác suất để có chuỗi trạng thái {*H, M, M, L, L, M*} là bao nhiêu?

→ Xác suất cần tìm =

$b_H(sun)*b_M(sun)*b_M(cloud)*b_L(rain)*b_L(cloud)*b_M(sun)$

= 0.8* 0.3*0.4*0.6*0.3*0.3

= $5.2*10^{-3}$

Cho mô hình Markov như hình vẽ trên. Tính xác suất để có được chuỗi quan sát O = {sun, sun, cloud, rain, cloud, sun} và chuỗi trạng thái là {H, M, M, L, L, M}.

→ Xác suất cần tìm =

$\pi_H*b_H(s)*a_{HM}*b_M(s)*a_{MM}*b_M(c)*a_{ML}*b_L(r)*a_{LL}*b_L(c)*a_{LM}*b_M(s)$

= 0.4*0.8*0.3*0.3*0.2*0.4*0.5*0.6*0.4*0.3*0.7*0.3

= $1.74*10^{-5}$

- Cho mô hình Markov như hình vẽ trên. Tính xác suất để có được chuỗi quan sát O={sun, sun, cloud, rain, cloud, sun}và chuỗi trạng thái là {H, H, M, L, M, H}.

→ Xác suất cần tìm =

$\pi_H*b_H(s)*a_{HH}*b_H(s)*a_{HM}*b_M(c)*a_{ML}*b_L(r)*a_{LM}*b_M(c)*a_{MH}*b_H(s)$

= 0.4*0.8*0.6*0.8*0.3*0.4*0.5*0.6*0.7*0.4*0.4*0.6

$= \quad 3.71*10^{-4}$

Với định nghĩa HMM ở trên, ta có ba vấn đề cơ bản cần quan tâm trước khi được áp dụng trong các ứng dụng thực tế.

i. **Vấn đề sự ước lượng** - cho trước một mô hình **Φ** và một dãy của các quan sát **X**=(X_1, X_2, ..., X_T), có xác suất là $P(\mathbf{X}|\ \mathbf{\Phi})$; xác suất của mô hình tạo ra cho các quan sát là gì?

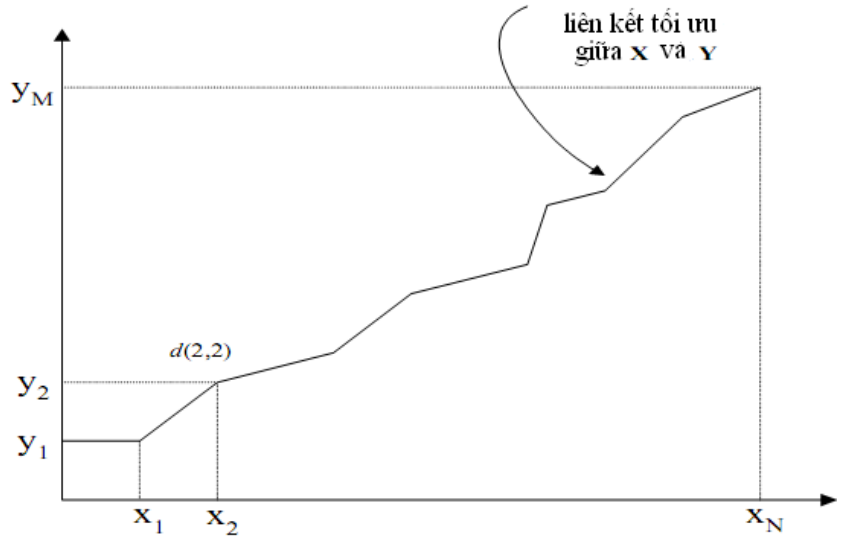
ii. **Vấn đề giải mã** - cho trước một mô hình **Φ** và một dãy của các quan sát **X**=(X_1, X_2, ..., X_T), dãy trạng thái phù hợp nhất **S** = (s_0, s_1, s_2, ..., s_T) trong mô hình tạo ra các quan sát là gì?

iii. **Vấn đề học** - cho trước một mô hình **Φ** và một tập các quan sát, làm thế nào chúng ta có thể điều chỉnh mô hình tham biến $\hat{\mathbf{\Phi}}$ để cực đại xác suất liên kết (có thể xảy ra) $\prod_X P(\mathbf{X}|\ \mathbf{\Phi})$?

Nếu chúng ta có thể giải quyết vấn đề sự ước lượng, chúng ta sẽ có một cách ước lượng làm sao với một HMM cho trước ăn khớp một dãy quan sát. Do đó, chúng ta có thể sử dụng HMM để nhận dạng mẫu, khi đó xác suất $P(\mathbf{X}|\mathbf{\Phi})$ có thể được sử dụng để tính toán xác suất hậu nghiệm $P(\mathbf{\Phi}|\mathbf{X})$, và HMM với xác suất hậu nghiệm cao nhất có thể được xác định như mẫu mong đợi cho dãy quan sát. Nếu chúng ta có thể giải quyết vấn đề giải mã, chúng ta có thể tìm được dãy trạng thái khớp nhất với một dãy quan sát cho trước, hay trong các từ khác, chúng ta có thể khám phá dãy trạng thái ẩn làm cơ sở cho quá trình giải mã trong nhận dạng tiếng nói liên tục. Sau cùng nếu chúng ta có thể giải quyết vấn đề học, chúng ta sẽ có thể ước lượng tự động mô hình tham biến Φ từ một bộ dữ liệu huấn luyện. Ba vấn đề này được liên kết dưới nền tảng xác suất tương tự. Sự bổ sung hiệu quả của các thuật toán chia sẻ các nguyên tắc của lập trình động sau.

3.1.2.1. Lập trình động và DTW:

Khái niệm lập trình động còn được biết như DTW *(Dynamic Time Warping)* trong nhận dạng tiếng nói, được sử dụng rộng rãi để dẫn suất toàn diện tình trạng không phân biệt rõ giữa hai mẫu tiếng nói. Phương pháp DTW có thể làm lệch hai mẫu tiếng nói (x_1,x_2,..., x_N) và (y_1, y_2, ..., y_M) trong chiều thời gian để giảm bớt tình trạng không rõ như minh họa trong hình dưới:



Hình 3.2. So sánh trực tiếp giữa hai mẫu tiếng nói

$X=(x_1,x_2,\ldots x_N)$ và $Y=(y_1, y_2, \ldots y_M)$

Điều này tương đương với vấn đề tìm kiếm khoảng cách cực tiểu trong lưới giữa hai mẫu. Được liên kết với mọi cặp (i, j) là một khoảng cách $d(i, j)$ giữa hai véctơ tiếng nói $\mathbf{x_i}$ và $\mathbf{y_i}$. Để tìm đường dẫn tối ưu giữa điểm bắt đầu (1, 1) và điểm cuối (N, M) từ trái sang phải, chúng ta cần tính khoảng cách chồng chất $D(N, M)$. Chúng ta có thể liệt kê tất cả khả năng khoảng cách chồng chất từ (1, 1) đến (N, M) và xác định mẫu có khoảng cách cực tiểu. Khi có M khả năng di chuyển cho mỗi bước từ trái sang phải trong hình trên, tất cả đường có khả năng từ (1, 1) đến (N, M) sẽ theo cấp số mũ. Nguyên tắc lập trình động có thể giảm mạnh lượng tính toán bằng cách tránh sự liệt kê của các dãy mà không thể tối ưu. Khi đường tối ưu tương tự sau đó mỗi bước phải dựa trên bước trước đó, khoảng cách cực tiểu $D(i, j)$ phải thỏa mãn biểu thức sau:

$$D(i, j) = \min_k [D(i\text{-}1, k)+d(k, j)] \qquad (3.3)$$

Công thức (3.3) cho biết ta chỉ cần xem xét và giữ lại chỉ bước đi tốt nhất đối với mỗi cặp mặc dù có thể có M khả năng bước đi. Sự đệ qui cho phép tìm kiếm đường dẫn tối ưu để được tiến hành gia tăng từ trái qua phải. Về bản chất, lập trình động giao phó giải pháp đệ quy cho vấn đề con của chính nó. Quá trình tính toán bắt nguồn từ vấn đề con $(D(i\text{-}1, k))$ đến vấn đề con lớn hơn $(D(i, j))$. Chúng ta có thể xác định $\mathbf{y}_j$ ăn khớp nhất với $\mathbf{x}_i$ và lưu lại chỉ mục trong bảng con trỏ lùi $B(i, j)$ là chúng ta đã đi qua. Đường dẫn tối ưu nhất có thể lần ngược lại sau khi đường dẫn tối ưu đã được xác định.

3.1.2.2. Ước lượng HMM - Thuật toán tiến:

Toán tử tiến $\propto_t (i)$ là xác suất của chuỗi quan sát từng phần **X** = (X_1, X_2, ..., X_t) và trạng thái quan sát S_i tại thời điểm t với điều kiện cho HMM λ.

$$\propto_t (i) = P(X_1 X_2 \dots X_t, q_t = s_i|\lambda)$$

Thuật toán tiến:

Bước 1: Khởi tạo

$$\propto_t (i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$$

Bước 2: Qui nạp

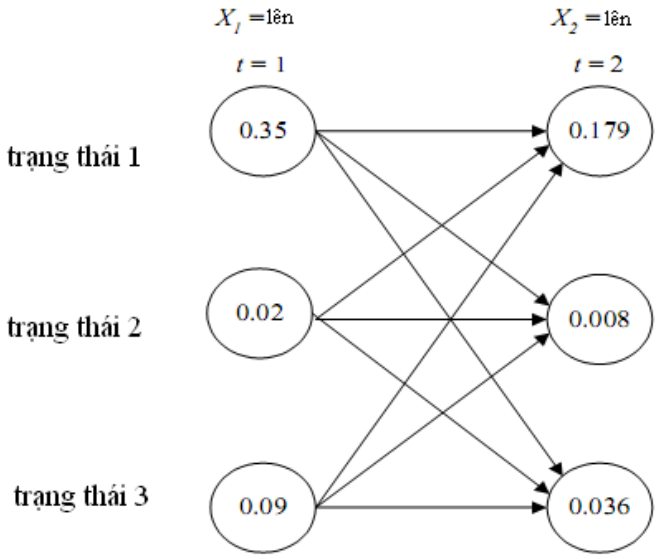
$$\propto_t (i) = \left[\sum_{i=1}^{N} \propto_{t-1} (i) a_{ij}\right] b_j(X_t) \qquad 2 \le t \le T;\ 1 \le j \le N$$

Bước 3: Kết thúc

$P(\mathbf{X}|\mathbf{\Phi}) = \sum_{i=1}^{N} \propto_T (i)$ nếu được yêu cầu để kết thúc trạng thái sau cùng,

$P(\mathbf{X}|\mathbf{\Phi}) = \propto_T (s_F)$

Ta có thể dễ dàng biết được độ phức tạp của thuật toán tiến là $O(N^2T)$ tốt hơn so với độ phức tạp cấp số mũ. Đó là bởi vì chúng ta có thể sử dụng toàn bộ các phần xác suất đã tính toán cho hiệu quả được cải tiến.



Hình 3.3. Quá trình tính toán lưới tiến cho HMM của Dow Jones Industrial

3.1.2.3. Giải mã HMM - Thuật toán Viterbi:

Thuật toán tiến, trong phần trước, tính toán xác suất mà một HMM tạo ra chuỗi quan sát bằng tổng các xác suất của tất cả đường dẫn có thể, cho nên nó không cung cấp đường dẫn tốt nhất (hoặc dãy trạng thái). Ở nhiều ứng dụng, người ta mong tìm được đường dẫn như vậy. Tìm đường dẫn tốt nhất (dãy trạng thái) là

nền móng cho quá trình tìm kiếm trong nhận dạng tiếng nói liên tục. Khi dãy trạng thái được ẩn (không được quan sát) trong nền tảng HMM, hầu hết sử dụng rộng rãi nhất tiêu chuẩn là để tìm dãy trạng thái có xác suất cao nhất được lấy trong khi tạo ra dãy quan sát. Nói cách khác, chúng ta đang tìm kiếm dãy trạng thái $\mathbf{S} = (s_1, s_2, \ldots, s_T)$ mà cực đại $P(\mathbf{S}, \mathbf{X}|\mathbf{\Phi})$. Vấn đề này rất giống với vấn đề tối ưu đường dẫn trong lập trình động. Hệ quả là, một kỹ thuật chính thức dựa trên lập trình động, gọi là thuật toán Viterbi, có thể được dùng để tìm dãy trạng thái tốt nhất cho HMM. Thực tế, phương pháp tương tự được dùng để đánh giá HMM mang lại cho giải pháp xấp xỉ gần với trường hợp đạt được việc sử dụng thuật toán tiến mô tả ở trên.

Thuật toán Viterbi có thể được xem như thuật toán lập trình động áp dụng cho HMM hay là thuật toán tiến sửa đổi. Thay vì tổng kết xác suất từ các con đường khác đến trạng thái đích, thuật toán Viterbi lấy và nhớ đường dẫn tốt nhất. Để định nghĩa xác suất đường dẫn tốt nhất:

$$V_t(i) = P(X_1^t, S_1^{t-1}, s_t = i|\Phi) \tag{3.4}$$

$V_t(i)$ là xác suất có khả năng nhất của dãy trạng thái ở thời điểm *t*, mà đã tạo ra quan sát X_1^t (cho đến thời điểm *t*) và kết thúc ở trạng thái *i*. Một thủ tục qui nạp tương tự cho thuật toán Viterbi có thể được mô tả như sau:

Thuật toán Viterbi:

Bước 1: Khởi tạo

$V_1(i) = \pi_i b_i(X_1)$ $\quad 1 \le i \le N$

Bước 2: Qui nạp

$V_t(j) = Max_{1 \le i \le N}\left[V_{t-1}(i)a_{ij}\right]b_j(X_t)$ $\quad 2 \le t \le T;\ 1 \le j \le N$

$B_t(j) = Arg\ max_{1 \le i \le N}\left[V_{t-1}(i)a_{ij}\right]$ $\quad 2 \le t \le T;\ 1 \le j \le N$

Bước 3: Kết thúc

Chỉ số tốt nhất = $Max_{1 \le i \le N}[V_t(i)]$

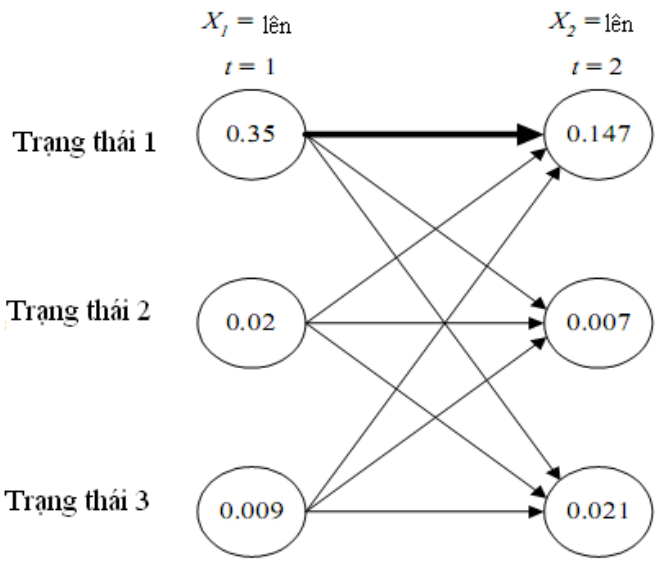
$S_T^* = Arg\ max_{1 \le i \le N}[B_T(i)]$

Bước 4: Quay lui

$S_T^* = B_{t+1}(S_{t+1}^*)$ $\quad t = T - 1, T - 2, \ldots, 1$

$\mathbf{S}^* = (s_1^*, s_2^*, \ldots, s_T^*)$ là dãy tốt nhất

Độ phức tạp của thuật toán Viterbi là $O(N^2T)$



Hình 3.4. Quá trình tính toán lưới Viterbi cho HMM của Dow Jones Industrial

3.1.2.4. Ước lượng các tham biến HMM - Thuật toán Baum-Welch:

Rất quan trọng đối với ước lượng các tham biến mô hình **Φ** = (**A**, **B**, **π**) để mô tả chính xác các dãy quan sát. Đây là vấn đề khó nhất, vì chưa biết phương pháp phân tích tối ưu xác suất tổ hợp của dữ liệu huấn luyện trong công thức dạng đóng. Thay vào đó, vấn đề có thể giải quyết bằng thuật toán lặp Baum-Welch, còn được biết là thuật toán tiến-lùi (forward-backward). Vấn đề học HMM là trường hợp điển hình của học không giám sát, nơi dữ liệu là không đầy đủ vì dãy trạng thái ẩn. Trước khi mô tả thuật toán Baum-Welch, đầu tiên chúng tôi định nghĩa một vài thuật ngữ cần thiết. Ở một hiệu chỉnh tương tự với xác suất tiến, chúng ta định nghĩa xác suất lùi như sau:

$$\beta_t(i) = P(X_{t+1}^T | s_t = i, \Phi) \qquad (3.5)$$

Trong đó $\beta_t(i)$ là xác suất tạo ra quan sát từng phần X_{t+1}^T (từ t+1 đến kết thúc) cho trước, HMM trong trạng thái i ở thời điểm t, $\beta_t(i)$ có thể sau đó được tính toán một cách qui nạp;

Khởi tạo:

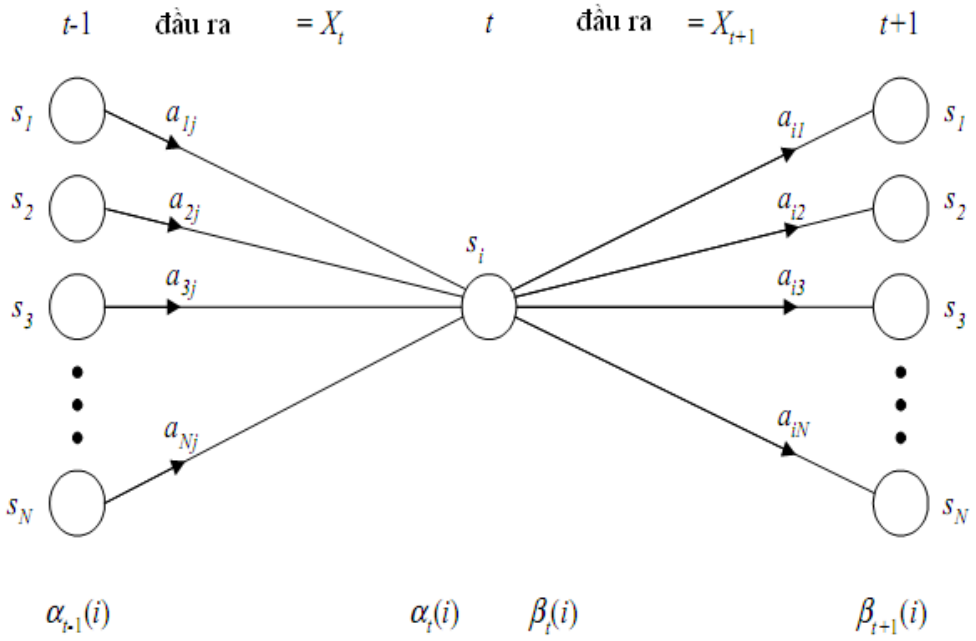
$$\beta_T(i) = 1/N \qquad 1 \le i \le N$$

Qui nạp:

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right] \qquad t = T - 1 \ldots 1; \quad 1 \le i \le N$$

Mối quan hệ liền kề α và β (α_{t-1} & α_t và β_t & β_{t+1}) có thể được mình họa như

hình bên dưới. α được tính một cách đệ qui từ trái sang phải, β đệ qui từ phải sang trái.



Hình 3.5. Mối quan hệ α_{t-1} & α_t và β_t & β_{t+1} trong thuật toán tiến-lùi

Tiếp theo chúng ta định nghĩa $\gamma_t(i, j)$ là xác suất của sự chuyển tiếp từ trạng thái i sang trạng thái j ở thời điểm t, cho trước mô hình và dãy quan sát.

$$\gamma_t(i,j) = \frac{\propto_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}$$

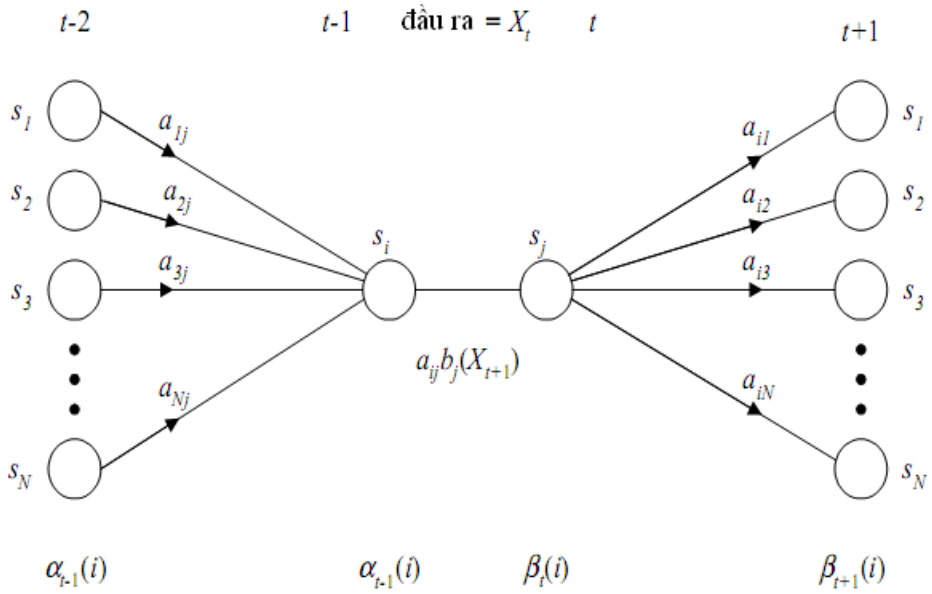
Chúng ta cải tiến lặp véctơ tham biến HMM $\mathbf{\Phi} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ bằng cách cực đại xác suất $P(\mathbf{X}|\mathbf{\Phi})$ cho mỗi lần lặp. Chúng ta sử dụng $\widehat{\mathbf{\Phi}}$ để biểu thị véctơ tham biến mới đã dẫn suất từ véctơ tham biến $\mathbf{\Phi}$ trong vòng lặp trước đó. Quá trình cực đại hóa là tương tự việc cực đại hàm Q như sau:

$$Q\left(\mathbf{\Phi}, \widehat{\mathbf{\Phi}}\right) = \sum_{all\ S} \frac{P(\mathbf{X}, \mathbf{S}|\mathbf{\Phi})}{P(\mathbf{X}|\mathbf{\Phi})} \log P(\mathbf{X}, \mathbf{S}|\widehat{\mathbf{\Phi}})$$

Trong đó:

$$P(\mathbf{X}, \mathbf{S}|\mathbf{\Phi}) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(X_t)$$

$$\log P(\mathbf{X}, \mathbf{S}|\mathbf{\Phi}) = \sum_{t=1}^{T} \log a_{s_{t-1}s_t} + \sum_{t=1}^{T} \log b_{s_t}(X_t)$$



Hình 3.6. Sự minh họa các phép toán yêu cầu cho việc tính toán của $\gamma_t(i, j)$.

Khi chúng ta tách hàm Q thành ba thuật ngữ độc lập, thủ tục cực đại hóa trên $Q(\mathbf{\Phi}, \hat{\mathbf{\Phi}})$ có thể được thực hiện bằng cực đại những thuật ngữ đơn rời rạc, đối tượng hướng đến là các ràng buộc xác suất. Chúng ta đạt được mô hình ước lượng như sau:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(i,k)} \tag{3.6}$$

$$\hat{b}_j(k) = \frac{\sum_{t \in X_t = o_k} \sum_i \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_i \gamma_t(i,j)} \tag{3.7}$$

Thuật toán tiến lùi (hay thuật toán Baum-Welch) có thể được mô tả như sau:

Thuật toán Baum-Welch:

Bước 1: Khởi tạo: chọn một ước lượng **Φ.**

Bước 2: E-step: tính hàm phụ trợ $Q(\mathbf{\Phi}, \hat{\mathbf{\Phi}})$ trên cơ sở **Φ.**

Bước 3: M-step: tính $\hat{\mathbf{\Phi}}$ theo ước lượng trong biểu thức (3.6) và (3.7) để cực đại hàm phụ trợ *Q*.

Bước 4: Quá trình lặp: thiết đặt $\mathbf{\Phi} = \hat{\mathbf{\Phi}}$, lập lại từ bước hai cho đến khi hội tụ.

3.1.3. Vấn đề thực tế trong sử dụng các HMM:

3.1.3.1. Ước lượng ban đầu:

Về mặt lý thuyết, thuật toán lượng giá của HMM nên đạt đến chỉ số tối đa cục bộ cho khả năng xảy ra. Câu hỏi then chốt là làm sao để chọn đúng ước tính ban đầu của các tham biến HMM sao cho chỉ số tối đa cục bộ trở thành tối đa toàn cục.

Ở HMM rời rạc, nếu một xác suất được khởi tạo là không, nó sẽ duy trì là không mãi. Do đó, điều quan trọng là phải có tập hợp các ước lượng ban đầu hợp lý. Nghiên cứu theo kinh nghiệm đã cho thấy, đối với HMM rời rạc, ta có thể sử dụng phân phối đồng bộ như ước lượng ban đầu. Nó thực hiện tốt một cách hợp lý cho hầu hết ứng dụng tiếng nói, ước lượng ban đầu tốt là luôn hữu ích để tính toán các xác suất đầu ra.

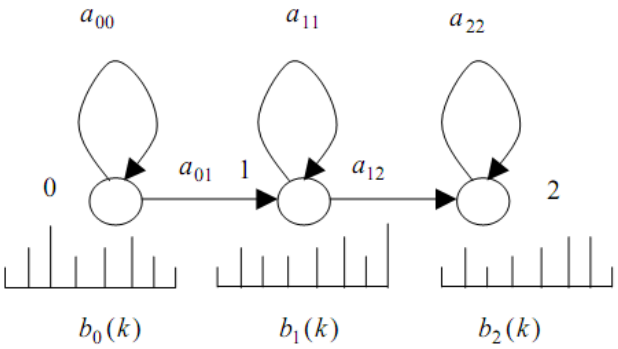
3.1.3.2. Cấu trúc liên kết mô hình:

Tiếng nói là tín hiệu không cố định. Mỗi trạng thái HMM có khả năng giữ một vài phân đoạn cố định trong tín hiệu tiếng nói không cố định. Cấu trúc từ trái sang phải, là thành phần tự nhiên để mô hình tín hiệu tiếng nói. Nó tự chuyển tiếp đến mỗi trạng thái, điều đó có thể được dùng để mô hình các đặc trưng tiếng nói liên tục thuộc về trạng thái giống nhau. Khi phân đoạn tiếng nói cố định rút ra, sự chuyển tiếp từ trái sang phải cho phép sự tiến triển tự nhiên của các sự thay đổi như vậy. Trong cấu trúc như vậy, mỗi trạng thái phụ thuộc phân phối xác suất đầu ra, có thể được dùng để thông dịch tín hiệu tiếng nói quan sát được. Cấu trúc này là một cấu trúc HMM phổ biến nhất được dùng trong các hệ thống nhận dạng tiếng nói tiên tiến nhất.

Trạng thái phụ thuộc phân phối xác suất đầu ra vừa có thể phân phối rời rạc hoặc hỗn hợp chức năng mật độ liên tục. Đây là trường hợp đặc biệt của chuyển tiếp-phụ thuộc các phân phối xác suất đầu ra. Trạng thái phụ thuộc các xác suất đầu ra có thể được xem như nếu sự chuyển tiếp phụ thuộc các phân phối xác suất đầu ra đã được gắn bó đối với mỗi trạng thái.

Đối với trạng thái HMM phụ thuộc từ trái sang phải, tham biến quan trọng nhất trong xác định cấu trúc là số trạng thái. Lựa chọn của mô hình cấu trúc tùy theo dữ liệu huấn luyện sẵn có và những gì mô hình được dùng. Nếu mỗi HMM được dùng để đại diện cho một âm, ta cần có ít nhất ba đến năm phân phối đầu ra. Nếu mô hình như vậy được dùng để đại diện cho một từ, nhiều hơn các trạng thái nói chung được yêu cầu, tùy vào phát âm và khoảng thời gian tồn tại của từ. Chẳng hạn như, từ tetrahydrocannabino nên có nhiều trạng thái trong so sánh với chữ a. Ta có thể dùng ít nhất 24 trạng thái cho phần trước và ba trạng thái cho phần sau. Nếu ta có số của trạng thái tùy vào khoảng thời gian tồn tại của tín hiệu, ta có lẽ cần dùng

15 đến 25 trạng thái cho mỗi giây của tín hiệu tiếng nói. Một ngoại lệ là, đối với khoảng lặng, ta có lẽ cần có một cấu trúc đơn giản hơn. Đây là vì khoảng lặng là cố định, và chỉ cần 1 hoặc 2 trạng thái sẽ đủ.



Hình 3.7. Mô hình Markov ẩn điển hình được dùng cho mô hình âm vị

Có 3 trạng thái (0-2) và mỗi trạng thái có một phân phối xác suất đầu ra kết hợp.

3.1.3.3. Tiêu chí huấn luyện:

Lập luận cho sự ước lượng khả năng xảy ra tối đa (MLE - Maximum Likelihood Estimation) được dựa trên một giả định là phân phối đúng của tiếng nói là một thành viên của các phân phối đã sử dụng. Các số lượng này xác nhận tiếng nói được quan sát thực sự được tạo ra bởi HMM đang dùng, và tham biến không rõ duy nhất là giá trị. Tuy nhiên, điều này có thể được thách thức. Các HMM điển hình tạo ra nhiều giả định không chính xác về quy trình tạo ra tiếng nói, như là giả định đầu ra độc lập, giả định Markov, và giả định hàm mật độ xác suất liên tục. Các giả định không chính xác làm yếu đi cơ sở hợp lý cho tiêu chí khả năng xảy ra tối đa. Chẳng hạn như, phương pháp ước lượng khả năng xảy ra tối đa là nhất quán (sự hội tụ đến giá trị đúng), nó là vô nghĩa để có một tính chất như vậy nếu mô hình sai được sử dụng. Tham biến đúng trong trường hợp này sẽ là tham biến đúng của các mô hình sai. Do đó, tiêu chuẩn phương pháp ước lượng có thể làm việc tốt mặc dù các giả định không chính xác này nên đưa ra độ xác nhận chính xác được so sánh với tiêu chuẩn khả năng xảy ra tối đa.

3.1.3.4. Phép nội suy loại bỏ:

Để cải thiện tính chắc chắn, thường cần thiết tổng hợp mô hình tổng quát đã được huấn luyện tốt (như độc lập người nói) với những mô hình được huấn luyện kém nhưng chi tiết hơn (phụ thuộc người nói). Chẳng hạn như, ta có thể nâng cao

độ chính xác nhận dạng tiếng nói với huấn luyện phụ thuộc người nói. Tuy vậy, ta có thể không có dữ liệu đủ cho người nói cụ thể vì vậy mong muốn để sử dụng một mô hình người nói độc lập là tổng quát hơn nhưng kém chính xác hơn trong tối ưu mô hình phụ thuộc người nói. Một phương pháp hiệu quả để đạt được sự chắc chắn là thêm vào cả hai mô hình với kỹ thuật được gọi là phép nội suy loại bỏ, trong đó đo phép nội suy đã sử dụng ước lượng qua việc hợp thức hóa dữ liệu. Hàm mục tiêu là để tối ưu xác suất của mô hình tạo ra dữ liệu.

Bây giờ, giả sử rằng chúng ta muốn nội suy hai tập hợp của các mô hình [$P_A(x)$ và $P_B(x)$, vừa có thể phân phối xác suất rời rạc hoặc hàm mật độ liên tục] để tạo thành một mô hình nội suy $P_{DI}(x)$. Thủ tục phép nội suy có thể biểu diễn ở dạng:

$$P_{DI}(x) = \lambda P_A(x) + (1-\lambda) P_B(x)$$

3.1.3.5. Tối ưu toán tử:

Một thực tế đơn giản cho mô phỏng xác suất là càng nhiều sự quan sát càng tốt, là cần thiết để ổn định mô hình ước lượng các tham biến. Tuy nhiên, thật ra, chỉ một số lượng hạn chế dữ liệu huấn luyện là sẵn có. Nếu dữ liệu huấn luyện bị giới hạn, điều này sẽ dẫn đến kết quả trong một vài tham biến đã huấn luyện là không thỏa đáng, và sự phân loại dựa trên các mô hình huấn luyện kém sẽ dẫn đến mức độ lỗi nhận dạng càng cao. Có nhiều giải pháp hợp lý để giải quyết vấn đề của dữ liệu huấn luyện không đầy đủ như sau:

- Ta có thể gia tăng kích thước của dữ liệu huấn luyện.
- Ta có thể giảm số tham biến tự do để được ước lượng lại. Điều này tạo nên các hạn chế của nó, vì một số các tham biến đáng kể luôn cần mô hình sự kiện.
- Ta có thể thêm vào một tập các tham biến ước lượng với một tập khác của tham biến ước lượng, theo đó đủ một lượng dữ liệu huấn luyện tồn tại. Xoá bỏ phép nội suy được đề cập ở trên, có thể được sử dụng hiệu quả. Trong HMM rời rạc, một phương pháp đơn giản là để thiết lập nền cho cả hai xác suất chuyển tiếp và xác suất đầu ra để loại bỏ khả năng ước lượng không.
- Ta có thể gom các tham biến với nhau để giảm số của tham biến tự do.
- Cho HMM hỗn hợp liên tục, ta cần chú ý đến tối ưu ma trận. Có một số kỹ thuật ta có thể sử dụng:

- Ta có thể nội suy ma trận với những mẫu huấn luyện tốt hơn.
- Ta có thể gom ma trận Gaussian thông qua các thành phần hỗn hợp khác nhau hoặc qua các trạng thái Markov khác nhau.
- Ta có thể sử dụng ma trận chéo nếu tương quan giữa các hệ số đặc trưng là yếu, sẽ đúng là trường hợp này nếu ta sử dụng các đặc trưng không tương quan như MFCC.
- Ta có thể kết hợp các phương pháp này với nhau.

Trong thực tế, chúng ta có thể giảm mức độ lỗi nhận dạng tiếng nói khoảng 5-20% với các kỹ thuật tối ưu khác nhau, tùy vào lượng dữ liệu huấn luyện sẵn có.

3.1.3.6. Biểu diễn xác suất:

Khi chúng ta tính toán các xác suất trước và sau trong thuật toán Forward-Backward, chúng sẽ tiếp cận không theo xu hướng cấp số mũ nếu chiều dài dãy quan sát, T, trở nên đủ lớn. Cho T đủ lớn, dãy linh động các xác suất sẽ vượt quá phạm vi độ chính xác của bất kỳ bộ máy nào về cơ bản. Do đó, trên thực tế, nó sẽ dẫn đến thiếu hụt trên máy tính nếu các xác suất được biểu diễn trực tiếp. Chúng ta có thể giải quyết vấn đề thi hành này bằng cách lấy tỉ lệ các xác suất này với một số hệ số tỉ lệ sao cho chúng ở bên trong dãy linh động của máy tính. Tất cả các hệ số tỉ lệ này có thể được xoá bỏ vào cuối quá trình tính toán không gây ảnh hưởng độ chính xác tổng thể.

Ví dụ, cho $\alpha_t(i)$ nhân với hệ số tỉ lệ, S_t:

$$S_t = 1/\sum_i \alpha_t(i) \tag{3.8}$$

Trong đó, $\sum_i S_t\alpha_t(i) = 1$, $1 \le t \le T$, $\beta_t(i)$ có thể được nhân bởi S_t, $1 \le t \le T$. Sự đệ quy được bao hàm trong quá trình tính toán các biến số trước và sau có thể được lấy tỉ lệ ở mỗi giai đoạn của thời gian t bởi S_t. Chú ý là $\alpha_t(i)$ và $\beta_t(i)$ được tính toán một cách đệ quy trong xu hướng cấp số mũ. Vì thế, ở thời điểm t hệ số tỉ lệ toàn bộ đã áp dụng cho biến số trước $\alpha_t(i)$ là:

$$Scale_\alpha(t) = \prod_{k=1}^{t} S_k \tag{3.9}$$

Và hệ số tỉ lệ toàn bộ cho biến số sau $\beta_t(i)$ là:

$$Scale_\beta(t) = \prod_{k=t}^{T} S_k \tag{3.10}$$

Đó là bởi vì các hệ số tỉ lệ riêng được nhân cùng với nhau trong đệ quy trước và sau. Cho $\propto'(i)$, $\beta'(i)$, và $\gamma_t'(i,j)$ biểu thị các biến số tỉ lệ tương ứng, một cách mong đợi. Chú ý là:

$$\sum_i \propto_T'(i) = Scale_\propto(T) \sum_i \alpha_T(i) = Scale_\propto(T) P(\mathbf{X}|\mathbf{\Phi}) \quad (3.11)$$

Xác suất tỉ lệ trực tiếp, $\gamma_t'(i,j)$, có thể sau đó được viết là:

$$\gamma_t'(i,j) = \frac{Scale_\propto(t-1) \propto_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j) Scale_\beta(t)}{Scale_\alpha(T) \sum_{i=1}^{N} \alpha_T(i)} \quad (3.12)$$

Như vậy, các xác suất trực tiếp có thể được sử dụng trong cùng một cách như các xác suất không tỉ lệ, bởi vì hệ số tỉ lệ đã được xóa bỏ trong biểu thức trên. Cho nên, việc ước lượng lại biểu thức có thể được giữ nguyên một cách chính xác ngoại trừ $P(\mathbf{X}|\mathbf{\Phi})$ nên được tính như sau:

$$P(\mathbf{X}|\mathbf{\Phi}) = \sum_i \propto_T'(i) / Scale_\propto(T) \quad (3.13)$$

Trong thực tế, toán tử tỉ lệ không cần biểu diễn ở mọi thời điểm quan sát. Nó có thể được sử dụng ở bất kỳ khoảng thời gian tính tỉ lệ nào cho sự thiếu hụt có thể xảy ra. Trong khoảng thời gian không tỉ lệ, $Scale_\propto$ có thể được giữ nguyên như hệ số đơn vị.

Một cách thay đổi để tránh sự thiếu hụt là sử dụng biểu diễn lôgarit cho tất cả các xác suất. Điều này không chỉ chắc chắn tính tỉ lệ là không cần thiết, vì thiếu hụt không thể xảy ra, mà còn cung cấp lợi ích là các số nguyên có thể được sử dụng để biểu diễn các giá trị lôgarit.

Trong thuật toán Forward-Backward, chúng ta cần xác suất thêm vào. Chúng ta có thể giữ một bảng lôgarit: $\log_b P_2$ - $\log_b P_1$. Nếu chúng ta biểu diễn xác xuất P bởi $\log_b P$, tăng độ chính xác có thể bao hàm bởi thiết lập b gần hơn đến hệ số đơn vị. Ta hãy xem là ta muốn thêm P_1 và P_2 và $P_1 \geq P_2$. Ta có:

$$\log_b(P_1 + P_2) = \log_b P_1 + \log_b(1 + b^{\log_b P_2 - \log_b P_1}) \quad (3.14)$$

Nếu P_2 mà quá nhiều ước lượng nhỏ hơn P_1 , thêm vào hai số sẽ chỉ có kết quả trong P_1. Chúng ta có thể lưu tất cả các giá trị của ($\log_b P_2$ - $\log_b P_1$). Sử dụng phương pháp lôgarit mang đến lỗi cho phép toán thêm vào. Trong thực tế, biểu diễn độ chính xác dấu chấm động kiểu double có thể được dùng để tối thiểu ảnh hưởng của vấn đề độ chính xác.

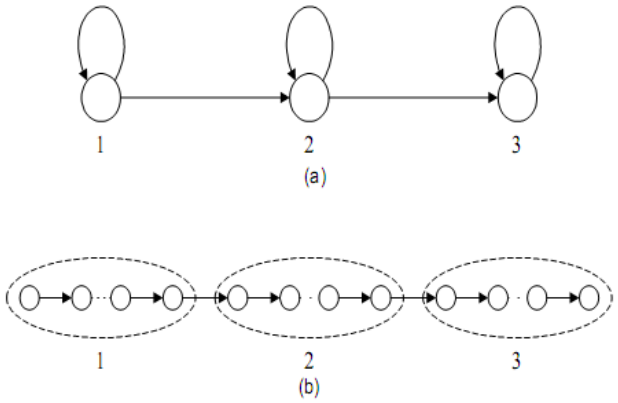
3.1.4. Những hạn chế của HMM:

Có một số hạn chế trong quy ước HMM. Chẳng hạn như, HMM lấy khoảng thời gian tồn tại như một phân phối theo cấp số mũ, xác suất chuyển tiếp chỉ dựa vào nguồn gốc và đích, và tất cả các khung quan sát đều phụ thuộc chỉ trên trạng thái đã tạo ra chúng, không phải gần kề các khung quan sát. Các nhà nghiên cứu đã đề xuất một số kỹ thuật để xử lý hạn chế này, mặc dù các giải pháp này đã không cải tiến đáng kể độ chính xác của nhận dạng tiếng nói trong các ứng dụng thực tế.

3.1.4.1. Mô phỏng khoảng thời gian tồn tại:

Một điểm yếu chính của quy ước HMM là chúng không cung cấp biểu diễn thích đáng của cấu trúc biểu thị thời gian của tiếng nói. Đây là vì xác suất của trạng thái thời gian giảm theo hàm mũ với thời gian như đã nêu trong biểu thức bên dưới. Xác suất của *t* các quan sát liên tục trong trạng thái *i* là xác suất của sự giữ vòng tự lặp ở trạng thái *i* cho thời gian *t*, có thể được viết như sau:

$$d_i(t) = a_{ii}^t(1 - a_{ii}) \tag{3.15}$$



Hình 3.8. Một HMM chuẩn

(a) và thời gian tồn tại quá trình HMM tương ứng (b) nơi mà các sự tự chuyển đổi được đổi chỗ với phân phối xác suất quy trình cho mỗi trạng thái

Cải tiến đến HMM chuẩn tạo ra bởi sử dụng HMM với phân phối quy trình thời gian rõ ràng cho mỗi trạng thái. Để giải thích nguyên tắc mô phỏng quy trình thời gian, quy ước HMM với mật độ quy trình trạng thái theo cấp số mũ và một quy trình thời gian HMM với các mật độ quy trình trạng thái đã xác định. Trong (a), xác suất quy trình trạng thái có một dạng theo cấp số mũ trong biểu thức (3.15). Trong (b), các xác suất tự chuyển đổi được thay thế với một phân phối xác suất quy trình rõ ràng. Ở thời điểm *t*, quá trình đưa vào trạng thái *i* cho quy trình τ với mật độ xác

suất $d_i(\tau)$, trong lúc các quá trình quan sát X_{t+1}, X_{t+2}, ...$X_{t+\tau}$ được tạo ra. Sau đó chuyển tiếp đến trạng thái j với xác suất chuyển đổi là a_{ij} chỉ sau đó các quan sát thích hợp τ xảy ra ở trạng thái i. Vì thế, bằng thiết lập mật độ xác suất quy trình thời gian để được mật độ theo cấp số mũ của biểu thức (3.15) quy trình thời gian HMM có thể được tạo ra tương đương với HMM chuẩn. Các tham biến $d_i(\tau)$ có thể được ước lượng từ các quan sát phù hợp với các tham biến khác của HMM đó. Xét tính thiết thực, mật độ quy trình thường bị cắt xén ở giá trị quy trình cực đại T_d. Để ước lượng lại các tham biến của HMM với mô phỏng quy trình thời gian, quá trình đệ quy ở trước đó phải được chỉnh sửa như sau:

$$\propto_t (j) = \sum_\tau \sum_{i,i\neq j} \propto_{t-\tau} (i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t-\tau+1}) \qquad (3.16)$$

Sự chuyển tiếp từ trạng thái i sang trạng thái j không chỉ phụ thuộc xác suất chuyển đổi a_{ij} mà còn trên tất cả các khả năng trong khoảng thời gian τ có thể xảy ra trong trạng thái j. Biểu thức (3.16) minh họa khi trạng thái j được đạt đến từ trạng thái i trước đó, các quan sát có thể giữ ở trạng thái j cho một khoảng thời gian τ với mật độ quy trình $d_i(\tau)$, và mỗi quan sát tạo ra xác suất đầu ra của chính nó. Tất cả quy trình có khả năng phải được xem xét, với sự tổng kết mong muốn đạt đến τ. Giả định độc lập của các quan sát mang đến kết quả trong thuật ngữ $\prod$ của các xác suất đầu ra. Tương tự, sự đệ quy ở phía sau có thể được viết như sau:

$$\beta_t(i) = \sum_\tau \sum_{j,j\neq i} a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t+1}) \beta_{t+\tau}(j) \qquad (3.17)$$

Thuật toán Baum-Welch cải tiến có thể được sử dụng trên cơ sở biểu thức (3.16) và (3.17).

Ngoài ra, mặt không thuận lợi để sử dụng mô phỏng quy trình thời gian là sự gia tăng lớn trong độ phức tạp tính toán bằng biểu thức $O(D^2)$. Vấn đề khác là số lượng lớn các tham biến thêm vào D phải được ước lượng. Một biện pháp đề suất là sử dụng hàm mật độ liên tục thay vì phân phối rời rạc $d_i(\tau)$.

Trong thực tế, các mô phỏng quy trình đã cung cấp sự cải tiến bình thường cho nhận dạng tiếng nói liên tục độc lập người nói. Nhiều hệ thống thậm chí rút ra xác suất chuyển tiếp hoàn toàn bởi vì các xác suất đầu ra mang tính chi phối. Tuy nhiên, thông tin quy trình rất hiệu quả cho việc cắt tỉa không chắc các phần tham gia trong quá trình giải mã nhận dạng tiếng nói có bộ từ vựng lớn.

3.1.4.2. Giả định bậc đầu tiên:

Khoảng thời gian tồn tại của mỗi phân đoạn cố định giữ bằng trạng thái đơn là không thỏa đáng mô hình. Cách khác để làm giảm nhẹ vấn đề khoảng thời gian tồn tại là để loại bỏ giả định sự chuyển tiếp bậc đầu tiên và để tạo nên dãy trình tự trạng thái dưới một chuỗi Markov bậc hai. Kết quả là xác suất chuyển tiếp giữa hai trạng thái ở thời điểm *t* phụ thuộc các trạng thái mà trong đó quá trình ở thời điểm t - 1 và t - 2. Cho trước một dãy trạng thái **S** = {s_1, s_2, … s_T}, xác suất của trạng thái nên tính toán như sau:

$$P(\boldsymbol{S}) = \prod_{\boldsymbol{t}} \boldsymbol{a}_{\boldsymbol{s_{t-2}s_{t-1}s_t}} \tag{3.18}$$

Trong đó $a_{s_{t-2}s_{t-1}s_t} = P(s_t|s_{t-2}s_{t-1})$ là xác suất chuyển tiếp ở thời điểm *t*, cho trước hai bậc trạng thái. Thủ tục sự ước lượng lại có thể được mở rộng sẵn sàng trên cơ sở (3.18).

Trong thực tế, mô hình bậc hai rất tốn kém trong quá trình tính toán như chúng ta phải xem xét không gian trạng thái gia tăng, mà có thể thường được nhận ra với mô hình Markov ẩn bậc một tương đương trên không gian trạng thái. Nó không cung cấp gia tăng độ chính xác một cách đáng kể để sắp xếp cho đều nhau sự gia tăng của nó trong độ phức tạp tính toán cho hầu hết ứng dụng.

3.1.4.3. Giả định độc lập có điều kiện:

Điểm yếu chính thứ ba của HMMs là tất cả các khung quan sát đều phụ thuộc chỉ trên trạng thái tạo ra chúng, không phải gần kề các khung quan sát. Giả định độc lập có điều kiện khiến nó khó mà xử lý một cách hiệu quả các khung không cố có mối tương liên mạnh mẽ. Có một số cách để làm giảm nhẹ giả định độc lập có điều kiện. Chẳng hạn như, chúng ta có thể giả định phân phối xác suất đầu ra phụ thuộc không những trên trạng thái mà còn trên khung trước đó. Do đó, xác suất của trình tự trạng thái cho trước có thể viết lại như:

$$P(\mathbf{X}|\mathbf{S}, \boldsymbol{\Phi}) = \prod_{\mathbf{t=1}}^{\mathbf{T}} P(X_t|X_{\boldsymbol{t-1}}, s_{\boldsymbol{t}}, \boldsymbol{\Phi}) \tag{3.19}$$

Vì không gian tham biến trở nên quá lớn, chúng ta thường cần lượng tử hóa X_{t-1} trong một tập hợp nhỏ hơn của các từ mã để có thể giữ cho số các tham biến tự do trong kiểm soát. Vì vậy, biểu thức (3.20) có thể được đơn giản như sau:

$$P(\mathbf{X}|\mathbf{S}, \boldsymbol{\Phi}) = \prod_{\mathbf{t=1}}^{\mathbf{T}} P(X_t|\Re(X_{\boldsymbol{t-1}}), s_{\boldsymbol{t}}, \boldsymbol{\Phi}) \tag{3.20}$$

Trong đó $\Re()$ biểu thị véctơ lượng tử có một kích cở các ký hiệu nhỏ, L. Mặc dù điều này có thể giảm không gian của các phân phối xác suất đầu ra có điều kiện tự do, số tổng cộng của các tham biến tự do sẽ vẫn tăng lên bằng L lần.

Sự ước lượng cho các HMM phụ thuộc điều kiện có thể được dẫn suất với sự thay đổi hàm Q, như đã thảo luận trong phần trước đó. Trong thực tế, nó không được chứng minh độ chính xác thuyết phục cải tiến cho nhận dạng tiếng nói bộ từ vựng lớn.

3.2. MÔ HÌNH ÂM HỌC:

Độ chính xác của nhận dạng tiếng nói tự động luôn là một trong những vấn đề nghiên cứu quan trọng nhất. Mô hình âm học đóng vai trò quyết định để cải thiện độ chính xác và có thể xem như thành phần trung tâm trong bất cứ hệ thống nhận dạng nào.

Với một chuỗi quan sát âm học cho trước $X = X_1, X_2, \ldots X_n$, mục tiêu của nhận dạng tiếng nói là tìm ra chuỗi tiếng tương ứng $\hat{W} = w_1, w_2, \ldots w_n$ có xác suất hậu cực đại P(W | X) biểu diễn bởi biểu thức:

$$\hat{W} = arg_w max P(W \mid X) = arg_w max \frac{P(W)P(X|W)}{P(X)} \qquad (3.21)$$

Với X cố định, biểu thức trên đạt cực đại khi biểu thức sau đạt cực đại:

$$\hat{W} = arg_w max P(W)P(X \mid W) \qquad (3.22)$$

Bài toán đặt ra là làm sao xây dựng các mô hình âm học, P(X | W) và mô hình ngôn ngữ P(W) thực sự phản ánh được ngôn ngữ nói được nhận dạng. Đối với nhận dạng với bộ từ vựng lớn, cần phải phân tích một tiếng ra thành chuỗi từ con (subword). Do đó, P(X | W) có liên hệ gần với mô hình âm tiết. P(X | W) cần tính đến những sự thay đổi về người nói, cách phát âm, môi trường xung quanh và sự kết hợp phát âm ngữ âm phụ thuộc ngữ cảnh. Bất cứ mô hình âm học hay ngôn ngữ mô hình thống kê nào cũng không thể đáp ứng được nhu cầu của các ứng dụng thực tế, vì vậy, điều quan trọng là làm thích ứng động cả P(W) và P(X|W) để cực đại hóa P(W|X) trong việc dùng các hệ thống ngôn ngữ nói.

3.2.1. Lựa chọn đơn vị thích hợp cho mô hình âm học:

Đối với mục tiêu nhận dạng tiếng nói trên bộ từ vựng lớn, việc xây dựng

các mô hình toàn từ gặp nhiều khó khăn vì:

- Mỗi tác vụ mới lại chứa các từ mới lạ mà không có bất cứ dữ liệu huấn luyện sẵn có nào, chẳng hạn những danh từ riêng và các thuật ngữ mới được đưa ra.

- Có quá nhiều từ, và các từ khác nhau này có thể có các đặc điểm âm thanh khác nhau.

- Việc lựa chọn các đơn vị cơ bản để biểu diễn đặc trưng âm học và thông tin ngữ âm cho ngôn ngữ là một vấn đề rất quan trọng trong việc thiết kế một hệ thống khả thi.

Một số vấn đề cần phải xem xét trong việc lựa chọn các đơn vị mô hình hóa chính xác:

- Đơn vị này phải chính xác để biểu diễn hiện thực âm thanh xuất hiện trong các ngữ cảnh khác nhau.

- Đơn vị này phải huấn luyện được. Phải có đủ dữ liệu để ước lượng các tham số cho đơn vị này. Mặc dù từ là đơn vị chính xác và tiêu biểu, chúng lại ít có khả năng huấn luyện nhất trong việc xây dựng một hệ thống khả thi do gần như không thể huấn luyện lặp lại hàng trăm lần cho tất cả các từ, trừ khi ta xây dựng một bộ nhận dạng trong một lĩnh vực cụ thể.

- Đơn vị này phải có tính tổng quát để bất cứ từ mới nào cũng có thể kế thừa từ một bản được định nghĩa trước đối với hệ thống nhận dạng tiếng nói độc lập tác vụ. Nếu có một tập tập cố định các mô hình từ thì gần như không có cách nào để một mô hình từ mới kế thừa từ đó.

3.2.1.1. So sánh các đơn vị khác nhau:

Trong tiếng Anh, từ thường được coi là đơn vị nhỏ nhất mang ý nghĩa và có thể sử dụng độc lập. Là đơn vị tự nhiên nhất của tiếng nói, mô hình toàn từ đã được sử dụng rộng rãi cho nhiều hệ thống nhận dạng tiếng nói. Một lợi thế của việc sử dụng mô hình từ là ta có thể nắm bắt cách phát âm vốn có trong những từ này. Khi bộ từ vựng nhỏ, ta có thể tạo các mô hình từ phụ thuộc ngữ cảnh.

Trong khi từ là đơn vị phù hợp cho nhận dạng tiếng nói trên bộ từ vựng nhỏ, chúng lại không phải là lựa chọn tốt đối với nhận dạng tiếng nói liên tục trên

bộ từ vựng lớn vì những lý do sau:

- Mỗi từ phải được xử lý riêng lẻ, và dữ liệu không thể được chia sẻ với nhau trong mô hình từ. Điều này khiến cho số lượng dữ liệu huấn luyện cần thiết là rất lớn.

- Đối với một số tác vụ, các từ vựng nhận dạng có thể bao gồm các từ không xuất hiện trong tập huấn luyện.

- Rất khó để làm thích nghi một mô hình từ sẵn có cho một người nói mới, một kênh mới hay một ngữ cảnh mới.

Thay vào đó, chỉ có khoảng 50 âm tố trong tiếng Anh và chúng có thể được huấn luyện đầy đủ chỉ với vài trăm câu. Không như mô hình từ, mô hình ngữ âm không phát sinh nhiều vấn đề trong việc huấn luyện. Hơn nữa, chúng độc lập với từ vựng và có thể được huấn luyện trên tác vụ này và kiểm tra trên tác vụ khác. Do đó, các âm tố có khả năng huấn luyện cao hơn và tổng quát hơn. Tuy nhiên, mô hình ngữ âm không thỏa đáng vì nó giả định rằng một âm vị trong mọi ngữ cảnh là giống nhau. Dù ta có thể cố gắng nói mỗi từ như là một chuỗi móc nối với nhau của các âm vị độc lập, các âm vị này không được phát sinh một cách độc lập vì khớp răng của ta không thể di chuyển ngay lập tức từ vị trí này đến vị trí khác. Do đó, hiện thực của một âm vị bị ảnh hưởng mạnh mẽ bởi các âm vị kề sát nó. Trong khi mô hình từ không tổng quát, mô hình ngữ âm lại quá tổng quát, và dẫn đến mô hình kém chính xác.

Một sự kết hợp giữa mô hình từ và mô hình ngữ âm là sử dụng một đơn vị âm tiết. Các đơn vị này bao gồm các bó âm tố chứa đựng hầu hết các tác động thay đổi ngữ cảnh. Tuy nhiên trong khi phần giữa của đơn vị này không phụ thuộc ngữ cảnh, phần bắt đầu và phần cuối vẫn bị tác động bởi một vài tác động ngữ cảnh.

3.2.1.2. Lựa chọn đơn vị huấn luyện cho tiếng Việt:

Trong tiếng Việt tiếng là đơn vị tự nhiên nhất cấu tạo nên lời nói, tuy số lượng tiếng trong tiếng Việt có giới hạn khoảng 6.000-8.000 nhưng nếu đứng ở góc độ nhận dạng tiếng nói thì đó là một số lượng đáng kể.

Trong khi đó, âm vị trong tiếng Việt bao gồm:

- 22 phụ âm đầu bao gồm /b, m, f, v, t, t', d, n, z, ʐ, s, ʂ, c, ʈ, ɲ, l, k, χ, ŋ, ɣ, h, ʔ/

- 1 âm đệm /w/ có chức năng làm *trầm hóa* âm sắc của âm tiết.

- 16 âm chính bao gồm 13 nguyên âm đơn và 3 nguyên âm đôi: /i, e, ɛ, ɤ, ɤˇ, a, ɯ, ă, u, o, ɔ, ɔˇ, ɛˇ, ie, ɯɤ, uo/

- 8 âm cuối tích cực bao gồm 6 phụ âm /m, n, ŋ, p, t, k/ và 2 bán nguyên âm /-w, -j/.

- 6 thanh điệu.

Có thể thấy số lượng âm vị không nhiều, do đó, việc ứng dụng mô hình ngữ âm vào nhận dạng tiếng Việt là một giải pháp đáng quan tâm. Tuy nhiên vấn đề khó khăn đối với tiếng Việt chính là thanh điệu.

Tuy thanh điệu ảnh hưởng lên toàn bộ tiếng, nhưng có thể thấy nó ảnh hưởng nhiều nhất là ở các nguyên âm. Vì vậy ta có thể chia mỗi nguyên âm ra thành 6 âm, tương ứng với 6 thanh điệu. Như vậy tổng số lượng âm cần huấn luyện là khoảng 137 âm, nhỏ hơn nhiều so với huấn luyện theo tiếng.

3.2.2. Đánh giá đặc trưng âm học:

Sau khi tách đặc trưng, ta có một tập các vector đặc trưng X, chẳng hạn vector MFCC là các dữ liệu đầu vào. Ta cần phải ước lượng xác suất của các đặc trưng âm học này, cho trước mô hình từ hoặc mô hình ngữ âm W, để có thể nhận dạng dữ liệu đầu vào cho từ đúng. Xác suất này được gọi là xác suất âm học, P(X|W).

3.2.2.1. Lựa chọn các phân phối đầu ra HMM:

Có thể sử dụng các HMM rời rạc, liên tục hoặc bán liên tục. Khi số lượng dữ liệu huấn luyện đã đủ, tham số ràng buộc trở nên không cần thiết. Một mô hình liên tục với một số lượng lớn các trộn lẫn dẫn đến độ chính xác nhận dạng tốt nhất, mặc dù độ phức tạp tính toán của nó cũng gia tăng tuyến tính với số lượng các hỗn hợp. Mặt khác, mô hình rời rạc có hiệu quả về mặt tính toán, nhưng có hiệu suất thấp nhất trong ba mô hình. Mô hình bán liên tục cung cấp một thay thế khả thi giữa khả năng huấn luyện và tính mạnh mẽ của hệ thống.

Khi một trong HMM rời rạc hay bán liên tục được sử dụng, việc dùng nhiều codebook cho một số đặc trưng sẽ nâng cao hiệu suất một cách đáng kể. Mỗi

codebook biểu diễn một tập các tham số khác nhau. Một cách để kết hợp các quan sát nhiều đầu ra là giả định rằng chúng độc lập với nhau, tính toán xác suất đầu ra như là sản phẩm của các xác suất mỗi codebook.

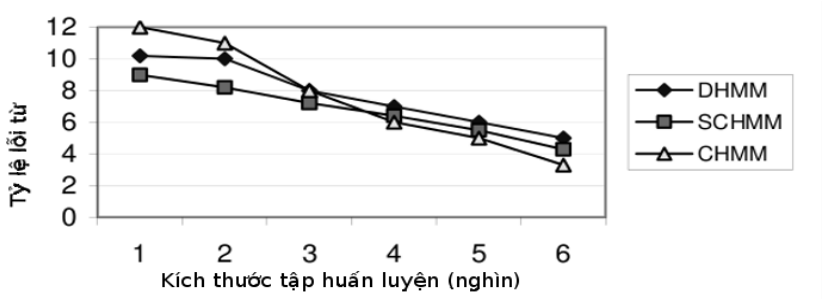
$$b_i(x) = \prod_m \sum_{k=1}^{L^m} f^m(x^m \mid o_k{}^m) b_i{}^m(o_k{}^m) \tag{3.23}$$

Trong đó, *m* biểu thị các tham số tương ứng codebook-*m*. Mỗi codebook gồm có các hàm mật độ liên tục hỗn hợp L^m.

Thuật toán đánh giá lại mô hình Markov ẩn dựa trên nhiều codebook (multiple-codebook-based HMM) có thể được mở rộng. Tích của mật độ xác suất đầu ra của mỗi codebook dẫn đến các term độc lập trong hàm *Q*, với codebook-*m*, $\xi_t(j, k^m)$ có thể được chỉnh lại như sau:

$$\zeta_t(j,k^m) = \frac{\sum_i \alpha_{t-1}(i) a_{ij} b_j{}^m(k^m) f^m(x_t|v_k{}^m) \prod_{m\neq n} \sum_k b_j{}^n(k^n) f^n(x_t|v_k{}^n) \beta_t(j)}{\sum_k \alpha_T{}^m(k)} \tag{3.24}$$

Sử dụng nhiều codebook có thể làm gia tăng nhanh chóng khả năng của VQ codebook và có thể cải tiến cơ bản độ chính xác nhận dạng tiếng nói. Ta có thể xây dựng một codebook điển hình cho c_k, Δc_k và $\Delta\Delta c_k$ lần lượt theo thứ tự. So sánh việc xây dựng một codebook đơn cho x_k, như hình dưới, hệ thống multiple-codebook có thể giảm thiểu tỷ lệ lỗi hơn 10%.



Hình 3.9. Tỉ lệ lỗi từ giữa các mô hình

Có thể thấy HMM bán liên tục có mức cải tiến độ chính xác nằm giữa mô hình HMM rời rạc và HMM liên tục khi số lượng dữ liệu huấn luyện có giới hạn. Khi ta tăng kích thước dữ liệu huấn luyện, HMM mật độ hỗn hợp liên tục bắt đầu tốt hơn hẳn so với cả HMM rời rạc và HMM bán liên tục, do đó việc chia sẽ các tham số mô hình trở nên ít quan trọng hơn.

Hiệu suất cũng phụ thuộc vào số lượng các hỗn hợp. Với một số lượng nhỏ

các hỗn hợp, HMM liên tục thiếu sức mạnh mô hình và nó thực sự kém hiệu quả. so với HMM rời rạc. Chỉ sau khi số lượng các hỗn hợp tăng lên đáng kể thì HMM liên tục bắt đầu gia tăng độ chính xác nhận dạng. HMM bán liên tục thường giảm thiểu tỷ lệ lỗi của HMM rời rạc từ 10-15%. HMM liên tục với 20 hàm mật độ chéo Gaussian thực thi kém hiệu quả hơn so với cả HMM rời rạc hay HMM bán liên tục khi kích thước dữ liệu huấn luyện nhỏ. Nó có hiệu suất vượt trội so với cả HMM rời rạc hay HMM bán liên tục khi có đủ dữ liệu huấn luyện. Khi số lượng huấn luyện đủ lớn, nó có thể giảm tỷ lệ lỗi của HMM bán liên tục từ 15-20%.

3.2.2.2. Huấn luyện tiếng nói rời rạc so với liên tục:

Nếu ta xây dựng một HMM từ cho mỗi từ trong bộ từ vựng cho nhận dạng tiếng nói rời rạc, quá trình huấn luyện hoặc nhận dạng có thể được thực hiện một cách trực tiếp, sử dụng các thuật toán cơ bản được trình bày ở phần mô hình Markov ẩn. Để ước lượng các tham số mô hình, các mẫu của mỗi từ trong bộ từ vựng đã được thu thập. Các tham số mô hình được ước lượng từ tất cả các các mẫu sử dụng thuật toán forward-backward và công thức ước lượng lại. Không cần thiết phải xác định điểm cuối do mô hình khoảng lặng tự động xác định giới hạn của nó nếu ta móc nối các mô hình khoảng lặng với mô hình từ ở cả hai điểm đầu và cuối.

Nếu các mô hình ngữ âm được sử dụng, ta cần phải chia sẽ chúng giữa các từ khác nhau đối với nhận dạng tiếng nói trên bộ từ vựng lớn. Các đơn vị ngữ âm được móc nối để tạo thành một mô hình từ, có thể thêm các mô hình khoảng lặng tại điểm đầu và điểm cuối.

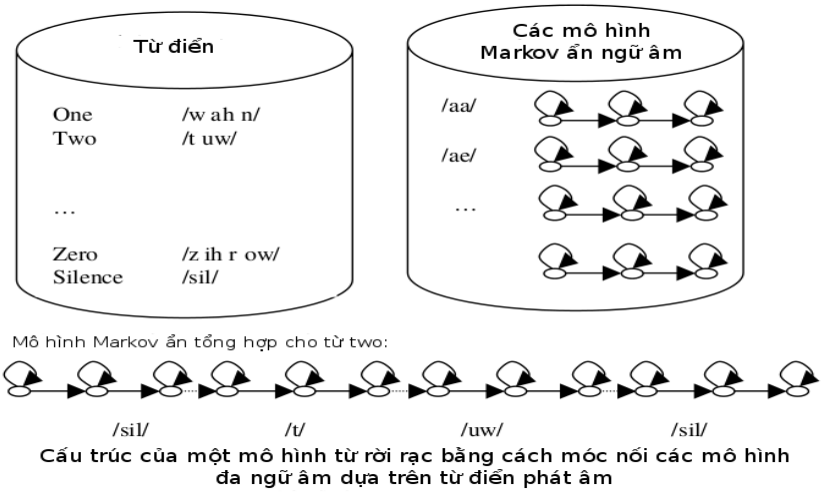
Để móc nối các ngữ âm thành dạng mô hình từ, có thể có sự chuyển đổi từ trạng thái cuối cùng của mô hình Markov ẩn ngữ âm trước sang trạng thái khởi tạo của mô hình Markov ẩn của ngữ âm kế tiếp. Có thể ước lượng các tham số của mô hình Markov ẩn móc nối. Lưu ý rằng việc thêm cung chuyển trạng thái rỗng nên thỏa mãn xác suất ràng buộc với xác suất chuyển trạng thái của mỗi mô hình Markov ẩn ngữ âm. Nếu ước lượng các tham số với mô hình móc nối, xác suất chuyển trạng thái cung rỗng a_{ij}^{g} phải thỏa mãn ràng buộc:

$$\sum_{j}(a_{ij} + a_{ij}{}^{g}) = 1 \tag{3.25}$$

Vì vậy, xác suất chuyển trạng thái vòng tự lặp của trạng thái cuối cùng luôn

nhỏ hơn 1. Đối với kết nối liên từ hay móc nối bao gồm nhiều cách phát âm, ta có thể sử dụng nhiều cung rỗng để móc nối các mô hình đơn lẻ với nhau.

Trong ví dụ trong hình dưới, ta có 10 chữ số tiếng Anh trong bộ từ vựng. Xây dựng một mô hình Markov ẩn cho mỗi âm tố tiếng Anh. Từ điển cung cấp thông tin cách phát âm của mỗi từ. Trong đó có một từ đặc biệt là Silence, ánh xạ với /sil/ trong mô hình Markov ẩn có dạng topology như mô hình Markov ẩn ngữ âm chuẩn. Với mỗi từ trong bộ từ vựng, đầu tiên ta dẫn xuất chuỗi ngữ âm cho mỗi từ trong từ điển. Sau đó kết nối các mô hình ngữ âm với nhau thành dạng một mô hình Markov ẩn một từ cho mỗi từ trong bộ từ vựng.



Hình 3.10. Cấu trúc của một mô hình từ rời rạc

Ví dụ với từ two, đầu tiên tạo một mô hình từ bắt đầu bởi silence /sil/, âm tố /t/, /uw/ và kết thúc bằng silence /sil/. Mô hình từ móc nối sau đó được xem như một mô hình Markov ẩn tổng hợp lớn chuẩn. Sử dụng thuật toán Forward-Backward chuẩn để ước lượng các tham số của mô hình Markov ẩn tổng hợp từ nhiều mẫu giọng nói của từ two. Sau vài lần lặp lại sẽ tự động thu được các tham số mô hình Markov ẩn cho /sil/, /t/ và /uw/. Do một âm tố có thể được chia sẻ trên các từ khác nhau, các tham số ngữ âm có thể được ước lượng từ dữ liệu âm học trong các từ khác nhau.

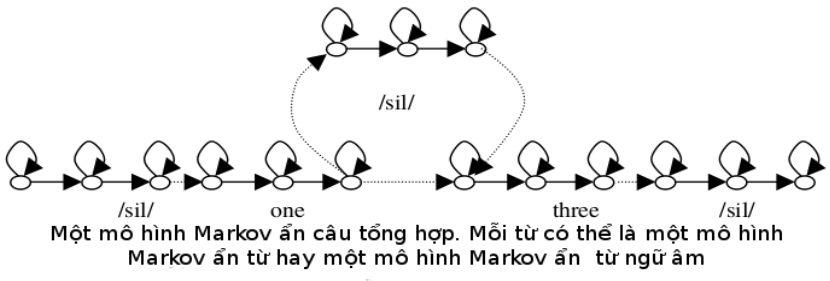
Khả năng tự động sắp xếp mỗi mô hình Markov ẩn đơn lẻ thành chuỗi quan sát tiếng nói không phân đoạn tương ứng là một trong những tính năng mạng mẽ nhất trong thuật toán Forward-Backward. Khi sử dụng phương pháp móc nối mô hình Markov ẩn cho tiếng nói liên tục cần phải sắp xếp nhiều từ thành dạng một mô

hình Markov ẩn câu dựa trên bản ghi của lời nói. Thuật toán Forward-Backward hấp thụ một dãy các thông tin ranh giới từ có thể của các mô hình một cách tự động, vì thế không cần phải phân đoạn tiếng nói liên tục một cách chính xác.

Để ước lượng các tham số của mô hình Markov ẩn, mỗi từ được khởi tạo với mô hình từ móc nối. Các từ trong câu được móc nối với các mô hình silence tùy chọn giữa chúng.

Nói chung, mô hình Markov ẩn câu kết nối có thể được huấn luyện sử dụng thuật toán forward-backward với chuỗi quan sát tương ứng. Do mô hình Markov ẩn toàn câu được huấn luyện trên toàn bộ chuỗi quan sát cho câu tương ứng, hầu hết các giới hạn từ có thể đều đã được xem xét. Các tham số của mỗi mô hình được dựa trên sự liên kết trạng thái với tiếng nói (state-to-speech aligments) đó. Phương pháp huấn luyện như vậy cho phép tự do hoàn toàn để liên kết các mô hình câu đối với quan sát này, và không cần phải cố gắng tìm giới hạn từ.

Trong giải mã tiếng nói, một từ có thể bắt đầu và kết thúc ở bất kỳ đâu trong phạm vi tín hiệu tiếng nói cho trước. Vì các giới hạn từ không thể được phát hiện một cách chính xác, tất cả các điểm bắt đầu và kết thúc phải được tính đến.



Hình 3.11. Mô hình Markov ẩn câu tổng hợp

3.2.3. Phương pháp tính toán lỗi:

Ước lượng hiệu suất của các hệ thống nhận dạng tiếng nói là một công việc rất quan trọng. Tỷ lệ lỗi nhận dạng từ được sử dụng rộng rãi. Khi so sánh các thuật toán mô hình âm học, điều quan trọng là so sánh sự giảm lỗi tương đối của chúng. Phải kiểm tra một tập dữ liệu gồm hơn 500 câu (với 6 đến 10 từ mỗi câu) của từ 5 đến 10 người nói để ước lượng tỷ lệ lỗi nhận dạng một cách tin cậy. Thông thường, một thuật toán mới được xem là phù hợp khi giảm thiểu được từ 10% lỗi trở lên.

Có 3 loại lỗi nhận dạng từ điển hình trong nhận dạng tiếng nói:

- Thay từ (Substitution): một từ không đúng thay thế cho một từ đúng.

- Xóa từ (Deletion): một từ đúng bị loại bỏ trong câu nhận dạng.

- Thêm từ (Insertion): một từ khác được thêm vào trong câu nhận dạng.

Để xác định tỷ lệ lỗi nhỏ nhất, sử dụng công thức sau:

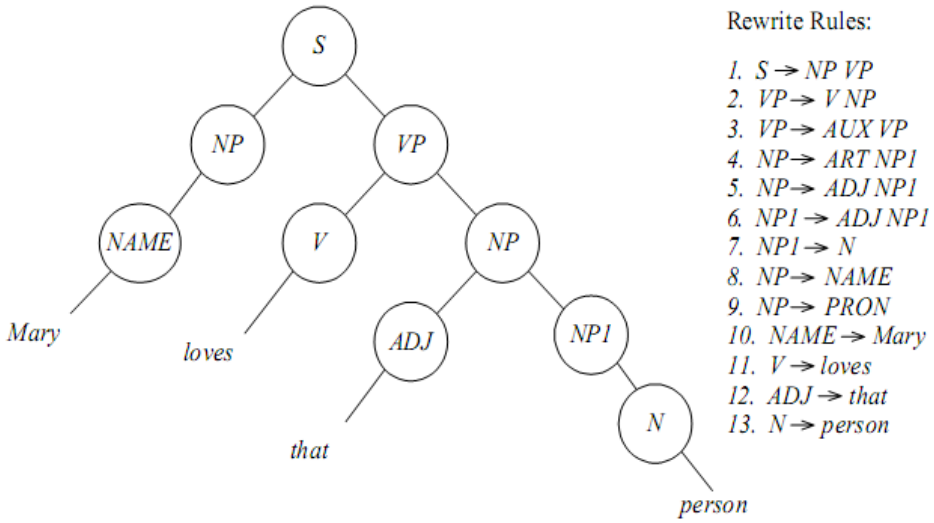
$$WER = 100 \times \frac{Subs+Defs+Ins}{NOW} \quad (3.26)$$

3.3. MÔ HÌNH NGÔN NGỮ:

Quá trình so khớp mẫu âm học và kiến thức về ngôn ngữ là quan trọng như nhau trong nhận dạng và hiểu tiếng nói tự nhiên. Trong nhận dạng tiếng nói thực tiễn, có thể không có khả năng để tách việc sử dụng của các cấp độ khác nhau của tri thức, vì thế chúng thường được tích hợp chặt chẽ. Chúng ta sẽ tìm hiểu về lý thuyết ngôn ngữ hình thức và xác suất mô hình ngôn ngữ. Ngôn ngữ hình thức có 2 phần cơ bản: ngữ pháp và thuật toán phân tích cú pháp. Ngữ pháp là sự mô tả hình thức của cấu trúc được cho phép đối với ngôn ngữ. Kỹ thuật phân tích cú pháp là phương pháp phân tích câu để thấy nếu cấu trúc của nó là tuân theo ngữ pháp. Với sự có mặt của khối lượng nhiều văn bản mà cấu trúc của nó được chú thích bằng tay. Mối quan hệ xác suất giữa dãy các từ có thể được dẫn suất trực tiếp và được mô hình từ tập văn bản được gọi là mô hình ngôn ngữ Stochastic, như n-gram. Mô hình ngôn ngữ Stochastic đóng vai trò thiết yếu trong xây dựng hoạt động một hệ thống ngôn ngữ nói.

3.3.1. Lý thuyết ngôn ngữ hình thức:

Cách phổ biến nhất biểu diễn câu: *"Mary loves that person"*, là sử dụng cây như minh họa trong hình (3.12). Nút *S* là nút cha của nút *NP* và *VP* cho cụm danh từ và cụm động từ một cách mong đợi. Nút *VP* là nút cha của nút *V* và *N* cho động từ và danh từ. Mỗi nút lá được liên kết với từ trong câu để được phân tích. Để xây dựng một cây cho một câu, chúng ta phải biết cấu trúc của ngôn ngữ. Vì vậy tập các nguyên tắc viết lại có thể được sử dụng để mô tả cấu trúc cây được cho phép. Những luật này, xác định một ký hiệu chắc chắn có thể được mở rộng trong cây bằng một dãy các ký hiệu. Cấu trúc ngữ pháp hỗ trợ trong việc xác định ý nghĩa của câu. Nó cho chúng ta biết *"that"* trong câu chỉ *person*.



Hình 3.12. Một biểu diễn cây của một câu và ngữ pháp tương ứng của nó

3.3.1.1. Hệ thống cấp bậc Chomsky:

Trong lý thuyết ngôn ngữ hình thức Chomsky, một cấu trúc ngữ pháp được định nghĩa *G = (V, T, P, S)*, trong đó *V* và *T* là tập hữu hạn của các ký hiệu cuối và các ký hiệu không phải cuối. *V* bao gồm tất cả các ký hiệu không phải ký hiệu kết thúc. Chúng ta thường sử dụng chữ hoa để biểu thị chúng. Tập thuật ngữ *T* bao gồm *Mary, loves, that, person*, được biểu thị chữ thường. *P* là tập hữu hạn của việc viết lại các quy tắc. *S* là một ký hiệu đặc biệt, được gọi là ký hiệu bắt đầu.

Bảng 3.1. Hệ thống cấp bậc Chomsky và máy tương ứng cho phép ngôn ngữ

Loại	Ràng buộc	Hệ thống tự động
Ngữ pháp cấu trúc	α→β. Đây là ngữ pháp tổng quát nhất.	Máy Turing
Ngữ pháp ngữ cảnh nhạy	Một tập con của ngữ pháp cấu trúc cụm từ |α|≤|β|, trong đó |.| cho biết chiều dài của chuỗi.	Hệ thống tự động tuyến tính
Ngữ pháp ngữ cảnh tự do (CFG - context free grammar)	Một tập con của ngữ pháp ngữ cảnh nhạy. Quy tắc tạo ra là A → β, trong đó A là không phải là ký hiệu kết thúc. Hình thức Chomsky: A→ *w* và A→BC, trong đó *w* là một ký hiệu kết thúc và B, C không phải.	Hệ thống tự động thúc đẩy
Ngữ pháp thông thường	Một tập con của CFG. Qui tắc tạo ra được mô tả: A→*w* và A→*w*B.	Hệ thống tự động trạng thái hữu hạn

Ngôn ngữ được phân tích về cơ bản là một chuỗi các ký hiệu thuật ngữ, như *"Mary loves that person"*. Nó được tạo ra bằng cách áp dụng qui tắc tạo ra theo chuỗi từ ký hiệu bắt đầu. Qui tắc tạo ra dạng α→β, trong đó α và β là các chuỗi tùy

ý của ký hiệu ngữ pháp *V* và *T*. Và α phải không được rỗng. Trong lý thuyết ngôn ngữ hình thức, 4 ngôn ngữ chính và ngữ pháp liên kết của chúng được cấu trúc một cách có cấp bậc. Đó là cấp bậc Chomsky như định nghĩa trong bảng trên. Có 4 loại hệ thống tự động mà có thể chấp nhận các ngôn ngữ được tạo bởi bốn loại cấu trúc ngữ pháp này. Giữa những hệ thống này, hệ thống tự động trạng thái hữu hạn không chỉ là hệ thống toán học được sử dụng để trang bị ngữ pháp thông thường mà còn là một trong những công cụ đáng kể trong ngôn ngữ tính toán. Sự đa dạng của hệ thống tự động như bộ chuyển đổi trạng thái hữu hạn, mô hình Markov ẩn và mô hình n-gram là những phần quan trọng trong xử lý ngôn ngữ nói.

3.3.1.2. Phân tích cú pháp đồ thị cho ngữ pháp ngữ cảnh tự do (CFG - Context Free Grammars):

Thuật toán này được sử dụng rộng rãi trong các hệ thống hiểu ngôn ngữ nói tiên tiến.

- ***Từ trên xuống hay từ dưới lên:***

Phân tích cú pháp là một trường hợp đặc biệt của vấn đề tìm kiếm một cách tổng quát bắt gặp trong nhận dạng tiếng nói. Thủ tục tìm kiếm có thể bắt đầu từ nút gốc của cây với ký hiệu *S*, ngoài ra có thể bắt đầu từ các từ trong câu nhập vào và xác định một câu từ sao cho khớp một vài ký hiệu không kết thúc. Thủ tục từ dưới lên có thể được nhắc lại với các ký hiệu phân tích từng phần cho đến khi nút gốc của cây hoặc ký hiệu bắt đầu *S* được xác định.

Một phương pháp từ trên xuống bắt đầu với *S*, sau đó tìm kiếm thông qua những cách khác để viết lại các ký hiệu cho đến khi câu nhập vào được tạo ra, hay đến khi tất cả xác suất được kiểm tra. Một dạng ngữ pháp được nói đến chấp nhận một câu nếu có một câu của các qui tắc cho phép chúng ta viết lại ký hiệu bắt đầu trong câu. Một câu của các qui tắc viết lại có thể được minh họa như sau:

S

→NP VP (viết lại S sử dụng s→NP)

→ NAME VP (viết lại NP sử dụng NP→NAME)

→ Mary VP (viết lại NAME sử dụng NAME→Mary)

…

→ Mary loves that person (viết lại N sử dụng N→person)

Một cách thay đổi, chúng ta có thể lấy cách tiếp cận từ dưới lên để bắt đầu với những từ trong câu nhập vào và sử dụng qui tắc ghi lại phía sau để giảm câu của các ký hiệu cho đến khi nó trở thành S. Phía bên tay trái hay mỗi qui tắc được sử dụng để ghi lại ký hiệu trên phía bên tay phải như sau:

→ NAME loves that person (ghi lại *Mary* sử dụng *NAME →Mary*)

→ NAME V that person (ghi lại *loves* sử dụng *B →loves*)

…

→ NP VP (ghi lại NP sử dụng S→NP VP)

→ S

Một thuật toán phân tích cú pháp phải được tổ chức có hệ thống với mọi trạng thái có khả năng mà biểu diễn nút trung gian trên cây phân tích cú pháp. Nếu có một lỗi xảy ra sớm trong việc lựa chọn quo tắc ghi lại *S*, các kết quả phân tích cú pháp trung gian có thể rất lãng phí nếu số các qui tắc trở nên càng lớn.

- ***Phân tích cú pháp đồ thị từ dưới lên:***

Thuật toán:

Bước 1: khởi tạo: định nghĩa một danh sách được gọi là biểu đồ để lưu trữ các hình cung và một danh sách đã gọi danh sách một số vấn đề để lưu các thành phần cho đến khi chúng được thêm vào biểu đồ.

Bước 2: lặp lại từ bước 2 đến bước 7 đến khi không còn dữ liệu đầu vào.

Bước 3: thêm vào và lấy ra từ danh sách: nếu danh sách là rỗng, tìm kiếm từ tiếp theo trong đầu vào và thêm chúng vào danh sách. Lấy thành phần *C* từ danh sách. Nếu *C* tương ứng với vị trí từ w_i đến w_j của câu nhập vào, chúng ta chi biểu thị nó *C[i, j]*.

Bước 4: thêm *C[i, j]* vào biểu đồ.

Bước 5: thêm hình cung đánh dấu vào biểu đồ. Với mỗi qui tắc trong ngữ pháp dạng $X \rightarrow C\ Y$, them vào đồ thị một hình cung dạng $X[i, j] \rightarrow {}^{o}CY$, trong đó ^{o} cho biết vị trí quan trọng gọi là khóa biểu thị mọi thứ trước ^{o} có thể nhìn thấy, nhưng sau ^{o} chưa được liên kết.

Bước 6: di chuyển ^{o} qua: cho bất cứ hình cung có hiệu lực của dạng $X[1,j] \rightarrow Y...{}^{o}C...Z$ (trước w_i) trong đồ thị, them một hình cung mới có dạng $X[1,j] \rightarrow Y...{}^{o}C...Z$ vào đồ thị.

Bước 7: thêm thành phần mới vào danh sách: với mỗi hình cung dạng $X[1,j] \rightarrow Y...\#C$ thêm thành phần mới vào danh sách *X[1,j]*.

Bước 8: kết thúc: nếu S[1,n] ở trong đồ thị, trong đó n là chiều dài của câu nhập vào, chúng ta có thể thoát một cách thành công nếu chúng ta không muốn

tìm tất cả cách hiểu có khả năng của câu. Đồ thị này có thể bao gồm nhiều cấu trúc S chứa toàn bộ tập các vị trí.

3.3.2. Mô hình ngôn ngữ Stochastic:

Mô hình ngôn ngữ Stochastic (SLM-stochastic language model) lấy một xác suất điểm nhìn của việc mô hình hóa ngôn ngữ. Chúng ta cần ước lượng chính xác xác suất $P(\mathbf{W})$ với một dãy từ cho trước $\mathbf{W} = w_1 w_2 ... w_n$. Trong lý thuyết ngôn ngữ hình thức $P(\mathbf{W})$ có thể được quan tâm như 1 hoặc 0 nếu dãy từ là được chấp nhận hoặc từ chối, một cách mong đợi theo cấu trúc ngữ pháp. Điều này có thể không phù hợp cho các hệ thống ngôn ngữ nói, khi ngữ pháp không thể có một mức độ hoàn toàn, không đề cập đến là ngôn ngữ nói thường không theo cấu trúc ngữ pháp trong các ứng dụng đàm thoại thực tế.

Mục tiêu chính của SLM là cung cấp đầy đủ thông tin xác suất vì thế các dãy từ có khả năng nên có một xác suất cao hơn. Nó không chỉ làm cho quá trình nhận dạng tiếng nói thêm chính xác mà còn hỗ trợ ràng buộc không gian tìm kiếm cho nhận dạng tiếng nói. Chú ý là SLM có thể có một bao phủ rộng trên tất cả các dãy từ có khả năng, khi xác suất được dùng để phân biệt những dãy từ khác nhau. Sử dụng rộng rãi nhất của SLM là mô hình n-gram. CFG có thể được tăng cường như là cầu nối giữa n-gram và ngữ pháp hình thức nếu chúng ta có thể kết hợp các xác suất trong các qui tắc quá trình tạo ra.

3.3.2.1. Xác suất ngữ pháp ngữ cảnh tự do (CFG):

CFG có thể được tăng lên với xác suất cho mỗi qui tắc tạo mới. Thuận lợi của các xác suất CFG trên khả năng của chúng để bắt nhiều độ chính xác hơn trong cấu trúc sử dụng đã nhúng vào của ngôn ngữ nói để cực tiểu sự nhập nhằng cú pháp. Việc sử dụng xác suất trở nên gia tăng quan trọng để phân biệt nhiều lựa chọn cạnh tranh khi số các qui tắc là lớn.

Vấn đề nhận dạng được quan tâm với quá trình tính toán xác suất của ký hiệu bắt đầu S tạo ra dãy từ $\mathbf{W} = w_1, w_2, ... w_T$, cho trước bộ ngữ pháp G:

$$P(S \rightarrow \mathbf{W}|G) \tag{3.27}$$

Vấn đề huấn luyện được quan tâm với việc xác định tập các qui tắc trong G trên cơ sở tập văn huấn luyện và quá trình ước lượng xác suất của mỗi luật. Nếu tập

các luật là cố định, phương pháp đơn giản nhất để dẫn suất các xác suất này là đếm số lần mỗi luật được sử dụng trong tập văn bao gồm những câu đã được phân tích cú pháp. Chúng ta biểu thị xác suất của một luật A→α bởi P(A→α|G). Nếu có m luật bên tay trái không phải nút cuối $A{:}A\rightarrow\alpha_1$, $A\rightarrow\alpha_2$, ...$A\rightarrow\alpha_m$, chúng ta có thể lượng giá xác suất các luật này như sau:

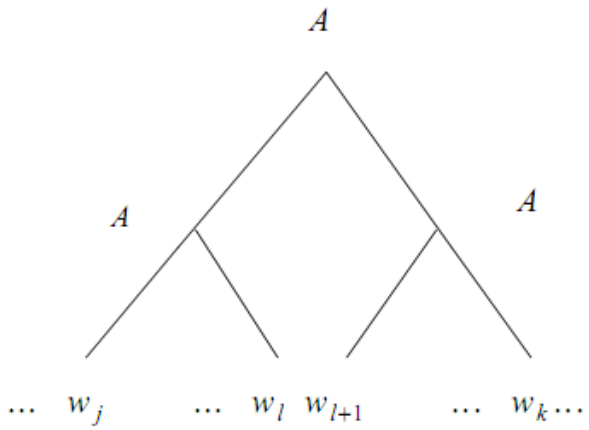
$$P\left(A \rightarrow \propto_j \middle| G\right) = C\left(A \rightarrow \frac{\propto_j}{\sum_{i=1}^{m} C(A \rightarrow \propto_i)}\right) \tag{3.28}$$

Chúng ta để cho dãy từ **W**=w_1, w_2, ...w_T được tạo bởi xác suất CFG G, với các qui tắc Chomsky:

$$A_i \rightarrow A_m A_n \text{ và } A_i \rightarrow w_l \tag{3.29}$$

Trong đó A_m và A_n không có khả năng là nút cuối mà mở rộng A_i ở vị trí khác. Xác suất cho các qui tắc này phải thỏa mãn ràng buộc sau:

$$\sum_{m,n} P(A_i \rightarrow A_m A_n | G) + \sum_l P(A_i \rightarrow w_l | G) = 1\,, \quad \text{đối với tất cả } i \tag{3.30}$$



Hình 3.13. Xác suất bên trong được tính toán một cách đệ quy như tổng của tất cả các dẫn suất

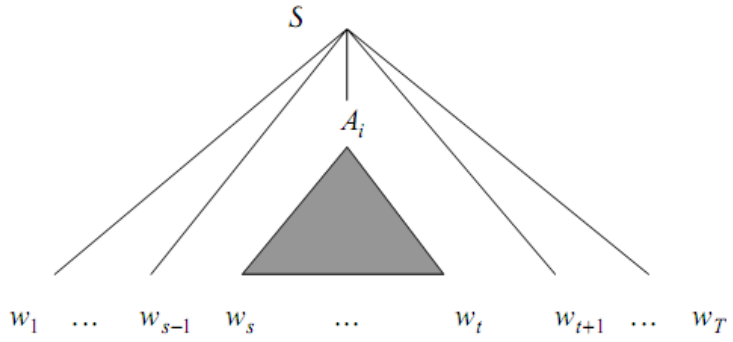
Xác suất bên trong:

$$inside(j, A_i, k) = P(A_i \rightarrow w_j w_{j+1} \ldots w_k | G) \tag{3.31}$$

Như xác suất cấu thành bên trong, nó hỗ trợ một xác suất cho một dãy từ bên trong quá trình tạo thành.

Ngoài ra còn có xác suất bên ngoài cho nút không phải là nút cuối A_i bao gồm w_s đến w_t , trong đó chúng có thể được dẫn suất từ ký hiệu bắt đầu S, cùng với phần còn lại của các từ trong câu:

$$outside(s, A_i, t) = P(S \rightarrow w_1 \ldots w_{s-1} A_i\, w_{t+1} \ldots w_T) \tag{3.32}$$



Hình 3.14. Định nghĩa xác suất bên ngoài

Xác suất bên trong và bên ngoài được sử dụng để tính toán xác suất câu:

$$P(S \rightarrow w_1 \dots w_T) = \sum_i inside(s, A_i, t) outside(s, A_i, t), \text{đối với mọi } s \leq t \quad (3.33)$$

Một vấn đề với xác suất CFG là nó giả định sự mở rộng bất kỳ nút không phải nút cuối là độc lập với sự mở rộng các nút khác. Vì vậy mỗi xác suất luật CFG được nhân với nhau mà không cần xem xét vị trí của nút trong cây phân tích cú pháp. Một vấn đề khác là sự thiếu nhay bén với các từ, mặc dù thong tin bộ từ vựng đóng vai trò quan trọng trong việc lựa chọn chính xác quá trình phân tích cú pháp của cụm từ nhập nhằng. Trong xác suất CFG, thông tin bộ từ vựng có thể chỉ được biểu diễn thông qua xác suất của các nút xuất hiện trước nút cuối, như động từ và danh từ, để được mở rộng theo bộ từ vựng. Ta có thể thêm các ràng buộc từ vựng cho xác suất CFG và tạo ra các xác suất CFG nhạy hơn trong cấu trúc cú pháp.

3.3.2.2. Mô hình ngôn ngữ n-gram:

Một mô hình ngôn ngữ có thể được lập công thức như một phân phối xác suất $P(\mathbf{W})$ thông qua các chuỗi từ **W** phản ánh làm thế nào một chuỗi **W** tìm thấy như một câu.

$$P(W) = \prod_{i=1}^{n} P(w_i | w_1, w_2, \dots, w_{i-1}) \quad (3.34)$$

Trong đó $P(w_i | w_1, w_2, \dots, w_{i-1})$ là xác suất mà w_i sẽ theo, cho trước dãy từ $w_1, w_2, \dots, w_{i-1}$. Lựa chọn w_i phụ thuộc vào toàn bộ lịch sử đầu vào. Cho một bộ từ vựng kích thước v, để xác định $(w_i | w_1, w_2, \dots, w_{i-1})$, giá trị v^i phải được lượng giá. Trong thực tế $P(w_i | w_1, w_2, \dots, w_{i-1})$ là không thể lượng giá cho giá trị trung hòa của i, khi hầu hết mẫu $w_1, w_2, \dots, w_{i-1}$ là duy nhất hay chỉ xảy ra tại một vài thời điểm. Một giải pháp thực tế cho vấn đề trên là giả định $P(w_i | w_1, w_2, \dots, w_{i-1})$ phụ thuộc một vài lớp tương đương. Lớp tương đương dựa trên cơ sở nhiều từ trước đó $w_{i-N+1}, w_{i-N+2}, \dots, w_{i-1}$. Điều này dẫn đến một mô hình ngôn ngữ n-gram. Nếu từ

phụ thuộc vào hai từ trước đó, chúng ta có mô hình ngôn ngữ trigram: $(w_i|w_{i-2}, w_{i-1})$. Tương tự ta có mô hình ngôn ngữ unigram: $P(w_i)$, hay bigram: $P(w_i|w_{i-1})$. Mô hình trigram rất mạnh, nó có thể lượng giá hợp lý với tập văn có thể đạt được.

3.3.3. Độ phức tạp của các mô hình ngôn ngữ:

Ta có hiệu suất entropy $H(\mathbf{W})$ của một mô hình $P(w_i|w_{i-n+1}, w_{i-n+2}, \dots, w_{i-1})$ trên dữ liệu **W**, với một dãy từ dài thích đáng, có thể được ước lượng:

$$H(\mathbf{W}) = -\frac{1}{N_w}\log_2 P(\mathbf{W}) \tag{3.35}$$

Trong đó N_w là chiều dài văn bản **W** được đo trong bộ từ.

Độ phức tạp $PP(\mathbf{W})$ của mô hình ngôn ngữ $P(\mathbf{W})$ được định nghĩa như hàm nghịch của xác suất trung bình được phân phối bởi mô hình cho mỗi từ trong tập kiểm tra **W**:

$$PP(\mathbf{W}) = 2^{H(\mathbf{W})} \tag{3.36}$$

Độ phức tạp có thể được dịch như nghĩa hình học của phân nhánh hệ số của văn bản khi được biểu diễn cho mô hình ngôn ngữ. Độ phức tạp định nghĩa trong (3.37) có hai đối số chính: mô hình ngôn ngữ và một dãy từ. Độ phức tạp tập kiểm tra lượng giá khả năng tạo ra mô hình ngôn ngữ. Độ phức tạp tập huấn luyện đo mô hình ngôn ngữ phù hợp với dữ liệu huấn luyện ra sao, giống như khả năng có thể xảy ra. Nó tổng quát là đúng, độ phức tạp thấp hơn tương liên với hiệu suất nhận dạng tốt hơn. Đó là bởi vì độ phức tạp được đo theo sự phân nhánh từ được tính một cách thống kê trên tập kiểm tra. Độ phức tạp cao hơn, nhiều nhánh hơn của trình nhận dạng tiếng nói cần được quan tâm một cách thống kê.

Khi độ phức tạp không đưa vào sự tính toán sự phức tạp âm học, ta cuối cùng có thể đo độ chính sát quá trình nhận dạng tiếng nói. Ví dụ, nếu bộ từ vựng của trình nhận dạng tiếng nói bao gồm tập E của các chữ cái: *B, C, D, E, G* và *T*, chúng ta có thể định nghĩa một CFG với giá đột phức tạp thấp hơn là 6. Độ phức tạp thấp hơn như vậy không đảm bảo chúng ta sẽ có hiệu suất quá trình nhận dạng tốt, bởi vì sự phức tạp âm học bên trong của tập E.

CHƯƠNG 4: CÔNG CỤ HỖ TRỢ NHẬN DẠNG TIẾNG NÓI

4.1. GIỚI THIỆU VỀ SPHINX:

Hiện nay có 2 bộ công cụ hỗ trợ nhận dạng tiếng nói tiếng Việt là HTK và Sphinx. Tuy nhiên, vẫn chưa có công trình nghiên cứu nào để khẳng định công cụ nào là tốt nhất. Luận văn này sẽ sử dụng Sphinx để làm công cụ nhận dạng tiếng nói tiếng Việt.

Sphinx là một hệ thống nhận dạng tiếng nói được viết bằng ngôn ngữ Java. Nó được tạo ra bởi sự tham gia cộng tác giữa nhóm Sphinx của CMU (Carnegie Mellon University), Sun Microsystems Laboratories, MERL (Mitsubishi Electric Research Labs) và HP (Hewlett Packard), với sự đóng góp của UCSC (University of California at Santa Cruz) và MIT (Massachusetts Institute of Technology).

Các tính năng chính:

- Nhận dạng tiếng nói ở chế độ trực tiếp và theo lô, có khả năng nhận dạng tiếng nói rời rạc và liên tục.

- Kiến trúc ngoại vi tổng quát có khả năng tháo lắp. Bao gồm khả năng bổ sung các tính năng tiền nhấn (preemphasis), cửa sổ Hamming, biến đổi Fourier nhanh, thang lọc tần số Mel, biến đổi cosine rời rạc, chuẩn hóa cepstral, và trích đặc trưng cepstra, delta cepstra, double delta cepstra.

- Kiến trúc mô hình ngôn ngữ tổng quát và có khả năng tháo lắp. Bao gồm hỗ trợ mô hình ngôn ngữ dạng ASCII và các phiên bản nhị phân của unigram, bigram, trigram, Java Speech API Grammar Format (JSGF), và ARPA-format FST grammars.

- Kiến trúc mô hình âm tổng quát. Bao gồm hỗ trợ các mô hình âm học của Sphinx3.

- Bộ quản lý tìm kiếm tổng quát. Bao gồm hỗ trợ các tìm kiếm breadth first và word pruning.

- Các tiện ích cho việc xử lý kết quả sau khi nhận dạng, bao gồm tính điểm số tin cậy, phát sinh các lưới và nhúng kịch bản ECMA vào thẻ JSGF. Các công cụ

độc lập bao gồm các công cụ để hiển thị dạng sóng và ảnh phổ và trích đặc trưng từ tập tin âm thanh.

Sphinx đã trở thành một framework nhận dạng tiếng nói mạnh mẽ, đã được sử dụng trong nhiều hệ thống nhận dạng bao gồm các chương trình điện đàm như Cairo, Freeswitch, jvoicexml,… các chương trình điều khiển như Gnome-Voice-Control, Voicekey, SpeechLion,…

Các lợi ích khi sử dụng Sphinx:

Đối với việc nghiên cứu nhận dạng tiếng nói dựa trên mô hình Markov ẩn:

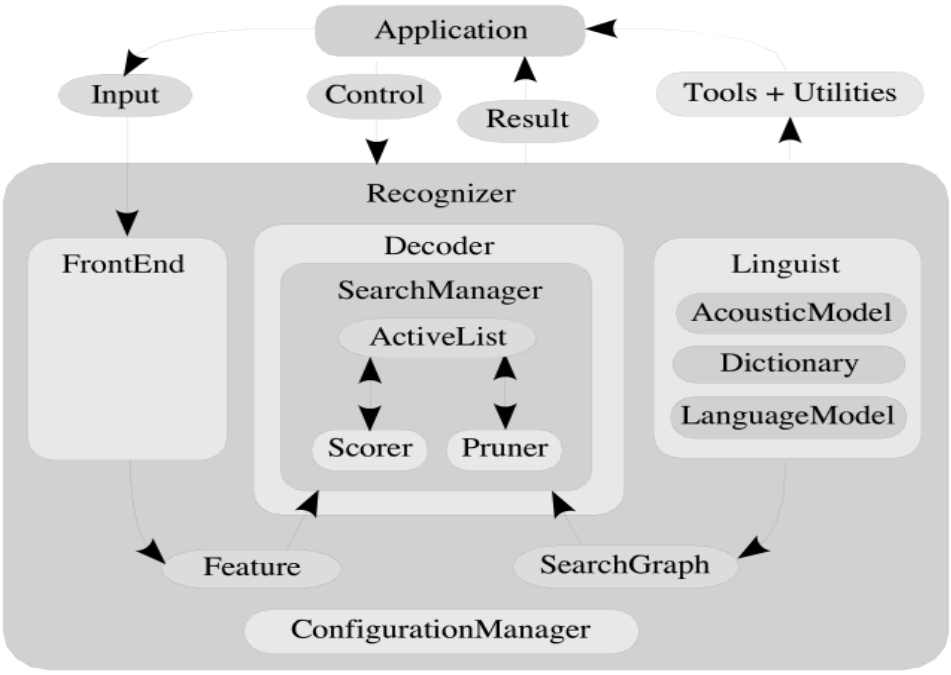
- Sphinx mặc định rằng việc tính toán Gaussian Mixture Model và xử lý tìm kiếm là tách biệt nên có thể thực hiện hai loại nghiên cứu khác nhau mà không bị xung đột với nhau. Ví dụ có thể thực hiện một sự quan sát xác suất mới mà không đụng đến mã nguồn thực hiện tìm kiếm. Cùng lúc có thể xây dựng một thuật toán tìm kiếm mới mà không phải suy nghĩ về tính toán GMM,...

- Đối với việc huấn luyện, hầu hết thời gian khi ta nghiên cứu mô hình hóa, điều mong muốn thay đổi là thuật toán ước lượng. Thuật toán Baunm-Welch của SphinxTrain giải quyết vấn đề này qua hai giai đoạn: Đưa thống kê xác suất đến sau ra một tập tin riêng biệt và có thể dễ dàng đọc lại bằng các thư viện của SphinxTrain. Bạn có thể chỉ làm việc với các bản thống kê này và không cần phải tự mình thực hiện huấn luyện Baum-Welch. Điều này giúp giảm thời gian nghiên cứu.

- Mã nguồn của Sphinx được viết rõ ràng và dễ đọc. Nhiều nhà nghiên cứu không chỉ muốn sử dụng Sphinx như một công cụ mà còn muốn thay đổi mã nguồn để phù hợp với mục đích của họ.

4.2. KIẾN TRÚC SPHINX:

Sphinx Framework được thiết kế với độ linh hoạt và tính mô đun hóa cao. Hình dưới đây biểu diễn bao quát kiến trúc của hệ thống. Mỗi thành phần được gán nhãn biểu diễn một mô đun có thể dễ dàng được thay thế, cho phép các nhà nghiên cứu thử nghiệm một mô đun khác mà không cần phải thay đổi các phần còn lại của hệ thống.



Hình 4.1. Kiến trúc tổng quát của Sphinx

Có 3 mô đun chính trong Sphinx Framework: Bộ ngoại vi (FrontEnd), Bộ giải mã (Decoder) và bộ ngôn ngữ (Linguist). Bộ ngoại vi nhận vào một hay nhiều tín hiệu số và tham số hóa chúng thành một dãy các đặc trưng (Feature). Bộ ngôn ngữ chuyển đổi tất cả các mô hình ngôn ngữ chuẩn, cùng với thông tin cách phát âm trong từ điển (Dictionary) và thông tin cấu trúc từ một hay nhiều các tập hợp các mô hình âm học (AcousticModel) vào một đồ thị tìm kiếm (SearchGraph). Bộ quản lý tìm kiếm (SearchManager) trong bộ giải mã sử dụng các đặc trưng từ bộ ngoại vi và đồ thị tìm kiếm từ bộ ngôn ngữ để thực hiện việc giải mã, phát sinh các kết quả (Result). Tại bất kỳ thời điểm trước và trong quá trình xử lý nhận dạng, ứng dụng (Application) đưa ra các điều khiển tới mỗi mô đun, trở thành một đối tác hiệu quả trong quá trình xử lý nhận dạng.

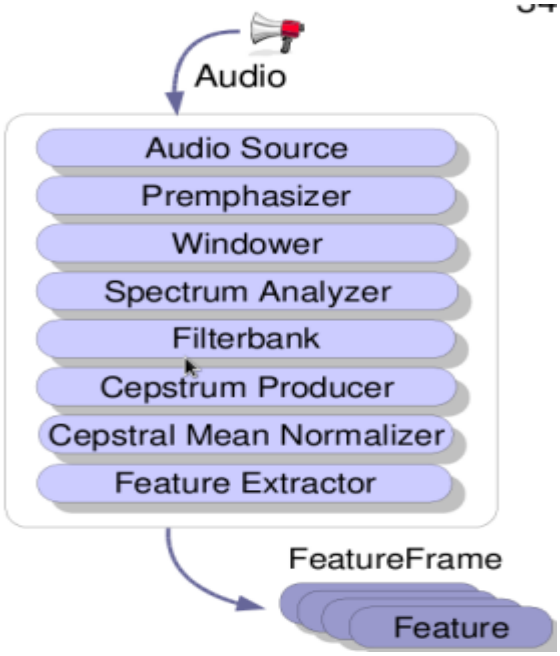
Hệ thống Sphinx4 có một số lượng lớn các tham số cấu hình để điều chỉnh hiệu suất của hệ thống. Thành phần quản lý cấu hình (ConfigurationManager) được dùng để cấu hình các tham số đó. Bộ quản lý cấu hình còn giúp cho Sphinx4 có khả năng nạp động và cấu hình các mô đun trong thời gian thực thi, làm cho Sphinx4 trở nên linh hoạt và có khả năng tháo lắp. Ví dụ Sphinx4 thường được cấu hình với một bộ ngoại vi tạo ra các MFCC (Mel-Frequency Cepstral Coefficient). Sử dụng bộ quản lý cấu hình có khả năng cấu hình lại Sphinx4 để xây dựng một ngoại vi khác phát sinh ra các PLP (Perceptual Linear Prediction coefficient) mà không cần phải

sửa đổi mã nguồn hay biên dịch lại hệ thống.

Sphinx còn cung cấp một số công cụ để giúp các ứng dụng và các nhà phát triển khả năng theo dõi các số liệu thống kê của bộ giải mã như tỷ lệ từ lỗi, tốc độ thực thi và bộ nhớ sử dụng. Cũng như các phần khác của hệ thống, các công cụ này có khả năng cấu hình mạnh mẽ, cho phép người dùng thực hiện việc phân tích hệ thống. Hơn nữa chúng còn cung cấp một môi trường thực thi tương tác, cho phép người dùng sửa đổi các tham số của hệ thống trong lúc hệ thống đang chạy, giúp cho việc thử nghiệm nhanh chóng với nhiều tham số cấu hình.

Sphinx4 cũng cung cấp các tiện ích hỗ trợ xem xét cấp độ ứng dụng (application-level) của các kết quả nhận dạng. Ví dụ, các tiện ích này bao gồm hỗ trợ xem kết quả thu được dạng lưới (lattice), các đánh giá độ tin cậy (confidence scores) và sự hiểu ngôn ngữ.

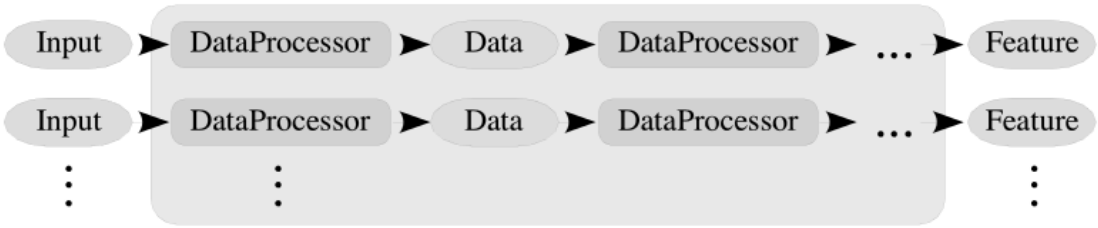
4.2.1. Bộ ngoại vi - FrontEnd:



Hình 4.2. Quá trình trích đặc trưng của bộ ngoại vi dùng MFCC

Mục đích của bộ ngoại vi là tham số hóa một tín hiệu đầu vào (âm thanh) thành một dãy các đặc trưng xuất ra. Bộ ngoại vi bao gồm một hay nhiều chuỗi song song các mô đun xử lý tín hiệu giao tiếp có khả năng thay thế gọi là các DataProcessor. Việc hỗ trợ nhiều chuỗi cho phép giả lập tính toán các loại tham số khác nhau trong cùng một hay nhiều tín hiệu vào. Điều này cho phép tạo nên các hệ thống có thể giải mã cùng một lúc sử dụng các loại tham số khác nhau, ví dụ MFCC

và PLP. và thậm chí các loại tham số dẫn xuất từ các tín hiệu không phải là tín hiệu tiếng nói như video.



Hình 4.3. Chuỗi các DataProcessor

Mỗi DataProcessor trong bộ ngoại vi hỗ trợ một đầu vào và một đầu ra có thể được kết nối với DataProcessor khác, cho phép tạo thành dãy các chuỗi dài chuyên biệt. Sphinx4 cho phép khả năng phát sinh dãy các đặc trưng song song và cho phép một số lượng tùy ý các dòng song song.

Sử dụng ConfigurationManager, người dùng có thể xâu chuỗi các DataProcessor vói nhau theo bất kỳ cách nào cũng như các bổ sung DataProcessor kết hợp chặt chẽ trong thiết kế riêng của họ.

4.2.2. Bộ ngôn ngữ - Linguist:

Bộ ngôn ngữ phát sinh đồ thị tìm kiếm (SearchGraph) để sử dụng trong bộ giải mã trong quá trình tìm kiếm, trong khi ẩn đi các phần phức tạp bao gồm phát sinh ra đồ thị này. Trong Sphinx4 bộ ngôn ngữ là một module có thể gắn thêm, cho phép người dùng có thể cấu hình động hệ thống với các cài đặt khác vào bộ ngôn ngữ.

Một bổ sung bộ ngôn ngữ thông thường xây dựng nên đồ thị tìm kiếm sử dụng cấu trúc ngôn ngữ được mô tả bởi LanguageModel cho trước và cấu trúc hình học tôpô của mô hình ngôn ngữ (các HMM cho các đơn vị âm cơ bản sử dụng bởi hệ thống). Bộ ngôn ngữ có thể cũng sử dụng một từ điển (thường là một từ điển phát âm) để ánh xạ các từ từ mô hình ngôn ngữ vào các chuỗi của các thành phần mô hình âm học. Khi phát sinh đồ thị tìm kiếm, bộ ngôn ngữ có thể còn kết hợp các đơn vị từ con (subword) với các ngữ cảnh độ dài tùy ý, nếu được cung cấp.

Bằng cách cho phép các bổ sung khác nhau của bộ ngôn ngữ được gắn kết vào trong thời gian chạy, Sphinx4 cho phép các cá nhân cung cấp các cấu hình khác nhau cho các hệ thống và các yêu cầu nhận dạng khác nhau. Ví dụ: một ứng dụng

nhận dạng các số đơn giản có thể sử dụng một bộ ngôn ngữ đơn giản để có thể lưu toàn bộ không gian tìm kiếm trong bộ nhớ. Mặt khác, một ứng dụng đọc chính tả với 100 ngàn từ vựng có thể dùng một bộ ngôn ngữ phức tạp để chỉ lưu một phần nhỏ của không gian tìm kiếm tiềm năng trong bộ nhớ trong một thời gian.

Bộ bộ ngôn ngữ bao gồm 3 thành phần: mô hình ngôn ngữ, từ điển, và mô hình âm học.

4.2.2.1. Mô hình ngôn ngữ:

Mô hình ngôn ngữ của bộ ngôn ngữ cung cấp cấu trúc ngôn ngữ cấp độ từ (word-level), có thể biểu diễn bởi bất cứ số lượng các bổ sung có thể gắn thêm. Những bổ sung này thường là một trong hai mục: các graph-driven grammar và các mô hình Stochastic N-Gram. Các Graph-driven grammar biểu diễn một đồ thị từ có hướng trong đó mỗi nút biểu diễn một từ đơn và mỗi cung biểu diễn xác suất dịch chuyển sang một từ. Các mô hình stochastic N-Gram cung cấp các xác suất cho các từ được cho dựa vào việc quan sát n-1 từ đứng trước.

Mô hình ngôn ngữ của Sphinx4 hỗ trợ nhiều định dạng khác nhau bao gồm:

- SimpleWordListGrammar: định nghĩa một từ dựa trên một danh sách các từ. Một tham số tùy chọn chỉ ra ngữ pháp có lặp hay không. Nếu ngữ pháp không lặp, ngữ pháp sẽ được dùng cho một nhận dạng từ tách biệt. Nếu ngữ pháp lặp, nó sẽ được dùng để hỗ trợ liên kết nhận dạng từ tầm thường, tương đương với một unigram grammar với xác suất bằng nhau.

- JSGFGrammar: Hỗ trợ JavaTM Speech API Grammar Format (JSGF), định nghĩa một biểu diễn theo BNF, độc lập nền tảng, Unicode của các ngữ pháp.

- LMGrammar: định nghĩa một ngữ pháp dựa trên một mô hình ngôn ngữ thống kê. LMGrammar phát sinh một nút ngữ pháp mỗi từ và làm việc tốt với các unigram và bigram, xấp xỉ 1000 từ.

- FSTGrammar: hỗ trợ một bộ chuyển đổi trạng thái giới hạn (finite-state tranducer) trong định dạng ngữ pháp ARPA FST.

- SimpleNGramModel: cung cấp hỗ trợ cho các mô hình ASCII N-Gram trong định dạng ARPA. SimpleNGramModel không cố làm tối ưu việc sử dụng bộ nhớ, do đó nó làm việc tốt với các mô hình ngôn ngữ nhỏ.

- LargeTrigramModel: cung cấp hỗ trợ các mô hình N-Gram đúng được phát sinh bởi CMU-Cambridge Statictical Language Modeling Toolkit. LargeTrigramModel tối ưu việc lưu trữ bộ nhớ, cho phép nó làm việc với các tập tin rất lớn, trên 100MB.

4.2.2.2. Từ điển:

Bộ từ điển cung cấp cách phát âm cho các từ tìm thấy trong mô hình ngôn ngữ. Các cách phát âm chia các từ thành các đơn vị từ phụ (sub-word) tìm được trong mô hình âm học. Bộ từ điển cũng hỗ trợ việc phân lớp các từ và cho phép một từ đơn ở trong nhiều lớp.

4.2.2.3. Mô hình âm học:

Mô hình âm học cung cấp một ánh xạ giữa một đơn vị tiếng nói và một HMM có thể được đánh giá dựa vào các đặc trưng được cung cấp bởi bộ ngoại vi. Các ánh xạ có thể đưa thông tin vị trí của từ và ngữ cảnh vào tài khoản. Ví dụ trong trường hợp các triphone, ngữ cảnh miêu tả các âm vị đơn ở bên trái và bên phải của âm vị đã cho, và vị trí của từ mô tả triphone ở vị trí bắt đầu, ở giữa hay ở cuối của một từ (hay chính nó là một từ). Định nghĩa ngữ cảnh này không bị cố định bởi Sphinx4, Sphinx4 cho phép định nghĩa các mô hình âm học chứa các tha âm vị cũng như các mô hình âm học mà ngữ cảnh của nó không cần phải sát với đơn vị.

Thông thường, bộ ngôn ngữ phân tích mỗi từ trong bộ từ vựng được kích hoạt thành một dãy các đơn vị từ con phụ thuộc ngữ cảnh. Bộ ngôn ngữ sau đó chuyển các đơn vị này và các ngữ cảnh của nó đến mô hình âm học, tìm các đồ thị HMM gắn với các đơn vị đó. Sau đó nó dùng các đồ thị HMM này kết hợp với mô hình âm học để xây dựng nên đồ thị tìm kiếm.

Không giống hầu hết các hệ thống nhận dạng tiếng nói biểu diễn các đồ thị HMM là các cấu trúc cố định trong bộ nhớ, HMM trong Sphinx4 chỉ đơn thuần là một đồ thị có hướng của các đối tượng. Trong đồ thị này, mỗi nút tương ứng với một trạng thái HMM và mỗi cung biểu diễn xác suất biến đổi từ trạng thái này sang trạng thái khác trong HMM. Bằng cách biểu diễn HMM như là các đồ thị có hướng của các đối tượng thay vì một cấu trúc cố định, một bổ sung của mô hình âm học có thể dễ dàng cung cấp các HMM với các dạng hình học tôpô khác. Ví dụ, các giao diện mô hình âm học không giới hạn số lượng trạng thái, số lượng chuyển trạng thái

hay hướng chuyển trạng thái của các HMM. Hơn nữa Sphinx4 cho phép số lượng các trạng thái trong một HMM có thể khác nhau từ một đơn vị tới đơn vị khác trong cùng một mô hình âm học.

Mỗi trạng thái HMM có khả năng phát sinh một đánh giá từ một đặc trưng quan sát. Quy tắc để tính toán điểm số được thực hiện bởi chính trạng thái HMM, do đó che dấu các thực thi của nó đối với phần còn lại của hệ thống, thậm chí cho phép các hàm mật độ xác suất khác nhau được sử dụng trên mối trạng thái HMM. Mô hình âm học cũng cho phép chia sẻ các thành phần khác nhau trên tất cả các cấp độ. Nghĩa là các thành phần tạo nên một trạng thái HMM như các hợp Gaussian (Gaussian mixture), các ma trận biến đổi và các trọng số hỗn hợp (mixture weight) có thể được chia sẽ bởi bất kỳ trạng thái HMM nào.

Như các phần còn lại của Sphinx4, người dùng có thể cấu hình Sphinx4 với các bổ sung khác của mô hình âm học. Sphinx4 hiện cung cấp một thực thi trạng thái HMM đặc biệt có khả năng nạp vào và sử dụng các mô hình âm học sinh ra bởi bộ huấn luyện Sphinx-3.

4.2.2.4. Đồ thị tìm kiếm - SearchGraph:

Mặc dù Sphinx4 có thể được thực thi trong nhiều cách khác nhau và các topology của các không gian tìm kiếm sinh bởi các bộ ngôn ngữ có thể rất đa dạng, các không gian tìm kiếm được mô tả hoàn toàn như một đồ thị tìm kiếm. Đồ thị tìm kiếm là cấu trúc dữ liệu chính sử dụng trong suốt quá trình giải mã.

Đó là một đồ thị có hướng trong đó mỗi nút, gọi là một Trạng thái tìm kiếm (SearchState), biểu diễn một trạng thái phát hay không phát (emitting state hay non-emitting state). Các trạng thái phát có thể được đánh giá dựa trên các đặc trưng âm học vào (incoming acoustic feature) trong khi các trạng thái không phát thông thường được dùng để biểu diễn các cấu trúc ngôn ngữ cấp độ cao như các từ và các âm vị không thể đánh giá trực tiếp dựa trên các đặc trưng đầu vào. Các cung giữa các trạng thái biểu diễn các biến đổi trạng thái có thể, mỗi cung có một xác suất chỉ khả năng biến đổi dọc theo các cung.

Giao diện đồ thị tìm kiếm có mục tiêu cho phép một phạm vi lớn các lựa chọn bổ sung. Thực tế, bộ ngôn nhữ không đặt các ràng buộc cố hữu theo:

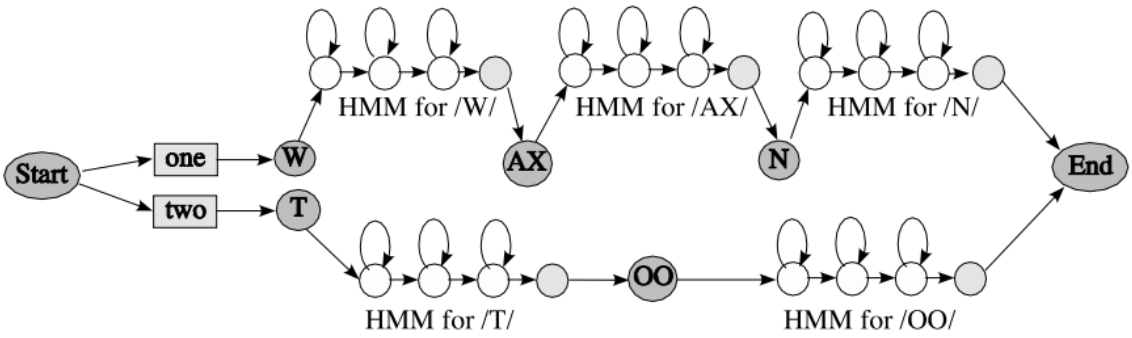
- Toàn bộ topology không gian tìm kiếm.

- Kích thước ngữ cảnh ngữ âm

- Loại của ngữ pháp (theo xác suất hay dựa trên luật)

- Chiều sâu của mô hình ngôn ngữ N-Gram.

Đặc điểm chính của đồ thị tìm kiếm là việc thực thi của trạng thái tìm kiếm không cần cố định. Như vậy, mỗi bổ sung bộ ngôn ngữ thông thường cung cấp thực thi cụ thể của SearchState của riêng nó mà có thể dựa trên các đặc trưng khác nhau của bộ ngôn ngữ cụ thể. Ví dụ, một bộ ngôn ngữ đơn giản có thể cung cấp một đồ thị tìm kiếm trong bộ nhớ mà mỗi trạng thái đơn giản là một ánh xạ một-một lên các nút của đồ thị trong bộ nhớ (in-memory graph). Một bộ ngôn ngữ mô tả một bộ từ vựng rất lớn và phức tạp, tuy nhiên có thể xây dựng một biểu diễn bên trong cô đọng của đồ thị tìm kiếm. Trong trường hợp này, bộ ngôn ngữ sẽ sinh ra một tập trạng thái tìm kiếm kế tiếp bằng cách chủ động mở rộng biểu diễn cô đọng của nó theo nhu cầu.



Hình 4.4. Một ví dụ đồ thị tìm kiếm

Thiết kế theo module của Sphinx cho phép đồ thị tìm kiếm biên dịch các chiến lược khác nhau để sử dụng mà không cần thay đổi các mặt khác của hệ thống. Lựa chọn giữa xây dựng các HMM nhôn ngữ động hay tĩnh phụ thuộc chính vào kích thước từ vựng, độ phức tạp mô hình ngôn ngữ và thỏa mãn yêu cầu bộ nhớ của hệ thống, và có thể thực hiện bởi ứng dụng.

4.2.3. Bộ giải mã - Decoder:

Vai trò chính của bộ giải mã là sử dụng các đặc trưng (Features) từ bộ ngoại vi kết hợp với đồ thị tìm kiếm từ bộ ngôn ngữ để phát sinh các kết quả (Result). Khối bộ giải mã bao gồm một bộ quản lý tìm kiếm (SearchManager) có khả năng

tháo lắp và các mã hỗ trợ khác để đơn giản hóa quá trình giải mã cho một ứng dụng. Do vậy, thành phần đáng quan tâm của bộ giải mã là bộ quản lý tìm kiếm

Bộ giải mã chỉ đơn thuần bảo bộ quản lý tìm kiếm nhận dạng một tập các cấu trúc đặc trưng. Tại mỗi bước xử lý, Bộ quản lý tìm kiếm tạo ra một đối tượng kết quả chứa tất cả đường dẫn đến một trạng thái không phát sinh cuối cùng (final non-emitting state). Để xử lý kết quả, Sphinx cung cấp các tiện ích có khả năng phát sinh một lưới và các đánh giá tin cậy từ kết quả. Không như các hệ thống khác, các ứng dụng có thể điều chỉnh không gian tìm kiếm và đối tượng kết quả giữa các bước, cho phép ứng dụng trở thành một đối tác trong quá trình xử lý.

Giống bộ ngôn ngữ, bộ quản lý tìm kiếm không bị ràng buộc với bất cứ bổ sung cụ thể nào. Ví dụ các bổ sung của bộ quản lý tìm kiếm có thể thực hiện các thuật toán tìm kiếm như Viterbi đồng bộ khung, A*, bi-directional,...

Mỗi bổ sung bộ quản lý tìm kiếm sử dụng một token đi qua thuật toán mô tả bởi Young. Một token Sphinx là một đối tượng được gắn với một trạng thái tìm kiếm và chứa đánh giá âm và ngôn ngữ của đường đi tại một điểm cho trước, một tham chiếu đếntrạng thái tìm kiếm, một tham chiếu đến một khung đặc trưng (Feature frame) vào và các thông tin liên quan khác. Tham chiếu trạng thái tìm kiếm cho phép bộ quản lý tìm kiếm liên hệ với một token tới phân bố trạng thái, đơn vị ngữ âm phụ thuộc ngữ cảnh, cách phát âm, từ và trạng thái ngữ pháp. Mọi phần giả thiết tận cùng bằng một token có hiệu lực.

Các bổ sung của một bộ quản lý tìm kiếm có thể xây dựng một tập các token có hiệu lực trong định dạng của một danh sách hoạt động (ActiveList) tại thời điểm mỗi bước (không yêu cầu). Sphinx cung cấp một framework bổ sung để hỗ trợ bộ quản lý tìm kiếm bao gồm một danh sách họa động, một bộ cắt tỉa (Pruner) và một bộ đánh giá (Scorer).

Framework bổ sung SearchManager cũng giao tiếp với Scorer, một module ước lượng xác xác suất trạng thái cung cấp các giá trị mật độ xuất trạng thái theo yêu cầu. Khi SearchManager yêu cầu một đánh giá cho một trạng thái cho trước tại một thời điểm cho trước, Scorer truy cập đến vector đặc trưng tại thời điểm đó và thự thi các phép toán để tính toán điểm số. Trong trường hợp này, việc giải mã song

song sử dụng các mô hình thính giác song song, Scorer ghép mô hình thính giác được dùng dựa vào loại đặc trưng.

Scorer giữ lại tất cả thông tin gắn liền với các mật độ xuất trạng thái. Do đó, SearchManager không cần biết việc đánh giá được hoàn thành với các HMM lên tục, bán liên tục hay rời rạc. Hơn nữa, hàm mật độ xác suất của mỗi trạng thái bị tách biệt trong cùng một kiểu. Bất cứ giải thuật heuristic kết hợp vào hàm đánh giá để tăng tốc độ có thể cũng được thực thi cục bộ bên trong bộ đánh giá. Thêm nữa, bộ đánh giá có thể tận dụng nhiều CPU nếu sẵn có.

Sphinx4 cung cấp các thành phần bổ sung của SearchManager có thể tháo lắp được, hỗ trợ đồng bộ khung Viterbi, Bushderby và giải mã song song:

- SimpleBreadthFirstSearchManager: Thực hiện một tìm kiếm Viterbi đồng bộ khung đơn giản với Pruner được gọi trên mỗi khung. Pruner mặc định quản lý cả các tia tuyệt đối và tương đối. Bộ quản lý tìm kiếm phát sinh các Result chứa các con trỏ tới các đường dẫn hoạt động ở frame cuối cung được xử lý.

- WordPruningBreadthSearchManager: thực hiện một tìm kiếm Viterbi đồng bộ khung với một bộ cắt tỉa có thể tháo lắp (pluggable Pruner) được gọi trên mỗi khung. Thay vì quản lý một ActiveList đơn giản, nó quản lý một tập các ActiveList, mỗi danh sách cho một loại trạng thái được định nghĩa bởi Linguist.

- BushderbySearchManager: thực thi một tìm kiếm theo chiều rộng đồng bộ khung tổng quát (generalized frame-synchronous breath-first search) sử dụng thuật toán Bushderby, thực hiện phân lớp dựa trên năng lượng tự do, trái với khả năng.

- ParallelSearchManager: thực thi một tìm kiếm Viterbi đồng bộ khung trên nhiều luồn đặc trưng sử dụng một tiếp cận HMM theo ngôn ngữ yếu tố, ngược lại với tiếp cận tiếp cận HMM kết hợp bởi AVCSR. Một ưu điểm của tìm kiếm theo yếu tố là có thể thực hiện nhanh hơn và nhỏ gọn hơn nhiều so với tìm kiếm toàn bộ một HMM tổng hợp.

4.3. QUẢN LÝ CẤU HÌNH SPHINX:

Hệ thống quản lý cấu hình Sphinx (Sphinx configuration manager system) có 2 mục đích chính:

- Xác định những thành phần nào được dùng trong hệ thống: Hệ thống Sphinx được thiết kế trở nên cực kỳ linh hoạt. Trong lúc chạy (runtime), bất cứ thành phần nào cũng có thể được thay thế bởi cái khác. Trong Sphinx, thành phần bộ ngoại vi cung cấp các đặc trưng âm học được dùng để đánh giá ngược lại mô hình âm học. Thông thường, Sphinx được cấu hình với 1 bộ ngoại vi phát sinh các Mel frequency cepstral coefficient (MFCCs), tuy nhiên vẫn có thể cấu hình lại Sphinx để dùng một bộ ngoại vi khác, ví dụ phát sinh Perceptual Linear Prediction coeficients (PLP). Bộ quản lý cấu hình Sphinx được dùng để cấu hình hệ thống theo cách này.

- Xác định cấu hình chi tiết của mỗi thành phần này: Hệ thống Sphinx có 1 số lượng lớn các thành phần điều khiển cách thức hệ thống thực thi. Ví dụ một beam width thường sử dụng để điều khiển số lượng đường dẫn tìm kiếm được duy trì trong quá trình giải mã âm thanh.

Cấu hình của một hệ thống Sphinx được xác định bởi một tập tin cấu hình dạng xml. Tập tin cấu hình này định nghĩa:

- Các tên và kiểu của tất cả các thành phần của hệ thống.

- Liên kết của các thành phần này. Đó là các thành phần nào liên hệ với thành phần khác.

- Cấu hình chi tiết cho mỗi thành phần này.

Bảng 4.1. Các thẻ định dạng trong tập tin cấu hình

Thẻ	Các thuộc tính	Thành phần con	Mô tả
<*config*>	không có	<*component*> <*property*> <*propertylist*>	Thành phần cấp cao nhất, không có thuộc tính và có thể chứa bất cứ thành phần nào trong *component*, *property* và *propertylist*
<*component*>	**name** - tên của thành phần **type** - loại của thành phần	<*property*> <*propertylist*>	Định nghĩa một thực thể của một thành phần. Thẻ này phải luôn có thuộc tính *name* và *type*
<*property*>	**name** - tên thuộc tính **type** - loại thuộc tính	Không có	Dùng để định nghĩa một thuộc tính đơn của thành phần hay một thuộc tính hệ thống toàn cục

<*propertylist*>	**name** - tên của danh sách thuộc tính	<*item*>	Dùng để định nghĩa một danh sách các chuỗi hay các thành phần. Thẻ này phải luôn có thành phần name. Nó có thể chứa bất cứ số lượng *item* con nào.
<*item*>	Không có	Không có	Nội dung của thẻ này là một chuỗi hay một tên của thành phần.

CHƯƠNG 5: CHƯƠNG TRÌNH DEMO

5.1. CÀI ĐẶT CHƯƠNG TRÌNH

Sphinx chạy tốt trong hệ điều hành Linux. Có thể cài đặt Sphinx trong hệ điều hành Windows bằng cách dùng Cygwin để tạo môi trường Linux. Tuy nhiên, tỷ lệ thành công không cao. Chương trình demo được cài đặt trong hệ điều hành Ubuntu.

5.1.1. Tải các gói Sphinx cần thiết:

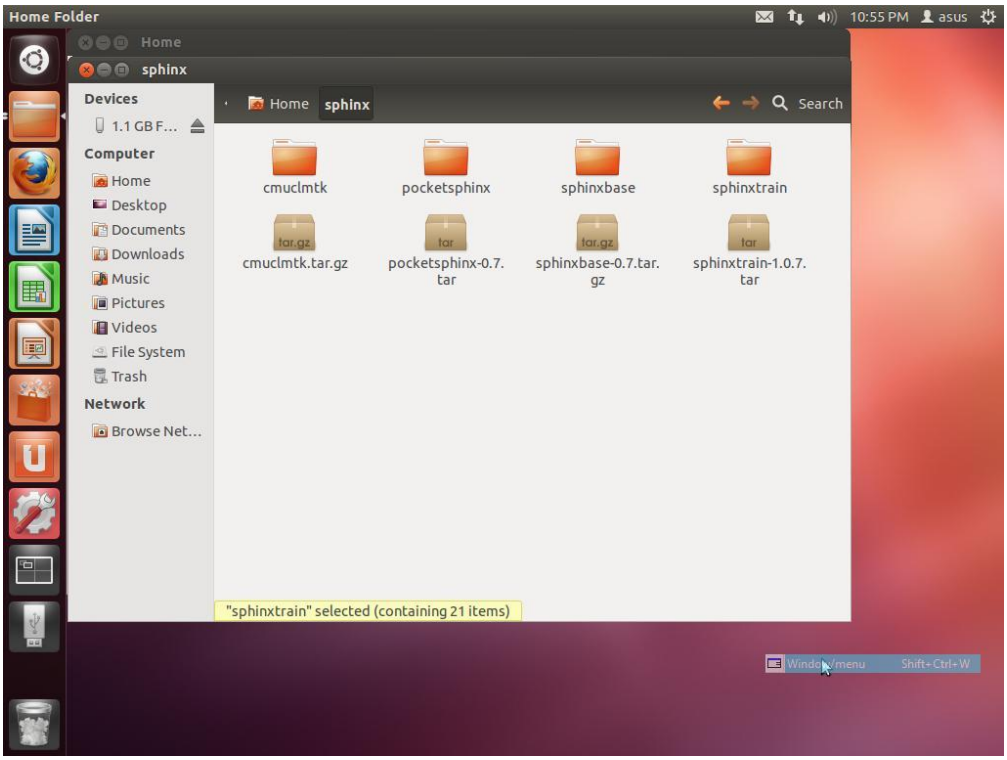
Để cài đặt được Sphinx trên Ubuntu cần tài các gói sau về và để trong cùng một thư mục (đã tạo và đặt tên, nằm trong thư mục Home).

Các gói bao gồm:

- *Pocketsphinx* - một thư viện nhận dạng viết bằng ngôn ngữ C.
- *Sphinxbase* - gói thư viện nền, hỗ trợ các thư viện cần thiết cho các gói khác.
- *Sphinx4* - gói hỗ trợ nhận dạng viết bằng java.
- *CMUclmtk* - bộ công cụ xây dựng mô hình ngôn ngữ.
- *Sphinxtrain* - bộ công cụ huấn luyện mô hình ngữ âm.

5.1.2. Cài đặt:

Tạo một thư mục tên sphinx trong Home folder (trong máy ảo Ubuntu). Chép các tập tin (Sphinxbase, Sphinxtrain, Pocketsphinx, CMUclmtk) vừa download trong mục trên vào đó và giải nén (lưu ý xóa đi chỉ số version sau khi extract).



Hình 5.1. Cài đặt Sphinx

Sử dụng cửa sổ Terminal trong Ubuntu: Ctrl+Atl+t.

Nhập vào sudo apt-get update sau đó nhập vào password lúc cài đặt (password sẽ không hiện lên, nhập cẩn thận và nhấn Enter). Lệnh trên để update cho các gói cài đặt dùng bằng lệnh apt-get. Chờ update xong

Nhập vào: cd sphinx để di chuyển tới thư mục *sphinx* vừa tạo.

Cài đặt các gói cần thiết trước khi cài SphinxBase:

Gõ các lệnh:

- sudo apt-get install bison, đồng ý để tải và cài bison
- sudo apt-get install autoconf
- sudo apt-get install automake
- sudo apt-get install libtool

5.1.2.1. Cài đặt SphinxBase

Nhập lệnh: cd sphinxbase để đi vào thư mục *sphinxbase*.

Gõ các lệnh sau và chờ thi hành:

- ./autogen.sh
- ./configure
- make

- sudo make install

5.1.2.2. Cài đặt Sphinxtrain

Từ thư mục sphinxbase ở trên, gõ lệnh để chuyển sang thư mục sang thư mục sphinctrain:

```
cd ../sphinxtrain
```

Gõ các lệnh sau và chờ thi hành:

- ./configure
- make
- sudo make install

5.1.2.3. Cài đặt PocketSphinx

Từ thư mục sphinxtrain ở trên, gõ lệnh để chuyển sang thư mục sang thư mục pocketsphinx:

```
cd ../pocketsphinx
```

Gõ các lệnh sau và chờ thi hành:

- ./autogen.sh
- ./configure
- make
- sudo make install

Gõ tiếp lệnh sau vào Terminal:

```
sudo ldconfig
```

5.2. XÂY DỰNG BỘ NGÔN NGỮ:

Bộ ngôn ngữ cho chương trình nhận dạng tiếng nói bao gồm 3 thành phần chính: Bộ từ điển, mô hình ngôn ngữ và mô hình âm học. Phần này sẽ mô tả quá trình xây dựng các thành phần đó bằng các công cụ của Sphinx.

5.2.1. Xây dựng bộ từ điển:

Công việc đầu tiên là tạo một bộ từ điển phù hợp. Bộ từ điển này bao gồm các từ mong muốn chương trình nhận dạng.

Do các công cụ huấn luyện của Sphinx chưa hỗ trợ tốt cho unicode nên các ký tự không thuộc bảng mã ASCII sẽ sử dụng phương pháp:

- Các ký tự không thuộc bảng mã ASCII sẽ được thay thế bằng kiểu gõ telex.

- Xây dựng bảng phiên âm tiếng Việt mức âm vị dưới dạng ASCII. (Tham khảo phụ lục)

Bảng phiên âm tiếng tiếng Việt mức âm vị được xây dựng dựa trên các tiêu chí:

- Biểu diễn được hết các âm vị có thể có trong tiếng Việt dưới dạng mã ASCII.

- Mọi âm vị đều được tổ hợp từ các ký hiệu sẵn có trên bàn phím để tiện lợi cho việc nhập liệu.

- Cách gõ telex tiện lợi, dễ sử dụng và dễ hiểu.

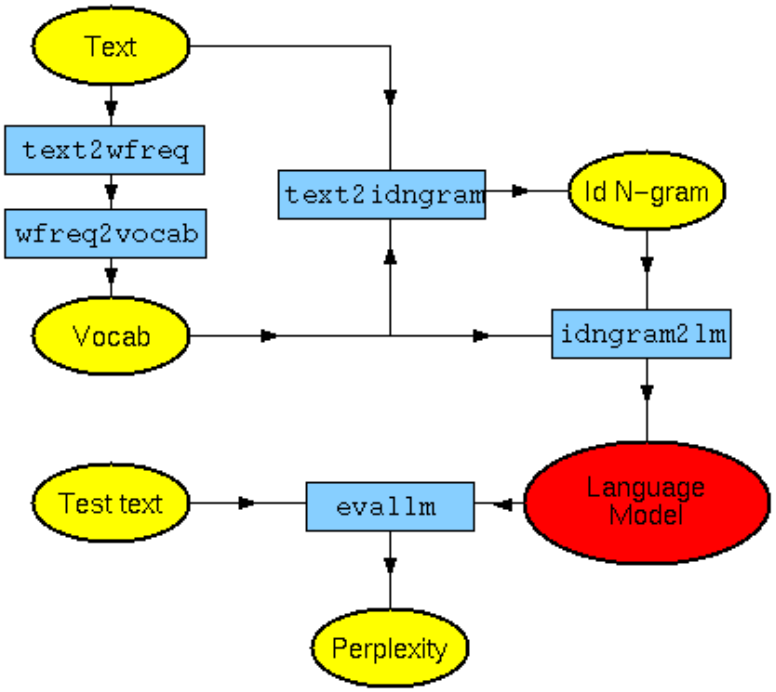
- Các thanh điệu được ký hiệu bằng các ký tự S, F, R, X, J và khoảng trắng.

Thanh điệu được đặt cho các nguyên âm.

Bộ từ điển được tổ chức như sau:

```
LEEN        L EE NZ
NUWXA       N UWX A
XUOOSNG         X U OOS NGZ
TRASI       TR AS IZ
PHARI       F AR IZ
QUA         K A
LAJI        L AJ IZ
TIEESP      T I EES PC
TRUWOWSC        TR WAS KC
VEEF        V EEF
DDAAFU          DD AAF UZ
CUOOSI      K UOS IZ
BAWST       B AWS TZ
NGHE        NG EZ
NHAJC       NH AJ KC
WEB         K ES PC
VAWN        V AW NZ
BARN        B AR NZ
…
```

5.2.2. Xây dựng mô hình ngôn ngữ:



Hình 5.2. Sơ đồ quá trình tạo mô hình ngôn ngữ bằng công cụ CMUclmk

Công cụ dùng để tạo mô hình ngôn ngữ là CMUclmk. Quá trình tạo một mô hình ngôn ngữ bao gồm các bước sau:

5.2.2.1. Chuẩn bị tập tin văn bản:

Bước này để phát sinh ra mô hình ngôn ngữ. Công cụ mô hình ngôn ngữ tiếp nhận đầu vào là một tập tin dạng text, có các câu được bao bởi các thẻ <s> và </s>. Ví dụ tập tin dkmt.txt

```
<s> SOẠN THẢO VĂN BẢN </s>
<s> DI CHUYỂN TRỎ VỀ ĐẦU </s>
<s> NGHE NHẠC </s>
....
```

Do CMUclmk chưa hỗ trợ tổ unicode nên để đảm bảo công cụ CMUclmk có thể xử lý tốt tập tin này, các ký tự không thuộc ASCII sẽ được chuyển thành kiểu gõ telex:

```
<S> SOAJN THARO VAWN BARN </S>
<S> DI CHUYEERN TROR VEEF DDAAFU </S>
<S> NGHE NHAJC </S>
...
```

5.2.2.2. Phát sinh bộ từ vựng:

Bộ từ vựng là một tập tin .vocab, chứa tất cả các từ hoặc tiếng trong tập tin văn bản. Nó được tạo ra bởi CMUclmk và sẽ được dùng để tạo mô hình ngôn ngữ. Tạo bộ từ vựng này bằng lệnh:

```
text2wfreq < dkmt.txt > dkmt.wfreq
wfreq2vocab < dkmt.wfreq > dkmt.vocab
```

Ta sẽ thu được tập tin dkmt.wfreq chứa danh sách tất cả các từ (tiếng) kèm theo số lần xuất hiện của nó trong văn bản. Tập tin từ vựng dkmt.vocab chứa tất cả các từ trong văn bản được sắp xếp theo thứ tự alphabet.

5.2.2.3. Phát sinh mô hình ngôn ngữ:

Mô hình ngôn ngữ có định dạng .arpa. Để tạo ra mô hình này, sử dụng 2 lệnh sau:

```
text2idngram -vocab dkmt.vocab -idngram dkmt.idngram < dkmt.txt
idngram2lm -vocab_type 0 -idngram dkmt.idngram -vocab dkmt.vocab -arpa dkmt.arpa
```

Định dạng ARPA (hay Doug Paul) cho mô hình N-gram backoff có cấu trúc như sau:

\data
ngram 1=*n1*
ngram 2=*n2*
...
ngram *N*=*nN*
\1-grams:
p *w* [*bow*]
...
\2-grams:
p *w1 w2* [*bow*]
...
*****N***-grams:***
p *w1 ... wN*
...
\end

Tập tin này có phần mở đầu với từ khóa \data\, liệt kê số lượng N-gram. Sau đó các N-gram được liệt kê mỗi dòng, được nhóm lại thành từng phần theo

chiều dài. Mỗi phần bắt đầu với từ khóa \N-gram; trong đó N là chiều dài 1, 2, …. Mỗi dòng N-gram bắt đầu với logarit (cơ số 10) của điều kiện xác suất p của N-gram đó, theo sau bởi các từ w1, w2, … wN tạo nên N-gram đó. Từ khóa \end\ kết thúc biểu diễn mô hình.

Để Sphinx có thể sử dụng được, phải chuyển tập tin này sang dạng nhị phân bằng công cụ sphinxbase. Lệnh chuyển đổi:

```
sphinx_lm_convert -i dkmt.arpa -o dkmt.lm.DMP
```

5.2.3. Xây dựng mô hình âm học:

Mô hình âm học bao gồm một biểu diễn thống kê các thanh âm riêng biệt tạo nên mỗi từ trong mô hình ngôn ngữ hay bộ ngữ pháp. Mỗi âm thanh riêng biệt tương ứng với một âm vị. Quá trình huấn luyện mô hình âm học được sử dụng bằng công cụ sphinxtrain.

Chuẩn bị dữ liệu:

Tạo một thư mục huấn luyện, mang tên dkmt.

Trong đó tạo 2 thư mục con là **etc**, **wav**.

Sau đó tạo các tập tin như cấu trúc sau:

etc
|___ dkmt.**dic** - *bộ tự điển âm vị, âm tiết*
|___ dkmt.**phone** - *tập tin chứa danh sách các âm vị*
|___ dkmt.**lm.DMP** - *Mô hình ngôn ngữ*
|___ dkmt.**filler** - *Danh sách các khoảng lặng*
|___ dkmt_train.**fileids** - *Danh sách các tập tin huấn luyện*
|___ dkmt_train.**transcription** - *Dữ liệu dạng text của tập tin huấn luyện*
|___ dkmt_test.**fileids** - *Danh sách các tập tin test*
|___ dkmt_test.**transcription** - *Dữ liệu dạng text của tập tin test*
wav
|___ train
    |___ speaker_1

```
                |___ file_1.wav- tập tin thu âm một câu nói của
người huấn luyện
                |___ ...
|___ test
        |___ speaker_1
                |___ file_1.wav
        |___ ...
```

Tập tin dkmt.dic

Tập tin này là tập tin từ điển đã chuẩn bị từ đầu. Nó chứa nội dung về cách phát âm của một từ trong bộ huấn luyện.

Mỗi một dòng trong tập tin là định nghĩa cách đọc của một từ.

Tập tin này có phân biệt ký tự hoa - thường. Thông thường để xây dựng được tập tin này, cần tìm hiểu về cách phát âm của một từ trong một ngôn ngữ nhất định. Nếu là tiếng Anh thì họ có cách đọc cho từ tiếng Anh có trong tự điển. Đây cũng làm một bước quan trọng để xây dựng thành công bộ huấn luyện.

Trong tiếng Việt, cách đọc và các viết một từ là gần như gắng liền với nhau. Không cần có hướng dẫn cách đọc khi học tiếng Việt, trong tiếng Anh cách đọc và cách viết không phụ thuộc nhau, vd “lead” (dẫn đầu) & “head” (cái đầu). Ví dụ: muốn xây dựng tập tin này cho tiếng Việt, ta có thể định nghĩa các từ bằng nhiều cách như sau:

- BAN B A N

Với cách trên, ta xem từ “BAN” là một âm tiết với sự kết hợp của 3 âm vị là B, A, N.

- BAN B AN

Với cách trên, ta xem từ “BAN” là một âm tiết với sự kết hợp của 2 âm vị là B, AN.

Sphinx không hỗ trợ định nghĩa ở dạng word-base, nghĩa là cách đọc của một từ không được chính là từ đó. Vd: BAN BAN là không được cho phép. Tuy nhiên có thể làm một phương pháp tương đương thay thế nếu mún xây dựng theo

kiểu word-base. Khi đó phải định nghĩa từ theo kiểu 1 từ có nhiều cách đọc, ví dụ:

BAN BAN BANG

Ý nghĩa của dòng định nghĩa trên là từ "ban" có thể đọc theo 2 cách là "ban" (cách đọc đúng chuẩn) hoặc đọc là "bang" (cách đọc người miền Nam).

Chỉ được dùng các ký hiệu a-z, A-Z, 0-9 để đảm bảo không gây lỗi cho tập tin này.

Vấn đề thanh điệu:

Ta sẽ xem các âm vị đi chung với thanh điệu sẽ là một âm vị đọc lập. Khi đó thay vì xem thanh điệu như một âm vị khác theo cách định nghĩa sau (định nghĩa cho từ "bản"):

BARN B A R N

Ta sẽ xem âm ả là một âm vị khác, độc lập với âm a khi đó ta định nghĩa như sau:

BARN B AR N

Tập tin dkmt.phone

Tập tin này chứa tất các các âm vị (phiên âm) sử dụng trong tập tin trên, mỗi một dòng là một âm vị, nên sắp xếp các âm vị đó theo thứ tự để Sphinx dễ quản lý. Lưu ý thêm một âm vị đặc biệt vào tập tin này đó là SIL, âm vị đại diện cho khoảng lặng.

Tập tin dkmt.lm.DMP

Tập tin này là mô hình ngôn ngữ thống kê được xây dựng từ trước bằng công cụ CMUclmk, định dạng ARPA hoặc DMP.

Tập tin dkmt.filler

Tập tin này chứa các âm tiết dùng để "làm đầy", thông thường là các khoảng lặng, được định nghĩa như sau:

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

Tập tin dkmt_train.fileids

Tập tin này là tập tin liệt kê đường dẫn đến các tập tin ghi âm trên mỗi dòng, nằm trong thư mục wav, có trong sơ đồ thư mục trình bày phía trên.

```
speaker_1/file_1
speaker_2/file_2
```

Không ghi đuôi tập tin .wav vào. Mỗi một dòng là một tập tin.

Tập tin dkmt_train.transcription

Đây là phần nội dung mà tập tin wav đã thu âm được. Để huấn luyện cho Sphinx hiểu những gì chúng ta nói, cần cung cấp một tập tin text để giúp cho Sphinx hiểu và học từ đó. Cấu trúc một tập tin .transcript gồm nhiều dòng, mỗi một dòng là nội dung của một tập tin wav kèm theo tên tập tin wav đó.

```
<S> DI CHUYEERN CHUOOJT </S> (file_1)
<S> MOWR TAAJP TIN </S> (file_2)
```

Dữ liệu âm thanh:

Dùng các chương trình ghi âm để ghi âm các câu nói sử dụng các từ (tiếng) cần huấn luyện. Âm thanh được ghi vào với các thông số sau [10]:

- Default Sample Rate Format: 16000Hz

- Default Sample Format: 16-bit

- Channels: 1(Mono)

- File Format: wav, raw hoặc sph

5.3. CẤU HÌNH HUẤN LUYỆN SPHINX:

5.3.1. Điều chỉnh tham số:

5.3.1.1. Cấu hình thư mục huấn luyện:

Sau khi đã cài đặt các gói cần thiết trong Ubuntu, chúng ta chép thư mục vừa tạo ở bước 5 vào cùng thư mục với thư mục Sphinx đã tạo trước đó [9].

Để bắt đầu quá trình huấn luyện, sử dụng các lệnh của sphinxtrain và pocketsphinx để cấu hình thư mục huấn luyện:

```
../SphinxTrain/scripts_pl/setup_SphinxTrain.pl -task dkmt
```

```
../pocketsphinx/scripts/setup_sphinx.pl -task dkmt
```

Với dkmt là tên của thư mục huấn luyện. Lệnh trên sẽ sao chép các phần cần thiết lên thư mục huấn luyện:

```
bin
bwaccumdir
etc
feat
logdir
model_parameters
model_architecture
python
scripts_pl
wav
```

5.3.1.2. Điều chỉnh các tham số:

Thông tin cấu hình nằm trong tập tin sphinx_train.cfg. Một số cấu hình quan trọng:

- Cấu hình để huấn luyện tập tin âm thanh định dạng wav:

```
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'wav';
$CFG_WAVFILE_TYPE = 'mswav';
```

- Điều chỉnh loại mô hình (huấn luyện HMM liên tục, bán liên tục), bỏ dấu # trước mô hình cần huấn luyện:

```
$CFG_HMM_TYPE = '.cont.'; # Sphinx 4, Pocketsphinx
#$CFG_HMM_TYPE  = '.semi.'; # PocketSphinx
#$CFG_HMM_TYPE  = '.ptm.'; # PocketSphinx (larger data sets)
```

- Cấu hình tham số mật độ CFG có thể nhận các giá trị 4, 8, 16, 32, 64 tùy theo độ lớn của dữ liệu:

```
$CFG_FINAL_NUM_DENSITIES = 8;
```

- Cấu hình số lượng các *senone* để huấn luyện trong một mô hình. Số lượng *senone* càng lớn, sphinx phân biệt các âm càng chính xác. Nhưng mặt khác, nếu bạn có quá nhiều *senone*, mô hình sẽ không được tổng quát đủ để nhận dạng các tiếng nói vô hình. Nghĩa là số từ lỗi sẽ tăng cao trên dữ liệu chưa huấn luyện. Đó là lý do

quan trọng để không nên huấn luyện quá mức các mô hình. Trong trường hợp có quá nhiều *senone* vô hình sẽ phát sinh cảnh báo lỗi.

```
# Number of tied states (senones) to create in decision-tree clustering
$CFG_N_TIED_STATES = 200;
```

Theo nghiên cứu của nhóm CMUSphinx thì cấu hình dựa theo bảng sau:

Bảng 5.1. Thông số cấu hình

Kích thước từ vựng	Số giờ huấn luyện	Senones	Densities	Ví dụ
20	5	200	8	Mô hình nhận dạng số
100	20	2000	8	Mô hình ra lệnh điều khiển
5000	30	4000	16	Mô hình đọc chính tả 5000 từ
20000	80	4000	32	Mô hình đọc chính tả 20000 từ
60000	200	6000	16	Mô hình HUB
60000	2000	12000	64	Mô hình Fisher Rich Telephone Transcription

5.3.2. Thực thi huấn luyện:

5.3.2.1. Tạo vector đặc trưng:

Hệ thống sẽ không làm việc trực tiếp với các tín hiệu âm thanh. Trước tiên, các tín hiệu được chuyển thành một chuỗi các vector đặc trưng, được dùng thay cho các tín hiệu âm thanh thực sự. Để thực thi biến đổi (hay sự tham số hóa) này, trong thư mục huấn luyện, thực thi 2 lệnh sau:

```
./scripts_pl/make_feats -ctl etc/dkmt_train.fileids

./scripts_pl/make_feats -ctl etc/dkmt_test.fileids
```

Tập tin kịch bản này sẽ tính toán một chuỗi các vector 13 hướng (các vector đặc trưng) cho mỗi cách nói, bao gồm các Mel-frequency cepstral coefficients (MFCCs). Các tập tin chứa đường dẫn tuyệt đối tới các tập tin âm thanh. Các MFCC sẽ được tự động đặt vào thư mục ./feat.

5.3.2.2. Huấn luyện:

Sử dùng lệnh:

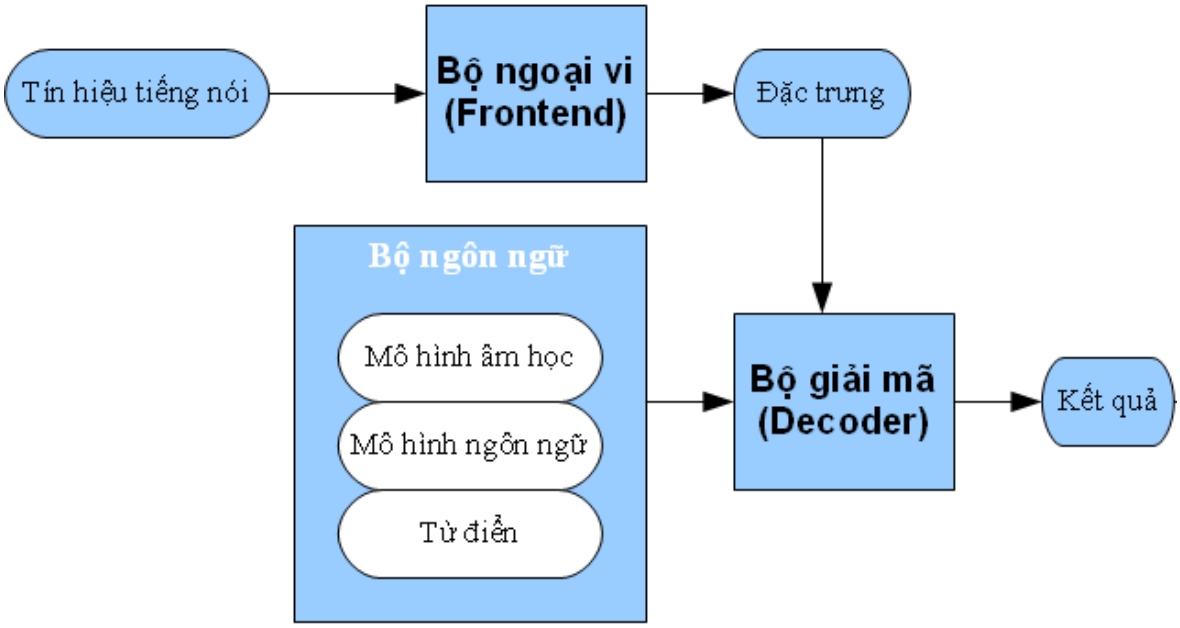
./scripts_pl/RunAll.pl

Lệnh trên sẽ duyệt qua các phần yêu cầu. Quá trình huấn luyện sẽ xuất ra các thông báo dạng sau:

```
Baum welch starting for 2 Gaussian(s), iteration: 3 (1 of 1)
   0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
   Normalization for iteration: 3
   Current Overall Likelihood Per Frame = 30.6558644286942
   Convergence Ratio = 0.633864444461992
   Baum welch starting for 2 Gaussian(s), iteration: 4 (1 of 1)
   0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
   Normalization for iteration: 4
```

5.4. KẾT QUẢ THỬ NGHIỆM:

Quá trình hoạt động:



Hình 5.3. Sơ đồ hoạt động của chương trình demo

Đầu tiên, tín hiệu tiếng nói qua micro sẽ được đưa vào bộ ngoại vi, ở đây tín hiệu được tham số hóa thành một dãy đặc trưng và chuyển vào cho bộ giải mã. Bộ ngôn ngữ chuyển đổi các mô hình ngôn ngữ, thông tin phát âm trong từ điển và thông tin cấu trúc âm trong mô hình âm học vào một đồ thị tìm kiếm trong bộ giải mã. Bộ giải mã sẽ xác định chuỗi đặc trưng gần giống nhất trong đồ thị tìm kiếm so với đặc trưng tiếng nói được cung cấp bởi bộ ngoại vi và phát sinh kết quả.

Thông số hệ thống:

Chương trình demo được xây dựng trên hệ thống với các thông số như sau:

- Máy laptop Dell Inspiron N4030.

- Bộ xử lý Intel Core i3, 2.5 GHz, 3GB RAM.

- Hệ điều hành Linux và Windows 7.

- Card âm thanh onboard.

- Micro dùng để thu và nhận dạng là dạng micro kèm headphone.

- Tiếng nói được thu với tần số lấy mẫu 16.000 Hz, kích thước mỗi mẫu là 16 bit.

Hiện tại luận văn đã xây dựng được chương trình demo nhận dạng tiếng Việt với khoảng trên 100 câu lệnh điều khiển cơ bản cho máy tính gồm:

DI CHUYỂN CHUỘT
DI CHUYỂN CHUỘT LÊN
DI CHUYỂN CHUỘT LÊN TRÊN
LÊN
LÊN TRÊN
CHUỘT LÊN TRÊN
CHUỘT LÊN TRÊN NỮA
TRÊN NỮA
TRÊN
LÊN TRÊN NỮA
NỮA
DI CHUYỂN CHUỘT XUỐNG
DI CHUYỂN CHUỘT XUỐNG DƯỚI
XUỐNG
XUỐNG DƯỚI
CHUỘT XUỐNG DƯỚI
CHUỘT XUỐNG DƯỚI NỮA
DƯỚI NỮA
DƯỚI
XUỐNG DƯỚI NỮA
DI CHUYỂN CHUỘT QUA PHẢI
QUA PHẢI NỮA
CHUỘT QUA PHẢI
QUA
QUA PHẢI
PHẢI
DI CHUYỂN CHUỘT QUA TRÁI
QUA TRÁI NỮA
CHUỘT QUA TRÁI

QUA TRÁI
TRÁI
DI CHUYỂN CHUỘT VỀ ĐẦU
VỀ ĐẦU
CHUỘT VỀ ĐẦU
ĐẦU
BẮT ĐẦU
DI CHUYỂN CHUỘT VỀ CUỐI
VỀ CUỐI
CHUỘT VỀ CUỐI
CUỐI
CUỘN CHUỘT
CUỘN CHUỘT LÊN
CUỘN CHUỘT XUỐNG
NHẠC
DỪNG
NGHE TIẾP
TẮT NHẠC
CHƯƠNG TRÌNH NGHE NHẠC
ĐÓNG CHƯƠNG TRÌNH NHẠC
WEB
ĐÓNG WEB
VĂN BẢN
CHỌN TẬP TIN
MỞ TẬP TIN
TẬP TIN
HỦY
SOẠN THẢO VĂN BẢN
ĐÓNG CHƯƠNG TRÌNH SOẠN THẢO
ĐÓNG SOẠN THẢO
ĐÓNG VĂN BẢN
THOÁT
CON TRỎ
TRỎ
TRỎ VỀ ĐẦU
TRỎ VỀ CUỐI
CON TRỎ LÊN
CON TRỎ LÊN TRÊN
CON TRỎ LÊN ĐẦU
TRỎ LÊN
TRỎ LÊN TRÊN
TRỎ LÊN TRÊN NỮA
TRỎ LÊN ĐẦU
CON TRỎ XUỐNG
CON TRỎ XUỐNG DƯỚI
CON TRỎ XUỐNG CUỐI
TRỎ XUỐNG

TRỎ XUỐNG DƯỚI
TRỎ XUỐNG DƯỚI NỮA
TRỎ XUỐNG CUỐI
THÊM THẺ
THẺ MỚI
ĐÓNG THẺ
THÊM TRANG
TRANG MỚI
ĐÓNG TRANG
XUỐNG DÒNG
ĐẦU DÒNG
CUỐI DÒNG
DÒNG TRÊN
DÒNG DƯỚI
XUỐNG HÀNG
ĐẦU HÀNG
CUỐI HÀNG
HÀNG TRÊN
HÀNG DƯỚI
TỚI
LUI
TO
PHÓNG TO
NHỎ
THU NHỎ

- Tổng thời gian ghi âm khoảng 50 giờ.

- Dữ liệu kiểm tra gồm 660 câu nói.

Kết quả thử nghiệm như sau:

- Số lượng câu đúng: 545/660. Độ chính xác: 82,57%.

- Số lượng từ đúng 1557/1675. Độ chính xác: 92,95%.

Như vậy có thể thấy kết quả nhận dạng câu không cao, độ chính xác chỉ đạt khoảng 82%. Tuy nhiên kết quả nhận dạng từ đúng lại cao hơn, trên 92%.

KẾT LUẬN

KẾT QUẢ ĐẠT ĐƯỢC:

Qua quá trình nghiên cứu về nhận dạng tiếng nói tiếng Việt và ứng dụng thử nghiệm trong điều khiển máy tính, luận văn đã làm được một số công việc sau:

- Nghiên cứu về tiếng nói, các phương pháp xử lý tiếng nói, rút trích đặc trưng.

- Nghiên cứu và thực hiện huấn luyện mô hình âm học theo âm vị, áp dụng cho tiếng Việt.

- Nghiên cức kiến trúc một hệ thống nhận dạng tiếng nói qua các công cụ của CMUSphinx.

- Xây dựng chương trình demo nhận dạng tiếng nói tiếng Việt liên tục.

Do chưa có nhiều kiến thức về xử lý tín hiệu số và xử lý tiếng nói nên luận văn không tránh khỏi nhiều thiếu sót. Tuy nhiên, với một số kết quả đã đạt được hy vọng luận văn sẽ góp một phần nhỏ vào việc nghiên cứu nhận dạng tiếng nói tiếng Việt.

HƯỚNG PHÁT TRIỂN:

Do việc thu âm xử lý dữ liệu chưa được phong phú nên kết quả chưa được tốt. Việc này có thể được khắc phục bằng cách thu nhiều mẫu hơn và huy động thêm những người tình nguyện để thu âm. Có thể xem xét tận dụng nguồn âm tiếng nói trên radio, internet để làm phong phú thêm bộ dữ liệu huấn luyện. Ngoài ra cần phát triển thêm các phần sau:

- Khảo sát thêm các đặc điểm ngữ âm tiếng Việt và quan sát ảnh phổ để tìm ra các đặc trưng ảnh hưởng đến thanh điệu, cải thiện việc nhận dạng các thanh điệu.

- Cải tiến phương pháp tách từ trong câu để có kết quả nhận dạng tốt hơn.

- Tìm hiểu thêm về mô hình ngôn ngữ và các thuật toán tìm kiếm trong nhận dạng tiếng nói để tăng tốc độ nhận dạng.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

[1]. Đặng Hoài Bắc (2006), *Xử lý tín hiệu số,* Học viện Công nghệ Bưu chính Viễn thông.

[2]. Đặng Ngọc Đức, Nguyễn Tiến Dũng, Lương Chi Mai (2011), *Mô hình và phiên âm tiếng Việt mức âm vị,* Institute of Information Technology, Vietnamese Academy of Science and Technology.

[3]. Cao Xuân Hạo (1998), *Tiếng Việt - mấy vấn đề ngữ âm, ngữ pháp, ngữ nghĩa,* NXB Giáo dục.

[4]. Quách Tuấn Ngọc, Mai Công Nguyên (1998), *Nhận dạng lời nói liên tục với bộ từ vựng lớn,* Tiểu luận môn Nhận dạng tiếng nói, Đại học Bách khoa Hà Nội.

[5]. Quách Tuấn Ngọc, Phạm Xuân Trường (1998), *Phương pháp phân tích và xử lý nhận dạng tiếng nói,* Tiểu luận môn Xử lý tiếng nói, Đại học Bách khoa Hà Nội.

[6]. Phan Nguyễn Phục Quốc, Hà Thúc Phùng (2009), *Hệ thống nhận dạng tiếng nói,* Luận văn Đại học, Đại học Bách khoa TP.HCM.

[7]. Thái Hùng Văn, Đỗ Xuân Đạt, Võ Văn Tuấn (2003), *Nghiên cứu các đặc trưng của tiếng Việt áp dụng vào nhận dạng tiếng nói tiếng Việt,* Luận văn Đại học, Đại học KHTN TP.HCM.

Tiếng Anh:

[8]. Xuedong Huang, Alex Acero, Hsiao-wuen Hon (2001), *Spoken language Processing,* Carnegie Mellon University.

[9]. CMUSphinx Wiki: http://cmusphinx.sourceforge.net/wiki/

[10]. Record your Speech with Audacity: http://www.voxforge.org/home/submitspeech/windows/step-2

PHỤ LỤC

BẢNG PHIÊN ÂM TIẾNG VIỆT DƯỚI DẠNG MÃ ASCII

STT	Âm vị		Chữ	Ví dụ	Mô tả
	IPA	ASCII			
Âm đầu					
1	**b**	**b**	b	ba	phụ âm tắc, hai môi, hữu thanh, không bật hơi, chỉ xuất hiện trong âm tiết không có âm đệm
2	**d**	**dd**	đ	đẩy	phụ âm tắc, đầu lưỡi lợi, hữu thanh, không bật hơi
3	**t**	**t**	t	tùng	phụ âm tắc, đầu lưỡi răng, vô thanh, không bật hơi.
4	**t’**	**th**	th	thích	phụ âm tắc, vô thanh, bật hơi, đầu lưỡi răng.
5	ʄ	**tr**	tr	trăng	phụ âm tắc, đầu lưỡi vòm miệng, vô thanh, không bật hơi.
6	**c**	**ch**	ch	chỉ	phụ âm tắc, vô thanh, mặt lưỡi, không bật hơi
7	**k**	**k**	k (trước i, e, ê)	keo	phụ âm tắc, vô thanh, gốc lưỡi, không bật hơi
			c (trước u, ư, a, o,...)	cảnh	
			q (trước u)	quậy	
8	**m**	**m**	m	mềm	phụ âm vang mũi, hai môi, xuất hiện trong âm tiết không có âm đệm
9	**n**	**n**	n	nóng	phụ âm vang mũi, đầu lưỡi lợi
10	ɲ	**nh**	nh	nhà	phụ âm vang mũi, mặt lưỡi
11	ŋ	**ng**	ng (trước u, ư, o, ô, ơ, a, ă, â)	ngủ	phụ âm vang mũi, gốc lưỡi
			ngh (trước i, e, ê)	nghỉ	
12	**f**	**f**	ph	phê	phụ âm xát, vô thanh, môi răng, xuất hiện trong âm tiết không có âm đệm
13	**v**	**v**	v	vội	phụ âm xá, hữu thanh, môi răng, xuất hiện trong âm tiết

					không có âm đệm
14	**s**	**x**	x	xa	phụ âm xát, vô thanh, đầu lưỡi lợi
15	**z**	**d**	d	dễ	phụ âm xát, hữu thanh, đầu lưỡi lợi
			gi	giỏi	
			g (trước i)	gì	
16	**l**	**l**	l	lắm	phụ âm vang bên, đầu lưỡi răng
17	**ʂ**	**s**	s	sơn	phụ âm xát, vô thanh, dầu lưỡi vòm miệng, uốn lưỡi
18	**ʐ**	**r**	r	rằm	phụ âm xát, hữu thanh, đầu lưỡi vòng miệng, uốn lưỡi
19	**χ**	**kh**	kh	khá	phụ âm xát, vô thanh, gốc lưỡi
20	**ɣ**	**g**	g (trước u, ư, o, ô, ơ, a, ă, â)	găm	phụ âm xát cuối lưỡi, hữu thanh
			gh (trước i, e, ê)	ghế	
21	**h**	**h**	h	hòa	Phụ âm xát, vô thanh, họng
22	**p**	**p**	p	pi	phụ âm tắc, hai môi, ...
Âm đệm					
23	**w**	**w**	o (trước nguyên âm rộng a, ă, e)	hoa	có cấu tạo giống nguyên âm chính /u/, có độ mở hẹp, phát âm cực trầm, tròn môi, thuộc hàng sau
			u (còn lại)	hủy	
Âm chính					
24	**i**	**i**	y (đứng sau u)	suy	nguyên âm đơn dài, hàng trước, hẹp, không tròn môi, có tính bổng, trước /k, ŋ/ bị rút ngắn
			i (còn lại)	tính	
25	**e**	**ee**	ê	chê	nguyên âm đơn, dài, hàng trước, hơi hẹp, không trong môi, có tính chất bổng, trước /k, ŋ/ bị rút ngắn
26	**ɛ**	**e**	e	chè	nguyên âm đơn, dài, hàng trước, hơi rộng, không tròn môi, có tính chất bổng
27	**ɛ̆**	**ea**	a (trước ch, nh)	sách	nguyên âm đơn, ngắn. Gần như là thể ngắn của /ɛ/
28	**u**	**u**	u	sung	nguyên âm đơn, dài, hàng sau, hẹp, tròn môi, có âm sắc trầm. Đứng trước /k, ŋ/ bị rút ngắn

29	**o**	**oo**	ô	cô	nguyên âm đơn, dài, hàng sau, hơi hẹp, tròn môi, có âm sắc trầm. Thể dài khi không đứng trước /k, ŋ/
30	ɔ	**o**	o	con	nguyên âm đơn, dài, hàng sau, hơi rộng, tròn môi, có âm sắc trầm. Thể dài khi không đứng trước /k, ŋ/
31	ɔ ˇ	**oa**	o (trước c, ng)	cọc	nguyên âm đơn, ngắn
32	ɯ	**uw**	ư	từ	nguyên âm đơn, dài, hàng sau, hẹp, không tròn môi, âm sắc trầm vừa
33	ɤ	**ow**	ơ	tơ	nguyên âm đơn, dài, hàng sau, hơi hẹp, không tròn môi, có âm sắc trầm vừa
34	ɤˇ	**aa**	â	ấm	nguyên âm đơn, ngắn, hàng sau, hơi hẹp, không tròn môi, có âm sắc trầm vừa. Xuất hiện trong mọi âm tiết trừ âm tiết mở
35	**a**	**a**	a	tan	nguyên âm đơn, dài, hàng sau, rộng, không tròn môi, có âm sắc trầm vừa. xuất hiện trong tất cả các âm tiết
36	**ă**	**aw**	ă	chặn	nguyên âm đơn, ngắn, hàng sau, rộng, không tròn môi, có âm sắc trầm vừa. xuất hiện trong tất cả các âm tiết trừ âm tiết mở
			a (trước u, y)	tay	
37	**ie**	**ie**	ia	bia	nguyên âm đôi yếu dàn, hàng trước, không tròn môi, yếu tố sau là nguyên âm hàng trước, hơi hẹp, không tròn môi
			ya (khi trước có âm đệm)	khuya	
			iê (khi trước không có âm đệm và sau có âm cuối)	tiền	
			yê (khi trước có âm đệm hoặc sau có âm cuối là bán nguyên âm	yêu	
38	**uo**	**uo**	ua (khi không có âm cuối)	chua	nguyên âm đôi yếu dần, hàng sau, tròn môi, yếu tố đầu là

			uô (khi só âm cuối)	cuốn	nguyên âm hàng sau, hơi hẹp, không tròn môi
39	ɯɤ	**wa**	ưa (khi không có âm cuối)	trưa	nguyên âm đôi yếu dần, hàng sau, không tròn môi, yếu tố đầu là nguyên âm hàng sau, hẹp, không tròn môi. Yếu tố sau là nguyên âm hàng sau, hơi hẹp, không tròn môi
			ươ (khi có âm cuối)	lười	
Âm cuối					
40	**p**	**pc**	p	mập	phụ âm cuối, ...
41	**t**	**tc**	t	chật	phụ âm cuối, ...
42	**m**	**mz**	m	câm	phụ âm cuối vang, mũi, môi
43	**n**	**nz**	n	nản	phụ âm cuối vang, mũi, đầu luỡi
44	**k**	**kc**	ch (đứng sau i, e, ê, a)	sạch	phụ âm cuối ồn, mặt lưỡi
			c (trường hợp còn lại)	cục	
45	**ŋ**	**ngz**	nh (đứng sau i, e, ê, a)	vành	phụ âm cuối vang, mũi, mặt lưỡi
			ng (trường hợp còn lại)	vàng	
46	**-w**	**uz**	o (đứng sau e, a)	leo	bán nguyên âm cuối vang, môi
			u (trường hợp còn lại)	cứu	
47	**-j**	**iz**	y (đứng sau nguyên âm ngắn a, â)	bay	bán nguyên âm cuối vang, lợi
			i (trường hợp còn lại)	cài	

Tên thanh điệu	Ngang	Sắc	Huyền	Hỏi	Ngã	Nặng
Ký hiệu		S	F	R	X	J