TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH KỸ THUẬT MÁY TÍNH

**PHÂN LỚP LÀN ĐIỆU CHÈO VÀ DÂN CA QUAN HỌ**

Sinh viên thực hiện: **Lại Văn Hải**

Lớp: IS2 – K58

Giáo viên hướng dẫn: PGS.TS **Trịnh Văn Loan**

HÀ NỘI 05-2018

**PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

1. Thông tin về sinh viên

Họ và tên sinh viên: Lại Văn Hải

Điện thoại liên lạc: 0166 455 2105 Email: hailv.hedspi@gmail.com

Lớp: IS2 Hệ đào tạo: Công nghệ thông tin Việt Nhật

Đồ án tốt nghiệp được thực hiện tại: trường đại học Bách Khoa Hà Nội

Thời gian làm ĐATN: Từ ngày 01/01/2018 đến 20/05/2018

2. Mục đích nội dung của ĐATN

Phân lớp làn điệu chèo và dân ca quan họ Bắc Ninh

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu khái quát về nhận dạng làn điệu âm nhạc

- Tìm hiểu mô hình GMM

- Tìm hiểu về bộ công cụ ALIZE, Spro.

- Sử dụng bộ công cụ ALIZE và Spro trong phân lớp làn điệu chèo và dân ca quan họ

4. Lời cam đoan của sinh viên:

Tôi Lại Văn Hải cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của PGS.TS.Trịnh Văn Loan

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 20 tháng 05 năm 2018*  Tác giả ĐATN  Lại Văn Hải |

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Giáo viên hướng dẫn  PGS.TS. Trịnh Văn Loan |

LỜI CẢM ƠN

Để có thể hoàn thiện được Đồ án tốt nghiệp như ngày hôm nay, nếu chỉ nhờ sự cố gắng của riêng bản thân em thì không thể nào hoàn thành được. Lời đầu tiên em xin chân thành cảm ơn thầy Trịnh Văn Loan đã dìu dắt em qua ba kỳ đồ án, xuất phát điểm chỉ với nền tảng duy nhất là sự hứng thú với đề tài của thầy, ngoài ra thì không có bất kỳ kiến thức nào khác, thầy đã tạo điều kiện cho nhóm đồ án được tham gia những tiết học mà trong chương trình đào tạo của chúng em không có, những câu chuyện bên lề mỗi buổi học của thầy giúp em định hướng được rất nhiều điều, nó vô cùng đáng quý với lứa tuổi chập chững như chúng em đang bị rất nhiều thứ khác cám dỗ. Em xin cảm ơn thầy Chu Bá Thành đã nhiệt tình chỉ dạy những mảng kiến thức vô cùng quan trọng liên quan đến đồ án của em, nhờ có sự chỉ dạy của thầy mà rất nhiều vấn đề khúc mắc của em đã được giải đáp. Em xin gừi lời cảm ơn đến người thân và những bạn bè xung quanh em đã luôn ở bên ủng hộ và khích lệ tinh thần cho em vào những lúc em yếu lòng nhất.

Cuối cùng em xin cảm ơn ngôi trường Bách Khoa đã tôi rèn em trưởng thành như ngày hôm nay, nhiều người bảo Bách Khoa là khó là khổ nhưng với em chưa từng một lần em hối hận khi chọn ngôi trường này. Rồi những ngày hôm nay sẽ trở thành kỷ niệm, nhưng em tin rằng mình đã có những kỷ niệm đẹp nhất ở cuối quãng đời sinh viên này.

TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Hiện nay sự bùng nổ về công nghệ thông tin đã giúp cho các dịch vụ trực tuyến phát triển mạnh mẽ, một trong số đó có thể kể tới dịch vụ nghe nhạc trực tuyến. Tuy nhiên, cùng với sự phát triển này là sự bùng nổ về dữ liệu dẫn tới việc quản lý, phân loại và truy xuất dữ liệu đòi hỏi phải có các công cụ hữu ích hỗ trợ cho công việc này chứ không chỉ đơn thuần làm thủ công. Xuất phát từ nhu cầu thực tiễn này thì việc tạo ra các công cụ hỗ trợ cho việc phân lớp là vô cùng cần thiết

Hiện nay trên thế giới đã có nhiều công trình nghiên cứu về phân loại các thể loại âm nhạc hiện đại như Pop, Rock, Ballad,…Trong đồ án tốt nghiệp này em sẽ trình bày về “**Phân lớp làn điệu chèo và dân ca quan họ**”.

Các vấn đề sẽ được đề cập trong đề tài:

MỤC LỤC

[LỜI CAM ĐOAN 1](#_Toc367651201)

[LỜI CẢM ƠN 2](#_Toc367651202)

[MỤC LỤC 3](#_Toc367651203)

[DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT 7](#_Toc367651204)

[DANH MỤC HÌNH 8](#_Toc367651205)

[DANH MỤC BẢNG 9](#_Toc367651206)

[MỞ ĐẦU 10](#_Toc367651207)

[Lý do chọn đề tài 10](#_Toc367651208)

[Mục đích, phạm vi nghiên cứu 11](#_Toc367651209)

[Đối tượng nghiên cứu 12](#_Toc367651210)

[Phương pháp nghiên cứu 12](#_Toc367651211)

[Nhiệm vụ nghiên cứu 12](#_Toc367651212)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 13](#_Toc367651213)

[1.1. Tổng quan về nhận dạng người nói 13](#_Toc367651214)

[1.1.1. Nhận dạng người nói là gì? 13](#_Toc367651215)

[1.1.2. Các ứng dụng của nhận dạng người nói 14](#_Toc367651216)

[1.2. Phương pháp nhận dạng người nói phụ thuộc từ khóa 14](#_Toc367651217)

[1.2.1. Giới thiệu chung 14](#_Toc367651218)

[1.2.2. Các thành phần của một hệ thống nhận dạng người nói phụ thuộc từ khóa …………………………………………………………………….18](#_Toc367651219)

[1.2.3. Các giai đoạn xử lý của một hệ thống nhận dạng người nói phụ thuộc từ khóa 20](#_Toc367651220)

[1.3. Tìm hiểu về các kết quả nhận dạng người nói đã có với Tiếng Việt 21](#_Toc367651221)

[1.3.1. Định danh người nói tiếng Việt sử dụng mô hình hỗn hợp Gaussian của nhóm tác giả Đinh Lê Thành, Nguyễn Quốc Linh, Trịnh Văn Loan 21](#_Toc367651222)

[1.3.2. Xây dựng và khảo sát độ dài từ khóa trong nhận dạng người nói phụ thuộc vào từ khóa tiếng Việt theo mô hình Markov ẩn tác giả: Ngô Minh Dũng, Đặng Văn Chuyết 22](#_Toc367651223)

[CHƯƠNG 2. CÁC CÔNG CỤ SỬ DỤNG TRONG ĐỂ TÀI 25](#_Toc367651224)

[2.1. Bộ công cụ ALIZE 25](#_Toc367651225)

[2.1.1. Nguồn gốc 25](#_Toc367651226)

[2.1.2. Giới thiệu về ALIZE 25](#_Toc367651227)

[2.1.3. Bộ công cụ SPro 27](#_Toc367651228)

[2.1.4. Các công cụ nhận dạng người nói sử dụng SPro và ALIZE 28](#_Toc367651229)

[2.2. Bộ công cụ Sphinx3 36](#_Toc367651230)

[2.2.1. Giới thiệu 36](#_Toc367651231)

[2.2.2. Tổng quan về bộ giải mã s3.X 37](#_Toc367651232)

[2.2.3. Công cụ nhận dạng tiếng nói sử dụng Sphinx3 39](#_Toc367651233)

[CHƯƠNG 3. TRIỂN KHAI HỆ THỐNG THỬ NGHIỆM NHẬN DẠNG NGƯỜI NÓI TIẾNG VIỆT 42](#_Toc367651234)

[3.1. Sơ đồ tổng quan quá trình xây dựng hệ thống nhận dạng người nói phụ thuộc từ khóa tiếng Việt 42](#_Toc367651235)

[3.2. Chuẩn bị dữ liệu 43](#_Toc367651236)

[3.2.1. Chuẩn bị dữ liệu cho nhận dạng bởi công cụ ALIZE 47](#_Toc367651237)

[3.2.2. Chuẩn bị dữ liệu cho nhận dạng bởi công cụ Sphinx3 50](#_Toc367651238)

[3.3. Nhận dạng người nói sử dụng ALIZE 53](#_Toc367651239)

[3.3.1. Bước 1: Tạo thư mục làm việc 54](#_Toc367651240)

[3.3.2. Bước 2: Tính tham số MFCC 54](#_Toc367651241)

[3.3.3. Bước 3: Dò tìm năng lượng 54](#_Toc367651242)

[3.3.4. Bước 4: Phát hiện tiếng nói trong tín hiệu 55](#_Toc367651243)

[3.3.5. Bước 5: Chuẩn hóa các tham số của tín hiệu 55](#_Toc367651244)

[3.3.6. Bước 6: Chuẩn hóa TrainWorldInit 55](#_Toc367651245)

[3.3.7. Bước 7: Chuẩn hóa TrainWorldFinal 56](#_Toc367651246)

[3.3.8. Bước 8: Huấn luyện GMM cho từng người nói 56](#_Toc367651247)

[3.3.9. Bước 9: Nhận dạng người nói 56](#_Toc367651248)

[3.4. Nhận dạng tiếng nói sử dụng Sphinx3 57](#_Toc367651249)

[3.4.1. Bước 1: Tạo cấu trúc thư mục và các file cấu hình cần thiết 58](#_Toc367651250)

[3.4.2. Bước 2: Cập nhật dữ liệu huấn luyện 58](#_Toc367651251)

[3.4.3. Bước 3: Huấn luyện mô hình âm học 58](#_Toc367651252)

[3.4.4. Bước 4: Tính tham số MFCC của file Wav 59](#_Toc367651253)

[3.4.5. Bước 5: Tính điểm số từng từ khóa qua Sphinx3 59](#_Toc367651254)

[3.5. Nhận dạng người nói phụ thuộc từ khóa tiếng Việt 61](#_Toc367651255)

[3.5.1. Nhận dạng người nói sử dụng ALIZE và Sphinx3 61](#_Toc367651256)

[3.5.2. Phân tích và đánh giá kết quả 65](#_Toc367651257)

[Chương 4. KẾT LUẬN VÀ KIẾN NGHỊ 68](#_Toc367651258)

[4.1. Những vấn đề đã giải quyết trong luận văn 68](#_Toc367651259)

[4.2. Hướng phát triển của đề tài 69](#_Toc367651260)

[TÀI LIỆU THAM KHẢO 70](#_Toc367651261)

[PHỤ LỤC 72](#_Toc367651262)

[A. Mô hình hỗn hợp Gauss 72](#_Toc367651263)

[A.1. Đặc tả mô hình 72](#_Toc367651264)

[A.2. Bài toán ước lượng mật độ 74](#_Toc367651265)

[B. Ba bài toán cơ bản của HMM 76](#_Toc367651266)

[B.1. Bài toán thứ nhất: Đánh giá xác suất 76](#_Toc367651267)

[B.2. Thuật toán thứ hai: Tìm dãy trạng thái tối ưu 78](#_Toc367651268)

[B.3. Thuật toán thứ ba: Ước lượng tham số của mô hình 79](#_Toc367651269)

[C. Cấu trúc các gói trong thư viện LIA-RAL 81](#_Toc367651270)

[C.1. Cấu trúc file NormFeat.cfg 81](#_Toc367651271)

[C.2. Cấu trúc file NormFeat\_energy.cfg 82](#_Toc367651272)

[C.4. Cấu trúc file TrainWorldInit.cfg 85](#_Toc367651273)

[C.5. Cấu trúc file TrainWorldFinal.cfg 86](#_Toc367651274)

[C.6. Cấu trúc gói TrainTarget 87](#_Toc367651275)

[C.7. Cấu trúc gói ComputeTest 88](#_Toc367651276)

[D. Code chương trình nhận dạng người nói tiếng Việt 89](#_Toc367651277)

[D.1. Code chương trình tổng hợp xác suất của ALIZE và Sphinx 89](#_Toc367651278)

[D.2. Code chương trình chạy xác suất tiên nghiệm với ALIZE và Sphinx 102](#_Toc367651279)

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| ***Chữ viết tắt*** | ***Chữ viết đầy đủ*** |
| **ANN** | Artificial Neural Network |
| **ASV** | Automatic Speaker Identification |
| **ASI** | Automatic Speaker Verification |
| **ASR** | Automatic Speaker Recognition |
| **DTW** | Dynamic Time Warping |
| **EM** | Expectance Maximization |
| **FSG** | Finite State Grammar |
| **GMM** | Gaussion Mixture Model |
| **HMM** | Hidden Markov Model |
| **MFCC** | Mel-Frequency Cepstral Coefficients |
| **LM** | Language Model |
| **UML** | Unified Modeling Language |
| **VQ** | Vector Quantization |

DANH MỤC HÌNH

[Hình 1.1. Hệ thống xác định người nói 15](#_Toc367523867)

[Hình 1.2. Mô hình xác định người nói 16](#_Toc367523868)

[Hình 1.3. Hệ thống xác minh người nói 17](#_Toc367523869)

[Hình 1.4. Mô hình xác minh người nói 17](#_Toc367523870)

[Hình 1.5. Các thành phần của một hệ thống nhận dạng người nói 19](#_Toc367523871)

[phụ thuộc từ khóa 19](#_Toc367523872)

[Hình 1.6. Hệ thống định danh người nói sử dụng giải thuật GMM 21](#_Toc367523873)

[Hình 1.7. Pha nhận dạng hệ nhận dạng người nói phụ thuộc từ khóa thay đổi 23](#_Toc367523874)

[Hình 1.8. Kết quả khảo sát độ chính xác nhận dạng người nói phụ thuộc độ dài từ khóa 24](#_Toc367523875)

[Hình 2.1. Thành phần của gói công cụ ALIZE 26](#_Toc367523876)

[Hình 2.2. Sơ đồ các công cụ sử dụng ALIZE trong nhận dạng người nói 28](#_Toc367523877)

[Hình 3.1. Hệ nhận dạng người nói phụ thuộc từ khóa tiếng Việt 42](#_Toc367523878)

[Hình 3.2. File wav biểu diễn tín hiệu phát âm từ khóa 0 đến 9 của người nói 45](#_Toc367523879)

[Hình 3.3. Các bước xây dựng hệ nhận dạng người nói sử dụng SPro & ALIZE 53](#_Toc367523880)

[Hình 3.4. Sơ đồ tổng quát xây dựng Forced alignment proces 57](#_Toc367523881)

[Hình 3.5. Biểu đồ thể hiện kết quả nhận dạng với Test 1 63](#_Toc367523882)

[Hình 3.6. Biểu đồ thể hiện kết quả nhận dạng với Test 2 64](#_Toc367523883)

[Hình 3.7. Biểu đồ thể hiện kết quả nhận dạng với các thử nghiệm 66](#_Toc367523884)

[Hình A.1. Hàm mật độ Gauss 72](#_Toc367523885)

[Hình A.2. Mô hình GMM 73](#_Toc367523886)

[Hình A.3. Hàm mật độ của GMM có 3 phân phối Gauss 74](#_Toc367523887)

DANH MỤC BẢNG

[Bảng 2.1: Mô tả tùy chọn sử dụng tiêu chuẩn hóa năng lượng 30](#_Toc367523888)

[Bảng 2.2: Mô tả tùy chọn sử dụng phát hiện tín hiệu tiếng nói 31](#_Toc367523889)

[Bảng 2.3: Mô tả tùy chọn chuẩn hóa đặc trưng 32](#_Toc367523890)

[Bảng 2.4: Mô tả tùy chọn chính sử dụng huấn luyện mô hình từ 33](#_Toc367523891)

[Bảng 2.5: Mô tả tùy chọn trong huấn luyện mô hình đích 34](#_Toc367523892)

[Bảng 2.6: Mô tả tùy chọn tính điểm số của mô hình GMM 36](#_Toc367523893)

[Bảng 3.1: Thông tin về dữ liệu 48 người nói 43](#_Toc367523894)

[Bảng 3.2: Kết quả nhận dạng người nói thử nghiệm với Test 1: sử dụng xác suất tiên nghiệm lấy từ mô hình HMM và GMM của người nói 63](#_Toc367523895)

[Bảng 3.3: Kết quả nhận dạng người nói thử nghiệm với Test 2: Xác suất tiên nghiệm lấy từ mô hình GMM và xác suất hậu nghiệm lấy từ mô hình HMM của từng người nói 64](#_Toc367523896)

[Bảng 3.4: Kết quả tổng hợp các thử nghiệm theo thời gian 65](#_Toc367523897)

[Bảng 3.5: Bảng tổng hợp kết quả nhận dạng các chương trình đã thử nghiệm 66](#_Toc367523898)

MỞ ĐẦU

Lý do chọn đề tài

Hiện nay đang trong giai đoạn của cuộc Cách mạng công nghệ 4.0, sự bùng nổ về thông tin đang diễn ra trên toàn cầu, chưa khi nào mà việc con người đặt câu hỏi và tìm kiếm câu trả lời lại dễ dàng như hiện nay. Nhưng đối với bất kỳ câu chuyện nào cũng tồn tại hai mặt của một vấn đề, sự bùng nổ về mặt dữ liệu dẫn tới nhu cầu về mặt quản lý cũng khắt khe hơn, một trong những nguồn dữ liệu mà em muốn nhắc tới đó chính là âm nhạc.

Âm nhạc là một trong những thông tin phổ biến nhất và trên các trang web về nghe nhạc trực tuyến có đến hàng triệu bản nhạc, điều này đặt ra một thách thức lớn trong việc tổ chức, truy xuất và tìm kiếm. Một trang web nghe nhạc mà chỉ liệt kê một cách đơn thuần danh sách các bài hát có trong hệ thống thì không thể coi đây là một trang web tốt được, chính vì vậy nó đòi hỏi phải được phân loại, sắp xếp theo tên ca sĩ, tên album, thể loại, năm phát hành,… Chính từ nhu cầu này đã nảy sinh ra yêu cầu về việc phân loại âm nhạc, nhưng điều gì sẽ xảy ra nếu chúng ra thực hiện điều này thủ công, có thể với số lượng vài trăm file âm thanh thì chưa có vấn đề gì xảy ra nhưng câu chuyện sẽ trở nên hoàn toàn khác khi con số này lên tới hàng triệu. Do đó phân loại nhạc tự động thành các loại khác nhau là một nhiệm vụ quan trọng để truy xuất và tổ chức các thư viện nhạc.

Là một người có tình yêu với văn hóa truyền thống nên em rất hứng thú với những giá trị văn hóa cổ truyền của dân tộc và nhạc cổ truyền cũng không phải là ngoại lệ. Chính vì vậy, cùng với nhu cầu thực tiễn được đề cập ở bên trên nên em đã quyết định lựa chọn đề tài “Phân lớp làn điệu chèo và dân ca quan họ Bắc Ninh”, làm để tài nghiên cứu của mình. Với mong muốn thông qua việc này có thể góp một nhỏ vào việc trùng hưng lại những làn điệu chèo và quan họ cổ, để không chỉ có thế hệ chúng em mà cả những thế hệ sau này vẫn sẽ tiếp tục duy trì ngọn lửa văn hóa truyền thống trong lòng mỗi người dân Việt Nam.

Mục đích, phạm vi nghiên cứu

* Tìm hiểu tổng quan về phân lớp làn điệu âm nhạc.
* Nghiên cứu mô hình GMM và bộ công cụ ALIZE, gói thư viện LIA-RAL để nhận diện làn điệu âm nhạc.
* Tìm hiểu công cụ Spro để trích chọn đặc trưng.

Đối tượng nghiên cứu

* Nghiên cứu tổng quan về nhận dạng thể loại âm nhạc.
* Nghiên cứu về mô hình hỗn hợp GMM.
* Nghiên cứu bộ công cụ ALIZE, công cụ Spro.

Phương pháp nghiên cứu

* Nghiên cứu lý thuyết về nhận dạng thể loại âm nhạc.
* Nghiên cứu sử dụng bộ công cụ ALIZE trên môi trường Linux trong phân lớp làn điệu âm nhạc.
* Nghiên cứu sử dụng công cụ Spro trên môi trường Linux trong trích chọn đặc trưng.
* Tập hợp các file âm thanh của chèo và quan họ, tiến hành thử nghiệm nhận dạng cho từng làn điệu một.

Nhiệm vụ nghiên cứu

* Tìm hiểu khái quát về nhận dạng làn điệu âm nhạc
* Tìm hiểu mô hình GMM
* Tìm hiểu về bộ công cụ ALIZE, Spro.
* Sử dụng bộ công cụ ALIZE và Spro trong phân lớp làn điệu chèo và dân ca quan họ.

Nội dung luận văn được trình bày trong 105 trang và được chia thành 4 chương:

* Chương 1: Trình bày tổng quan về nhận dạng người nói, các ứng dụng của nhận dạng người nói và đi sâu vào nhận dạng người nói phụ thuộc từ khóa tiếng Việt. Chương này cũng trình bày các kết quả nghiên cứu đã được thực hiện về đề tài nhận dạng người nói tiếng Việt
* Chương 2: Trên cơ sở lý thuyết đã được tìm hiểu ở chương 1, trong chương này sẽ trình bày về cài đặt, cũng như chi tiết từng chức năng của công cụ nhận dạng người nói sử dụng ALIZE và nhận dạng tiếng nói sử dụng Sphinx3
* Chương 3: Chương này sẽ trình bày chi tiết về các bước tạo cơ sở dữ liệu tiếng nói được sử dụng để thử nghiệm và xây dựng hệ thống nhận dạng người nói phụ thuộc từ khóa tiếng Việt thông qua sử dụng công cụ ALIZE và kết hợp sử dụng công cụ Sphinx3.
* Chương 4: Trình bày kết luận và một số hướng phát triển tiếp theo của đề tài

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

* 1. Tổng quan về nhận dạng người nói
     1. Nhận dạng người nói là gì?

Nhận dạng người nói là một hệ thống sinh trắc học, thực hiện các nhiệm vụ tính toán chứng thực người dùng trên cơ sở khẳng định danh tính của một người bằng cách sử dụng đặc điểm đặc trưng riêng lấy từ tiếng nói của họ. Nhận dạng tiếng nói đã phát triển hơn bốn thập kỷ và sử dụng các tính năng âm thanh của tiếng nói phân biệt khác nhau của từng cá nhân. Đặc điểm tiếng nói của con người được phát ra chứa 2 thông tin như sau:

* Thông tin mức thấp: độ cao, trường độ, cường độ, tần số, nhịp điệu, giai điệu, quang phổ, âm formant …những đặc điểm này thường được nhận dạng bởi hệ thống nhận dạng qua quá trình huấn luyện để đưa ra quyết định
* Thông tin mức cao: bao gồm phương ngữ, dấu nhấn khi nói chuyện, ngữ cảnh, phong cách chủ đề của bối cảnh ... những đặc điểm này thường liên quan đến việc áp dụng khả năng tự nhiên của con người để xác định ai là người nói?

Các thông tin này không phải bất biến từ lúc người biết nói đến lúc già, nhưng nó có tính khá ổn định trong giai đoạn dài của cuộc đời. Khi con người đã trưởng thành, những thói tật khi nói, những đặc trưng khu biệt trong cấu âm sẽ hình thành và mang tính ổn định cao. Đây chính là cơ sở khoa học cho việc xây dựng các hệ thống nhận dạng người nói.

Một hệ thống nhận dạng người nói lý tưởng phải có khả năng nhận dạng đúng người nói trong mọi môi trường. Tuy nhiên, một số yếu tố ảnh hưởng trong quá trình nhận dạng gây nên kết quả lỗi như sau:

* Trạng thái cảm xúc của người nói. Ví dụ như căng thẳng hay ép buộc…
* Vị trí đặt của thiết bị thu âm ở mỗi thời điểm ghi âm khác nhau
* Âm thanh được ghi ở môi trường không phù hợp: phòng nhiều tiếng ồn, bị ảnh hưởng bởi nhiễu
* Sử dụng kênh ghi âm, xác thực không phù hợp
* Tốc độ phát âm khi thử nghiệm khác so với dữ liệu huấn luyện
* Người thử nghiệm bị bệnh (như cảm lạnh… ảnh hưởng đến đường âm), người cao tuổi (đường âm thanh bị thay đổi theo độ tuổi)

Các nguồn nêu các lỗi xác minh dẫn đến kết quả việc nhận dạng người nói thực sự khá phức tạp.

* + 1. Các ứng dụng của nhận dạng người nói

Nhận dạng người nói là lĩnh vực ứng dụng rất nhiều trong đời sống:

* Xác minh danh tính và kiểm soát truy cập:
* Truy cập ngân hàng qua điện thoại
* Mua thẻ điện thoại
* Truy cập cơ sở dữ liệu, thông tin bí mật máy tính từ xa
* Nhận dạng tội phạm:
* Kiểm soát sử dụng điện thoại trong nhà tù khi người nhà phạm nhân tới thăm
* Tìm ra tội phạm thông qua việc so khớp giọng nói
* Hệ thống bán hàng gián tiếp (lấy lại thông tin khách hàng)
* Lấy được thông tin khách hàng gọi tới trung tâm.
* Lấy mẫu giọng nói của khách hàng qua điện thoại.
* Mục đích pháp y
  1. Phương pháp nhận dạng người nói phụ thuộc từ khóa
     1. Giới thiệu chung

Tùy thuộc vào ứng dụng, nhận dạng người nói đươc chia thành 2 bài toán cụ thể: Nhận dạng (Identification), phát hiện /xác minh (Detection/Verification). Cả 2 bài toán đều sử dụng một cơ sở dữ liệu các mẫu tham khảo cho N người nói đã biết trước và đều sử dụng các kỹ thuật phân tích và quyết định tương tự nhau.

Nhận dạng người nói (Speaker Identification) là xác định người nói là ai trong một nhóm người nói mà máy tính đã có đầu vào là các mẫu tiếng nói biết trước.

Xác minh người nói (Speaker Verification) là xác định xem tiếng nói được phát ra có đúng là giọng của người cần kiểm tra hay không? Cách này còn được gọi là xác nhận giọng nói, chứng thực, xác thực người nói, phát hiện người nói.

Xác minh người nói thì đơn giản hơn nhận dạng người nói vì nó chỉ yêu cầu so sánh mẫu kiểm tra so với một mẫu tham khảo và cho ra quyết định mẫu kiểm tra có khớp với mẫu tham khảo hay không? Trong khi đó nhận dạng người nói lại yêu cầu chọn ra trong số N giọng nói được báo trước cho hệ thống, giọng nào khớp tốt nhất là giọng kiểm tra. Vì phải thực hiện N so sánh và quyết định nên sai số trong nhận dạng có thể nhiều hơn trong xác minh người nói.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dương** |  | ***Tôi muốn đăng nhập hệ thống***  **Identification**  System |  | **Xin chào,**  **bạn Dương !** |
| **Cường** |  | ***Tôi muốn đăng nhập hệ thống*** | **Xin chào,**  **bạn Cường !** |
| **??** |  | ***Tôi muốn đăng nhập hệ thống*** | **Xin lỗi, bạn không được hệ thống nhận dạng!** |

Hình 1.1. Hệ thống xác định người nói

Hình 1.1. là một hệ thống xác định người nói thông qua cách truy cập bằng dữ liệu tiếng nói vào hệ thống. Với những người nói đã được lưu trữ giọng nói có trong dữ liệu của hệ thống, sau khi đăng nhập bằng mật khẩu tiếng nói, hệ thống sẽ xác định xem ai là người đang nói, đã có trong dữ liệu chưa. Nếu đã có rồi thì hệ thống đưa ra lời chào và đồng ý đăng nhập. Nếu chưa có trong hệ thống thì sẽ đưa ra thông báo không hợp lệ và không được truy cập vào hệ thống

Cơ sở

dữ liệu

người nói

Huấn luyện người nói

***Huấn luyện***



***Thử nghiệm***

Lựa chọn

tốt nhất

Hình 1.2. Mô hình xác định người nói

Hình 1.2 là mô hình xác định người nói, ở mô hình này sẽ mô tả các bước cho giai đoạn huấn luyện và thử nghiệm của một hệ thống xác định người nói. Đầu tiên hệ thống sẽ lấy ra đặc trưng từ dữ liệu tiếng nói. Sau đó, ở bước huấn luyện sẽ tạo ra mô hình của từng người nói riêng biệt, tất cả mô hình này được đưa vào cơ sở dữ liệu chung. Ở bước thử nghiệm: dữ liệu tiếng nói sau khi được trích chọn đặc trưng sẽ đem so khớp với dữ liệu tiếng nói ở trong cơ sở dữ liệu của toàn bộ người nói đã được lưu trong bước huấn luyện. Kết quả nào có độ tương đồng cao nhất hoặc điểm số lớn nhất sẽ được đưa ra và tương ứng là tên của người nói cần tìm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dương** |  | ***Tôi là Dương!*** | **Verification**  **System** | **Xin chào,**  **bạn Dương !** |
| **Cường** |  | ***Tôi là Tuấn!*** | **Xin lỗi,**  **Không phải bạn Tuấn!** |

Hình 1.3. Hệ thống xác minh người nói

Hình 1.3. là một hệ thống xác minh người nói thông qua cách truy cập bằng dữ liệu tiếng nói vào hệ thống. Bước thử nghiệm như sau: dữ liệu tiếng nói được đưa kèm theo tên của người cần kiểm tra. Nếu dữ liệu tiếng nói đúng với tên của người cần kiểm tra, hệ thống sẽ hiển thị lời chào cùng tên. Ngược lại, nếu là giả mạo thì hệ thống cũng đưa ra cảnh báo.

Cơ sở

dữ liệu

Huấn luyện người nói

***Huấn luyện***



***Thử***

***nghiệm***

Hình 1.4. Mô hình xác minh người nói

Hình 1.4. là mô hình xác minh người nói, mô hình này có cách thực hoạt động gần giống như mô hình xác định người nói đã trình bày ở trên. Nhưng chương trình đơn giản hơn khá nhiều, ở bước thử nghiệm để đưa ra kết quả nhận dạng, hệ thống không cần so khớp với tất cả dữ liệu tiếng nói có trong cơ sở dữ liệu mà chỉ so khớp với dữ liệu tiếng nói của người cần kiểm tra và câu trả lời là có Đúng/Sai người cần kiểm tra với giọng nói đưa vào hay không?

Theo các ràng buộc về tín hiệu tiếng nói sử dụng trong hệ thống huấn luyện và kiểm tra. Hệ thống nhận dạng người nói được chia thành 2 loại: ***Độc lập văn bản và phụ thuộc văn bản.***

Trong nhận dạng phụ thuộc văn bản, người sử dụng phải nói vào cụm từ được biết trước của hệ thống, có thể được cố định hoặc gợi ý từ hệ thống. Tức là sử dụng cùng một văn bản cho cả giai đoạn huấn luyện và kiểm tra. Với thông tin biết trước của cụm từ đã cung cấp sẽ cho kết quả nhận dạng tốt hơn. Hệ thống phụ thuộc văn bản đòi hỏi đoạn văn được nói khi sử dụng hệ thống (trong giai đoạn kiểm tra) phải giống với đoạn văn được dùng trong lúc huấn luyện (trong giai đoạn lưu giữ) hoặc phải là một sự kết hợp của đoạn văn được dùng trong lúc huấn luyện. Trong luận văn này, tác giả sử dụng mô hình phụ thuộc văn bản với các từ khóa để huấn luyện và thử nghiệm.

Trong hệ thống nhận dạng độc lập văn bản, hệ thống không biết cụm từ mà người sử dụng sẽ nói. Mặc dù điều này sẽ thêm tính linh hoạt của một ứng dụng, nhưng nó có thể giảm độ chính xác của hệ thống nhận dạng. Để có thể thu được kết quả tốt của hệ nhận dạng độc lập văn bản, ta thường cần thu nhiều dữ liệu tiếng nói hơn cho cả giai đoạn huấn luyện và giai đoạn kiểm tra so với hệ phụ thuộc văn bản.

* + 1. Các thành phần của một hệ thống nhận dạng người nói phụ thuộc từ khóa

Cũng như một hệ nhận dạng người nói nói chung, hệ nhận dạng người nói phụ thuộc từ khóa có thành phần tổng quát như hình vẽ dưới đây:

**.**

**.**

**.**

Luật quyết định

Huấn luyện người nói

Đặc trưng tiếng thứ 2

Đặc trưng tiếng thứ 1

Đặc trưng tiếng thứ N

Cơ sở

dữ liệu

người nói

***Huấn luyện***



***Thử nghiệm***

Hình 1.5. Các thành phần của một hệ thống nhận dạng người nói

phụ thuộc từ khóa

Hình 1.5 cho ta thấy các hệ thống nhận dạng người nói phụ thuộc vào từ khóa gồm các thành phần sau:

- Trích chọn đặc trưng: Tín hiệu tiếng nói đưa vào được biến đổi giọng nói thô thành những vector đặc trưng của từng người nói.

- Huấn luyện mô hình người nói: huấn luyện người nói dựa vào một phương pháp cụ thể

- Cơ sở dữ liệu người nói: Lưu trữ tất cả dữ liệu người nói với mô hình người nói đã được trích chọn đặc trưng vào một kho dữ liệu

- Luật quyết định: xác định và đưa ra quyết định dựa vào việc so khớp, tính toán từng từ khóa và tổng hợp cụm từ khóa đã đưa vào kiểm tra với cơ sở dữ liệu người nói có trước để đưa ra kết quả nhận dạng ai là người nói

* + 1. Các giai đoạn xử lý của một hệ thống nhận dạng người nói phụ thuộc từ khóa

Hình 1.5 cho ta thấy quá trình xử lý của các hệ thống nhận dạng ng ười nói phụ thuộc vào từ khóa gồm hai giai đoạn:

* ***Giai đoạn huấn luyện:***
* Ghi âm giọng nói những người mà hệ thống cần huấn luyện: mỗi người nói được ghi âm tiếng nói của mình theo từng cụm từ khóa hoặc dãy từ khóa. Số lượng giọng nói đưa vào càng nhiều thì việc nhận dạng càng chính xác tuy nhiên cần chú ý đến việc lưu trữ dữ liệu tiếng nói có thể lớn.
* Trích chọn đặc trưng: là việc biến đổi tín hiệu tiếng nói thực sang dạng tham số đặc trưng phục vụ tốt hơn cho quá trình nhận dạng, thử nghiệm. Việc trích chọn đặc trưng có ý nghĩa hết sức quan trọng, ảnh hưởng trực tiếp đến kết quả nhận dạng. Có một số phương pháp để sử dụng trích chọn đặc trưng của hệ nhận dạng là: hệ số ceptrum tần số Mel MFCC, hệ số cepstrum dự đoán tuyến tính LPCC, LPC
* Huấn luyện mô hình người nói: Giai đoạn này mục đích là để tạo cho mỗi người nói có một mô hình đặc trưng riêng theo các một phương pháp nhất định. Có nhiều phương pháp để huấn luyện mô hình người nói: mô hình hỗn hợp Gaussion GMM, mô hình Markov ẩn HMM, mô hình lượng tử hóa Vector VQ, mô hình mạng Noron nhân tạo…
* Lưu lại các đặc trưng của mỗi người nói sau khi đã được huấn luyện.
* ***Giai đoạn kiểm thử:***
* Ghi giọng nói người cần kiểm tra
* Trích chọn đặc trưng người nói vừa cung cấp
* Tính toán các đặc trưng tương ứng với từng từ khóa. Nếu người đọc 1,2,5 hay 10 từ khóa thì mỗi từ, cụm từ có một đặc trưng tương ứng với từ khóa của từng người nói lưu trong cơ sở dữ liệu.
* Đưa ra quyết định nhận dạng: Trong bước này thì hệ thống sẽ tính toán và tổng hợp từng từ khóa của mỗi người nói xem có độ tương đồng cao nhất hay có điểm số lớn nhất sẽ là người nói.

CHƯƠNG 2. CÁC CÔNG CỤ SỬ DỤNG TRONG ĐỂ TÀI

* 1. Bộ công cụ ALIZE
     1. Nguồn gốc

ALIZE là một thư viện hỗ trợ lĩnh vực nhận dạng tự động người qua giọng nói, được phát triển từ một hệ thống có sẵn tại KTH (Kungliga Tekniska högskolan - Royal Institute of Technology), gọi là GIVES (General Identity Verification System). Gói LIA-RAL sử dụng thư viện ALIZE và nó là giao diện được dùng trong suốt dự án. Cả hai gói đều được cài đặt bằng C++. Frederic Wils là tác giả của phiên bản ALIZE đầu tiên. Bây giờ ALIZE được bảo trì cùng với Jean-Francois Bonastre, tác giả của LIA-RAL tại phòng thí nghiệm máy tính của trường đại học Avignon, Pháp.

ALIZE là phần mềm miễn phí, có thể dowload tại địa chỉ: [***http://mistral.univ-avignon.fr/download\_en.html***](http://mistral.univ-avignon.fr/download_en.html)***.*** Với 2 gói: gói ALIZE 2.0 có kích thước : 1.028M và gói LIA – RAL2.0 có kích thước: 9.761M.

Để chạy công cụ ALIZE thì cần cài đặt hệ điều hành Linux. Hoặc nếu chạy trên Windows thì cần cài môi trường chạy lệnh UNIX như công cụ CYGWIN. CYGWIN là công cụ miễn phí có thể tải trên mạng ở địa chỉ [***http://www.cygwin.com/***](http://www.cygwin.com/)**.** Trong luận văn này, tác giả thử nghiệm trên môi trường Linux, cài máy ảo phiên bản ***VMware Workstation v9.0.0*** (426M) và hệ điều hành sử dụng là ***CentOS6.3*** (706.5M)

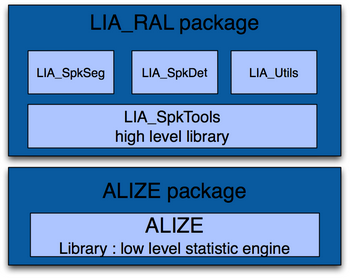
* + 1. Giới thiệu về ALIZE

ALIZE được phát triển trong C + + theo hướng đối tượng UML. Gói LIA-RAL được xây dựng trên thư viện ALIZE nên không thể sử dụng chỉ LIA-RAL. Mặc khác, vì ALIZE chỉ là một thư viện, quá tổng quát để trở thành một hệ thống xác minh người qua giọng nói. Tuy nhiên, LIA-RAL dùng để ước tính các công việc bên trong của ALIZE. Vì vậy, các gói phải được sử dụng cùng nhau. Hầu hết các hoạt động của người dùng đều thông qua LIA -RAL, chỉ duy nhất bộ phân tích các tập tin cấu hình là sử dụng từ ALIZE. Mã trong gói LIA-RAL được chia thành hai loại:

Thứ nhất là thư viện tổng quát (có tên Spk-Tools), chứa các giải thuật huấn luyện và các hàm thống kê được áp dụng trên các véc tơ đặc trưng.

Thứ hai là gói chứa nhiều chương trình con, được thiết kế để làm việc trong phương thức đường ống (pipeline).

Tuy nhiên, không có khung sườn chính thức để cài đặt một hệ thống xác minh người qua giọng nói (ASV) hoàn chỉnh. Việc kết nối tất cả các chương trình con để tạo nên chương trình hoàn chỉnh đòi hỏi việc đọc tài liệu, các tập tin cấu hình có sẵn và kiểm tra tính năng của mỗi chương trình con bên trong.

****

Hình 2.1. Thành phần của gói công cụ ALIZE

**Để cho phép tất cả mọi người sử dụng ALIZE, ALIZE là tập hợp của một số bộ**:

* Một thư viện ở mức độ thấp (ALIZE): thư viện này chứa tất cả các chức năng cần thiết để sử dụng hỗn hợp Gaussian
* Bộ công cụ cấp cao (LIA\_RAL): bộ công cụ phân chia thành các phần như sau:
* *LIA\_SpkTools:* thư viện cấp cao
* *LIA\_Utils:* công cụ cần thiết để xử lý tất cả các định dạng được sử dụng trong ALIZE: mô hình GMM, các thông số, ...
* *LIA\_SpkDet:* một tập hợp các công cụ để làm tất cả các nhiệm vụ được yêu cầu bởi một hệ thống xác thực sinh trắc học: huấn luyện mô hình (tiếng nói / từ), các thông số tiêu chuẩn hóa, chuẩn hóa điểm số, …
* *LIA\_SpkSeg:* một công cụ mới để phát hiện tiếng nói.
  + 1. Bộ công cụ SPro

SPro là một bộ công cụ cung cấp các lệnh thực thi các thuật toán trích chọn đặc trưng tiếng nói và trong các ứng dụng nhận dạng người nói[15].

SPro ban đầu được thiết kế để hiển thị giải phân tích quang phổ nhưng cũng cung cấp cho trích chọn đặc trưng cổ điển sử dụng trong ứng dụng tiếng nói. Có những lệnh chính sau đây:

* Năng lượng filter – bank
* Hệ số cepstral (filter-bank và dự đoán tuyến tính)
* Dự đoán tuyến tính dẫn xuất tiêu biểu (dự báo và phản ánh hệ số, tỷ lệ phạm vi và những dòng cặp phổ).

Các thư viện được viết trong ANSI C, cung cấp các chức năng sau:

* Tín hiệu đầu vào dạng sóng
* Xử lý tín hiệu ở mức thấp (FFT, phân tích LPC,..)
* Xử lý đặc trưng mức thấp ( CMS, chuẩn hóa sự khác biệt, …)

Thư viện không cung cấp các hàm cho trích chọn đặc trưng ở mức cao mà trực tiếp chuyển đổi một dạng sóng vào các đặc trưng, chủ yếu là do hàm như vậy sẽ đòi hỏi một số lượng lớn các đối số để được linh hoạt.

Spro được sử dụng trên nền Linux, SPARC/SunOS và HP-UX. Có thể dowload Spro phiên bản SPro4.0.1, dung lượng 1.3M tại địa chỉ đường link: [***http://www.irisa.fr/metiss/guig/spro/spro-4.0.1/***](http://www.irisa.fr/metiss/guig/spro/spro-4.0.1/)***.***

* + 1. Các công cụ nhận dạng người nói sử dụng SPro và ALIZE

**Tính kết quả**

**nhận dạng**

Huấn luyện mô hình đích (gmm)

**Train**

**Target**



Huấn luyện mô hình nền

Chuẩn hóa đặc trưng

Loại bỏ

khoảng lặng

Trích chọn đặc trưng

**NormFeat**

**TrainWorld**

**Compute**

**Test**

**Energy Detector**

**sfbcep**

**SPro**

**LIA\_SpkDet**

Hình 2.2. Sơ đồ các công cụ sử dụng ALIZE trong nhận dạng người nói

Hình 2.2 mô tả các bước và công cụ liên kết tại mỗi bước, dưới đây là trình bày các bước như sau:

1. Trích chọn đặc trưng

Các đặc trưng được trích chọn thực hiện bởi công cụ dựa trên phân tích filter –bank, sử dụng *sfbcep* trong *SPro*. Việc thực hiện dựa trên câu lệnh sau:

*sfbcep [option] inputfile outputfile*

Các tùy chọn(option) như sau:

*-F, --format=str*

Xác định sóng đầu vào của file. Định dạng hiển thị là ‘PCM16’, ‘wave’ hoặc ‘sphere’. Mặc định là: ‘PCM16’

*-f, --sample-rate=f*

Thiết lập sóng đầu vào tỉ lệ tần sô cho ‘PCM16’. Mặc định là 8kHz

*-x, --channel=n*

Thiết lập đặc trưng cho các kênh. Mặc định là 1

*-B, --swap*

Trao đổi các mẫu sóng đầu vào

*-I, --input-bufsize=n*

Thiết lập bộ đệm đầu vào kích cỡ tới n kbytes. Mặc định: 10MB

*-O, --output-bufsize=n*

Thiết lập bộ đệm đầu vào kích cỡ tới n kbytes. Mặc định: 10MB

*-k, --pre-emphasis=f*

Thiết lập hệ số tiền nhấn mạnh đến f. Mặc định: 0.95

*-l --length=f*

Thiết lập các phân tích dài khung tới *f ms.* Mặc định: 20.0 ms.

*-d, --shift=f*

Thiết lập khoảng thời gian giữa hai khung liên tiếp tới ms.

Mặc định là: 10ms

*-p, --num-ceps=n*

Thiết lập số lượng đầu ra hệ số Cepstral đến n. Mặc định là 12.

*-e, --energy*

Thêm năng lượng cho vector đặc trưng

*-w, --window=str*

Xác định các cửa sổ dạng sóng. Cửa sổ có sẵn là ‘Hamming’, ‘Hanning', `Blackman' hay không có gì. Mặc định là của số ‘Hamming’

*-n, --num-filters=n*

Thiết lập số kênh trong filter bank. Mặc định là: 24

*-a, --alpha=f*

Thiết lập ràng buộc tần số thấp đến f Hz. Mặc đinh: 0 Hz

*-u, --freq-max=f*

Thiết lập tần số giới hạn trên đến f Hz. Mặc định: Niquist.

*-b, --fft-length=n*

Thiết lập độ dài FFT tới n mẫu. Mặc đinh: 512

*-D, --delta*

Thêm các dẫn xuất đầu tiên đến các vector đặc trưng

*-A, --acceleration*

Thêm các dẫn xuất tự thứ hai với vector đặc trưng. Yêu cầu *`--delta'.*

*-v, --verbose*

Bật chế độ verbose

*-h, --help*

In thông báo và giúp các công cụ thoát

*-V, --version*

In thông tin phiên bản và thoát..

*Inputfile*: Đầu vào là đường dẫn đến file tín hiệu âm thanh

*Outputfile:* Đầu ra của file âm thanh

1. Loại bỏ khoảng lặng

Khi tất cả các vector đặc trưng được tính toán, một bước rất quan trọng là quyết định vector đó sử dụng được hay không. Một cách để phát hiện ra vấn đề đó là so sánh với những khung có khoảng lặng. Vì vậy, để chọn các khung tương ứng với tín hiệu, sử dụng một phương pháp tiếp cận dựa trên sự phân bố năng lượng của mỗi tín hiệu. Để chuẩn hóa các hệ số năng lượng, chạy lệnh *NormFeat* từ thư viện *LIA SpkDet*. Câu lệnh thực hiện như sau:

*NormFeat [option] --inputFeatureFileName inputfile*

*NormFeat* nhằm mục đích xử lý tín hiệu đầu vào liên quan đến đặc trưng bằng cách áp dụng bất kỳ tiêu chuẩn hóa

*inputfile:* là tên của file cần làm việc, là một danh sách, có phần mở rộng là *.lst*

*options:* là các tùy chọn.

Giá trị tùy chọn đượcmô tả ở bảng dưới đây:

Bảng 2.1: Mô tả tùy chọn sử dụng tiêu chuẩn hóa năng lượng

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Giá trị** | **Mô tả** |
| mode | .norm | Xác định các loại tiêu chuẩn hóa( Định mức đặc trưng bình thường là phân bố gaussion trên 1,0) |
| vectSize | 60 | Xác định kích thước của các đặc trưng |
| loadFeatureFileFormat  saveFeatureFileFormat | SPRO4 | Định dạng gọi, lưu các đặc trưng của file |
| loadFeatureFileExtension | .prm | Định dạng phần mở rộng để gọi đặc trưng của file (tương ứng với đầu ra của các bước trích chọn đặc trưng) |
| saveFeatureFileExtension | .enr.prm | Định dạng phần mở rộng để lưu đặc trưng của file |
| featureFilesPath | ./prm/ | Định dạng đường dẫn để lưu và gọi file đặc trưng |
| writeAllFeatures | True | Khi thiết lập đúng, tất cả các tính năng về tập tin ban đầu được ghi |

Sau đó, để chọn các khung với năng lượng cao nhất, sử dụng lệnh *EnergyDetector* của thư viện *LIA SpkDet*. Câu lệnh thực hiện như sau:

*EnergyDetector.exe [options] --inputFeatureFileName inputfile*

*EnergyDetector* nhằm mục đích phân tích thành phần năng lượng đặc trưng đầu vào bằng cách cung cấp nhãn năng lượng đầu ra ở mức cao. Đây là một kiểu phát hiện tiếng nói/ không nói

*Inputfile*: là tên của đặc trưng mà năng lượng đã được tiêu chuẩn hóa.

*options:* là các tùy chọn.

Giá trị tùy chọn đượcmô tả ở bảng dưới đây:

Bảng 2.2: Mô tả tùy chọn sử dụng phát hiện tín hiệu tiếng nói

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Giá trị** | **Mô tả** |
| labelOutputFrames | Speech | Nhãn ảnh hưởng đến khung được chọn |
| LabelFilesPath | ./lbl | Xác định đường dẫn lưu trữ file |
| saveLabelFileExtension | .lbl | Lưu trữ các file mở rộng |
| Alpha | 0.0 | Tỷ lệ khung hình được chọn trong gaussian trung tâm |
| nbTrainIt | 10 | Số bước lặp của EM để ước lượng năng lượng phân phối |
| baggedFrameProbabilityInit | 0.001 | Xác định tỷ lệ giữa các khung hình được lựa chọn và tổng số các khung được sử dụng để khởi tạo |

1. Chuẩn hóa đặc trưng

Các vector tham số được chuẩn hóa để phù hợp với trung bình bằng không và các thuộc tính không phù hợp. Giá trị trung bình và phương sai được sử dụng cho chuẩn hóa được tính bằng tập tin trên tổng số các khung sau khi xử lý loại bỏ khung.

Việc chuẩn hóa đặc trưng cũng giống như sử dụng chuẩn hóa năng lượng. Chỉ khác một điểm khi chạy lệnh *NormFeat* là gọi nhãn file. Vì vậy, để thực hiện các chuẩn hóa năng lượng, *NormFeat* chỉ sử dụng các khung với năng lượng cao nhất. Bảng dưới đề cập đến những thay đổi so với chuẩn hóa năng lượng:

Bảng 2.3: Mô tả tùy chọn chuẩn hóa đặc trưng

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Giá trị** | **Mô tả** |
| saveFeatureFileExtension | .norm.prm | Xác định định dạng để lưu file |
| labelSelectFrames | speech | Xác định nhãn để làm việc |
| LabelFilePath | ./prm | Đường dẫn để lưu file |

1. Huấn luyện mô hình nền

Mục đích của bước này là để tạo ra một mô hình chung nhất của tất cả người nói. Mô hình này được lựa chọn từ một số người nói sau đó huấn luyện một mô hình duy nhất được gọi là mô hình nền. Mô hình GMM được ước tính bằng cách sử dụng thuật toán EM (tối đa hóa kỳ vọng).

Huấn luyện mô hình nền, thực thi bởi lệnh *TrainWorld* của thư viện *LIA\_SpkDet.* Câu lệnh được thực hiện như sau:

*TrainWorld [options] --inputFeatureFileName inputfile*

*--outputWorldFilename worldfile*

*TrainWorld:* học mô hình GMM thông qua thuật toán EM.

*inputfile:* là tên của các chuẩn hóa đặc trưng để làm việc với nó và nó có thể là một danh sách mở rộng, có đuôi là .*lst*

*worldfile:* là tên file kết quả từ mô hình.

*options:* là các tùy chọn.

Giá trị tùy chọn đượcmô tả ở bảng dưới đây:

Bảng 2.4: Mô tả tùy chọn chính sử dụng huấn luyện mô hình từ

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Ví dụ** | **Mô tả** |
| distribType | GD | Xác định kiểu của thuộc tính |
| saveLabelFileExtension | .gmm | Định dạng để lưu thuộc tính |
| loadFeatureFileExtension | .norm.prm | Định dạng đầu vào của file |
| MixtureDistribCount | 20 | Xác định số thuộc tính gaussion trong hỗn hợp |
| mixtureFilePath | ./gmm | Đường dẫn để lưu thư mục file |
| baggedFrameProbabilityInit | 0.0005 | Xác định tỷ lệ giữa các khung được lựa chọn và tổng số các khung được sử dụng để khởi tạo. |
| baggedFrameProbability | 0.01 | Xác định tỷ lệ giữa các khung hình được lựa chọn và tổng số các khung được sử dụng để huấn luyện |
| nbTrainIt | 25 | Số lần lặp của EM liên quan đến baggedFrameProbabilityInit |
| nbTrainFinalIt | 4 | Số lần lặp của EM không có baggedFrameProbability |
| featureServerMask | 0-15, 17-33 | Được sử dụng để chọn một tập hợp các đặc trưng trong vector |
| vectSize | 33 | Xác định kích thước của các đặc trưng |
| NormalizeModel | true | Áp dụng N(0, 1) sắp xếp ở cuối quá trình huấn luyện |

1. Huấn luyện mô hình đích (Target model training)

Để huấn luyện mô hình đích, sử dụng phương pháp sau đây: xây dựng mô hình người nói xuất phát từ việc điều chỉnh các thông số của mô hình nền bằng cách sử dụng tiếng nói trong giai đoạn huấn luyện và cực đại hóa xác suất hậu nghiệm (MAP- *Maximum A Posterior)*. Hệ số năng lượng được loại bỏ để xây dựng mô hình.

Huấn luyện mô hình đích, chạy lệnh *TrainTarget* của gói *LIA\_SpkDet.* Sử dụng câu lệnh như sau:

*TrainTarget [options] --targetIdList inputfile*

*--inputWorldFilename worldfile*

*TrainTarget* nhằm mục đích huấn luyện đích người nói bằng cách làm phù hợp mô hình từ thông qua phương pháp MAP.

*Inputfile* là danh sách đầu, chính là danh sách các tên file đưa vào huấn luyện

*Worldfile: world -- debug false –verbose true*

*options:* là các tùy chọn.

Giá trị tùy chọn đượcmô tả ở bảng dưới đây:

Bảng 2.5: Mô tả tùy chọn trong huấn luyện mô hình đích

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Giá trị** | **Mô tả** |
| baggedFrameProbability | 0.1 | Xác định tỷ lệ giữa các khung được lựa chọn và tổng số các khung được sử dụng để thích ứng |
| nbTrainIt | 5 | Số lần lặp EM liên quan tới baggedFrameProbability |
| nbTrainFinalIt | 1 | Số lần lặp EM không liên quan tới baggedFrameProbability |
| targetIdList | Ndx | Danh sách đầu vào: mô hình trên cột đầu tiên, nhập vào tên tập tin đặc trưng về những người khác nhau |
| MAPAlgo | MAPOccDep | Xác định phương pháp thích ứng để sử dụng: MAPOccDep hoặc MAPConst |
| MAPRegFactor | 10 | Tham số được sử dụng bởi những kỹ thuật thích ứng MAPOccDep |

Phương thức MAP sử dụng là phương pháp tiếp cận MAPOCCDep: biến ngẫu nhiên để ước lượng được tính bằng sự kết hợp tuyến tính các giá trị trong mô hình nền với dữ liệu trên thuật toán EM. Phương pháp này sẽ đưa vào một xác suất hậu nghiệm cho mỗi Gaussion. Trọng số của sự kết hợp này được cung cấp bởi các tùy chọn MAPREGFactor r

1. Thử nghiệm

Mục tiêu của bước này là tính toán điểm số cho từng thử nghiệm của các đặc trưng vector để đưa ra một mô hình đích và mô hình nền (số điểm là xác suất tính được qua thử nghiệm từng người nói). Để tính toán số điểm này, chỉ xem xét 10 thuộc tính Gaussian đầu tiên của mô hình. Câu lệnh được thực hiện như sau:

*ComputeTest [options] --ndxFileName inputfile*

*--worldModelName worldfile*

*--outputFile outputfile*

*ComputeTest:* đưa ra điểm số liên quan đến một phần thử nghiệm và mô hình đích.

*Inputfile:* là đưa ra danh sách các file thử nghiệm người nói

*Worldfile:* là tên của mô hình nền

*Outputfile:* là kết quả ghi ra tập tin

*options:* là các tùy chọn.

Giá trị tùy chọn đượcmô tả ở bảng dưới đây:

Bảng 2.6: Mô tả tùy chọn tính điểm số của mô hình GMM

|  |  |  |
| --- | --- | --- |
| **Tùy chọn** | **Giá trị** | **Mô tả** |
| topDistribsCount | 10 | Số Gaussians sử dụng để tính toán điểm số |
| computeLLKWithTopDistribs | COMPLETE | Tính toán với phân phối đầu |
| Gender | M | Giới tính của tập tin ndx |
| loadMixtureFileExtension | .gmm | Xác định phần mở rộng để gọi mô hình |
| loadFeatureFileExtension | .norm.prm | Xác định phần mở rộng để gọi đặc trưng |
| mixtureFilesPath | ./gmm | Xác định đường dẫn để gọi hỗn hợp |
| outputFile | test1.res | Kết quả điểm số tập tin |

CHƯƠNG 3. TRIỂN KHAI HỆ THỐNG THỬ NGHIỆM NHẬN DẠNG LÀN ĐIỆU CHÈO VÀ DÂN CA QUAN HỌ

* 1. Sơ đồ tổng quan quá trình xây dựng hệ thống nhận dạng làn điệu chèo và dân ca quan họ

|  |
| --- |
| Chia dữ liệu thành 5 phần bao gồm 4 phần để train và 1 phần để test  Chuẩn bị dữ liệu  Thực hiện train và phân dữ liệu test đầu vào thành 2 lớp: chèo và quan họ  Đưa những file test có kết quả phân lớp theo thể loại đúng vào nhận diện theo làn điêụ cụ thể  Tổng hợp kết quả |

Hình 3.1. Hệ thống phân lớp làn điệu chèo và dân ca quan họ

Hình 3.1 là các bước thực hiện của hệ thống phân lớp làn điệu được thử nghiệm trong đề tài. Bước đầu tiên là việc chuẩn bị dữ liệu, dữ liệu này là các file âm thanh có định dạng là .wav, mono, với tần số 16000Hz, được cắt bỏ phần nhạc dạo bằng phần mềm Wavepad Sound Editor. Danh sách các làn điệu sẽ được trình bày cụ thể ở mục 3.2.1 của chương này. Sau khi chạy chương trình thử nghiệm trên bộ công cụ ALIZE sẽ cho ra số(Score) của file âm thanh đưa vào test với danh sách các làn điệu đã được train. File âm thanh này sẽ được xếp vào làn điệu nào được đánh điểm cao nhất.

* 1. Chuẩn bị dữ liệu

Chương trình được thử nghiệm trên 23 làn điệu chèo và 23 làn điệu quan họ. Thông tin của toàn bộ các làn điệu được mô tả trong bảng dưới đây:

Bảng 3.1: Thông tin về 23 làn điệu chèo

|  |  |  |
| --- | --- | --- |
| STT | Tên làn điệu | Tên viết tắt |
| 01 | Bình thảo | CH\_BT |
| 02 | Cấm giá | CH\_CG |
| 03 | Chinh phụ | CH\_CP |
| 04 | Chức cẩm hồi văn | CH\_CCHV |
| 05 | Con nhện giăng mùng | CH\_CNGM |
| 06 | Đò đưa | CH\_DD |
| 07 | Đào liễu | CH\_DL |
| 08 | Đào lý một cành | CH\_DLMC |
| 09 | Duyên phận phải chiều | CH\_DPPC |
| 10 | Đường trường bắn thước | CH\_DTBT |
| 11 | Đường trường tiếng đàn | CH\_DTTK |
| 12 | Đường trường thu không | CH\_DTTK |
| 13 | Đường trường trên non | CH\_DTTN |
| 14 | Đường trường vị thủy | CH\_DTVT |
| 15 | Du xuân | CH\_DXN |
| 16 | Dương xuân | CH\_DX |
| 17 | Lới lơ | CH\_LL |
| 18 | Luyện năm cung | CH\_LNC |
| 19 | Nón thúng quai thao | CH\_NTQT |
| 20 | Quá giang | CH\_QG |
| 21 | Quạt màn | CH\_QM |
| 22 | Quân tử vu dịch | CH\_QTVD |
| 23 | Sắp qua cầu | CH\_SQC |

Bảng 3.2: Thông tin về 23 làn điệu quan họ

|  |  |  |
| --- | --- | --- |
| STT | Tên làn điệu | Tên viết tắt |
| 01 | Ba quan | QH\_BQ |
| 02 | Còn duyên | QH\_CD |
| 03 | Cổ tay vừa trắng vừa tròn | QH\_CTVTVT |
| 04 | Đôi bên bác mẹ cùng già | QH\_DBBMCG |
| 05 | Đối ca sông cầu | QH\_DCSC |
| 06 | Gió đưa cây cải | QH\_GDCC |
| 07 | Giăng thanh gió mát | QH\_GTGM |
| 08 | Lý giao duyên | QH\_LGD |
| 09 | Lòng vẫn đợi chờ | QH\_LVDC |
| 10 | Nguyệt gác mái đình | QH\_NGMD |
| 11 | Năm liệu bảy lo | QH\_NLBL |
| 12 | Nhớ mãi khôn nguôi | QH\_NMKN |
| 13 | Người ơi đến hẹn lại về | QH\_NODHLV |
| 14 | Se chỉ luồn kim | QH\_SCLK |
| 15 | Súc miệng ấm đồng | QH\_SMAD |
| 16 | Tưởng đến gần xa | QH\_TDGX |
| 17 | Tìm em trong chiều hội Lim | QH\_TETCHL |
| 18 | Thân lươn bao quản lấm đầu | QH\_TLBQLD |
| 19 | Tương phùng tương ngộ | QH\_TPTN |
| 20 | Trên rừng ba mươi sáu thứ chim | QH\_TRBMSTC |
| 21 | Tay tiên chuốc chén rượu đào | QH\_TTCCRD |
| 22 | Vui bốn mùa | QH\_VBM |

File nhạc sau khi được tải về từ trên mạng sẽ cắt và chuyển về định dạng file Wav. Với việc tiến hành phân lớp theo thể loại(chèo hay quan họ) thì sẽ được huấn luyện với 23\*15 = 345 file âm thanh, sau đó khi tiến hành phân lớp theo làn điệu thì mỗi làn điệu sẽ được huấn luyện với 15 file âm thanh.

Kết quả đã tạo được cơ sở dữ liệu âm nhạc phục vụ cho quá trình phân lớp làn điệu chèo và dân ca quan họ. Cơ sở dữ liệu này được đặt trong thư mục **wav**: gồm có 1,27GB dữ liệu âm nhạc.

Với mỗi làn điệu, định dạng file được được lưu dựa theo tên viết tắt được đề cập trong hai bảng liệt kê danh sách làn điệu chèo(bảng 3.1) và làn điệu dân ca quan họ(bảng 3.2) phía trên. Định dạng file được tổ chức như sau:

CH/QH\_Chữ cái bắt đầu của từng chữ trong tên làn điệu-Thứ tự trong của file gốc trong danh sách các file của một làn điệu-Vị trí của phần được cắt ra trong file gốc.

Trong đó:

* CH: Ký hiệu là file chèo
* QH: Ký hiệu là file dân ca quan họ

Ví dụ: làn điệu Bình Thảo có các file mp3 gốc là:

* 1\_BinhThao-ThuHuyen\_ThanhTam.mp3
* 2\_BinhThao-MinhToan.mp3
* 3\_BinhThao-DoanThanhBinh.mp3

Từ file 1\_BinhThao-ThuHuyen\_ThanhTam.mp3, ta cắt ra được 3 file wav, thì định dạng tên của từng file sẽ là: CH\_BT-01-01, CH\_BT-01-02, CH\_BT-01-03. Tương tự với các file khác ta sẽ được là CH\_BT-02-01, CH\_BT-02-02,…

Dữ liệu được mô tả ở trên được chia thành 2 tập:

* **Tập dữ liệu dành để huấn luyện:** gồm dữ liệu của 46 làn điệu(23 làn điệu chèo, 23 làn điệu quan họ), mỗi làn điệu sẽ có 12 file được đưa vào huấn luyện. tổng cộng là 552 file
* **Tập dữ liệu để Test:** Lấy 4 file trong mỗi làn điệu để làm dữ liệu test

Các file được dùng để huấn luyện và test được luân phiên đổi chỗ cho nhau. Ví dụ 15 file của một làn điệu sẽ được chia ra làm 5 phần: phần 1, phần 2, phần 3, phần 4, phần 5. Đầu tiên sẽ lấy phần 1 làm test, còn các phần 2, 3, 4, 5 sẽ đưa vào huấn luyện, sau khi thu được kết quả thì lại lấy phần 2 làm test, các phần 1, 3, 4, 5 đưa vào huấn luyện,… Thực hiện liên tục cho đến khi tất cả các file đều đã được test, thì tiến hành tổng hợp kết quả.

* + 1. Chuẩn bị dữ liệu cho phân lớp theo thể loại bởi công cụ ALIZE

1. Tạo file danh sách chứa trong thư mục lst

Các file cần chuẩn bị để đưa vào dữ liệu huấn luyện và thử nghiệm là: *all.lst, all1.lst, TEN\_THE\_LOAItrain, TEN\_THE\_LOAItest, world.lst, world.weight*. Sau đây là nội dung của mỗi file cần ghi lại:

* ***all.lst***: file này chứa danh sách tất cả các file là danh sách các file có đuôi .wav, nó là tham số đầu vào của hệ thống. Định dạng mỗi file được ghi trên một dòng

Ví dụ:

*CH\_BT-01-01.wav*

*CH\_BT-01-02.wav*

*CH\_BT-01-03.wav*

*CH\_BT-02\_01.wav*

*……………*

*QH\_VBM-05-04.wav*

*QH\_VBM-06-02.wav*

*QH\_VBM-06\_03.wav*

*QH\_VBM -06\_04.wav*

* ***all1.lst****:* file này là file *all.lst* nhưng được xóa bỏ phần đuôi .wav

Ví dụ:

*CH\_BT-01-01*

*CH\_BT-01-02*

*CH\_BT-01-03*

*CH\_BT-02\_01*

*……………*

*QH\_VBM-05-04*

*QH\_VBM-06-02*

*QH\_VBM-06\_03*

*QH\_VBM -06\_04*

- ***TEN\_THE\_LOAItrain.lst***: file này chứa tất cả tên file dữ liệu (file .wav) của các file nhạc đưa vào để huấn luyện(mỗi làn điệu chia các file âm nhạc thành 5 phần lấy 4 phần dữ liệu để đưa vào huấn luyện), định dạng tên mỗi file lưu trên một dòng. Ở đây dùng để huấn luyện nhận diện theo thể loại nên chỉ có 2 file CH\_train.lst và QH\_train.lst. Nội dung file sẽ như sau:

Ví dụ: File *CHtrain.lst*:

*CH\_BT-02-02*

*CH\_BT-02-03*

*CH\_BT-03-01*

*CH\_BT-03-02*

*CH\_BT-03-03*

*…*

*CH\_XX-10-01*

*CH\_XX-10-02*

- ***TEN\_THE\_LOAItest.lst***: file này chứa tất cả tên file dữ liệu (file .wav) của các file âm thanh đưa vào để huấn luyện(mỗi làn điệu lấy 1 phần dữ liệu để đưa vào thử nghiệm), định dạng tên mỗi file được ghi trên một dòng. Tương tự như bên trên nội dung file sẽ như sau:

Ví dụ: File *CHtest.lst*:

*CH\_BT-01-01*

*CH\_BT-01-02*

*CH\_BT-01-03*

*CH\_BT-02-01*

*…*

*CH\_XX-02-01*

*CH\_XX-02-02*

* ***world.lst:*** nội dung của file này là chỉ đường dẫn đến tất cả các danh sách file âm thanh được dùng để huấn luyện (chính là file *TEN\_THE\_LOAItrain.lst* đã trình bày ở trên). Mỗi tên file được lưu trên một dòng.

Ví dụ:

*D:\Alize\lst\CHtrain.lst*

*D:\Alize\lst\QHtrain.lst*

* ***world.weight***: file trọng số, vì ở đây chỉ phân theo 2 lớp(chèo hoặc quan họ) nên trọng số sẽ là 0.5.

Ví dụ:

*0.5*

*0.5*

1. Tạo file chứa danh sách các file được huấn luyện trong thư mục ndx

* ***TheLoai.ndx:*** file này chứa chính nội dung của file *TheLoai\_train.lst* nhưng nội dung được ghi trên một dòng, có khoảng cách giữa các file và có định dạng TenTheLoai\_gmm đầu của mỗi dòng.

Ví dụ: file ch.ndx sẽ cấu trúc như sau:

*CH\_gmm CH\_BT-01-01 CH\_BT-01-02 CH\_BT-01-03 CH\_BT-02-01 CH\_BT-03-03 CH\_BT-04-02 CH\_BT-04-03 CH\_BT-05-02 CH\_BT-05-03 CH\_BT-06-01 CH\_BT-06-02 CH\_BT-07-01 CH\_BT-07-02 CH\_BT-08-01 CH\_BT-08-02 CH\_BT-08-03 CH\_CCHV-03-04 CH\_CCHV-04-02 CH\_CCHV-04-03 CH\_CCHV-05-02 CH\_CCHV-06-04*

*........* *CH\_XX-07-01 CH\_XX-07-02 CH\_XX-08-01 CH\_XX-08-02 CH\_XX-09-01 CH\_XX-09-02 CH\_XX-10-01 CH\_XX-10-02*

1. Tạo các file để thử nghiệm trong thư mục test

* **TenLanDieu.ndx:** file này chính là các file đưa vào thử nghiệm của chương trình, tên của các file là nội dung của file CHtest.lst và QHtest.lst trong thư mục lst. Nó có định dạng như sau: *TenLanDieu TenFileGMM*

*+ TenLanDieu:* là tên các file đưa vào kiểm tra nhưng không có đuôi .wav. Đối với việc chia 5 phần và lấy một phần để thử nghiệm thì trong thư mục này sẽ có 1000/5 = 200 file

*+ TenFileGMM:* Danh sách thể loại và có thêm định dạng \_gmm.

Ví dụ: file *CH\_BT-02-02.nd*x có nội dung là:

*CH\_BT-02-02 ch\_gmm qh\_gmm*

* + 1. Chuẩn bị dữ liệu cho phân lớp theo làn điệu cụ thể bởi công cụ ALIZE

Từ kết quả của việc phân thành hai lớp theo thể loại ở bước trên ta tiến hành chuẩn bị dữ liệu cho phân lớp theo làn điệu cụ thể.

1. Tạo file danh sách chứa trong thư mục lst\_ch và lst\_qh

Từ nội dung của các file có định dạng TEN\_THE\_LOAItrain.lst trong thư mục lst sẽ tạo ra các file chứa nội dung những file được đưa vào huấn luyện tương ứng trong thư mục lst\_ch và lst\_qh có định dạng tên là TEN\_LAN\_DIEUtrain.lst. Ngoài ra còn có các file world.lst và word.weight. Với lst\_ch sẽ chứa nội dung những file được đưa vào huấn luyện để phân lớp theo làn điệu chèo còn đối với thư mục lst\_qh sẽ chứa những file được đưa vào để huấn luyện nhằm mục đích phân lớp theo làn điệu quan họ.

Dưới đây là nội dung các file được ghi lại:

- ***TEN\_LAN\_DIEUtrain.lst***: file này chứa tất cả tên file dữ liệu (file .wav) của các file nhạc đưa vào để huấn luyện(mỗi làn điệu chia các file âm nhạc thành 5 phần lấy 4 phần dữ liệu để đưa vào huấn luyện), định dạng tên mỗi file lưu trên một dòng. Nội dung file sẽ như sau:

Ví dụ: File ***CH\_BTtrain.lst***trong thư mục***lst\_ch***sẽ có 16 dòng như bên dưới(mỗi làn điệu có 20 file, lấy 4 phần nên có 16 dòng):

*CH\_BT-01-01*

*CH\_BT-01-02*

*CH\_BT-01-03*

*CH\_BT-02-01*

*CH\_BT-02-02*

*CH\_BT-02-03*

*CH\_BT-03-01*

*CH\_BT-03-02*

*CH\_BT-03-03*

*CH\_BT-04-02*

*CH\_BT-04-03*

*CH\_BT-05-02*

*CH\_BT-07-02*

*CH\_BT-08-01*

*CH\_BT-08-02*

*CH\_BT-08-03*

* ***world.lst:*** nội dung của file này là chỉ đường dẫn đến tất cả các danh sách âm thanh được dùng để huấn luyện (chính là file *TEN\_LAN\_DIEUtrain.lst* đã trình bày ở trên). Mỗi tên file được lưu trên một dòng.

Ví dụ: với 25 làn điệu chèo thì file world.lst trong thư mục lst\_ch sẽ có nội dung như sau:

*D:\Alize\lst\_ch\CH\_BTtrain.lst*

*D:\Alize\lst\_ch\CH\_CCHVtrain.lst*

*D:\Alize\lst\_ch\CH\_CGtrain.lst*

*D:\Alize\lst\_ch\CH\_CNGMtrain.lst*

*D:\Alize\lst\_ch\CH\_CPtrain.lst*

*D:\Alize\lst\_ch\CH\_DDtrain.lst*

*D:\Alize\lst\_ch\CH\_DLMCtrain.lst*

*D:\Alize\lst\_ch\CH\_DLtrain.lst*

*D:\Alize\lst\_ch\CH\_DPPCtrain.lst*

*D:\Alize\lst\_ch\CH\_DTBTtrain.lst*

*D:\Alize\lst\_ch\CH\_DTTDtrain.lst*

*D:\Alize\lst\_ch\CH\_DTTKtrain.lst*

*D:\Alize\lst\_ch\CH\_DTTNtrain.lst*

*D:\Alize\lst\_ch\CH\_DXNtrain.lst*

*D:\Alize\lst\_ch\CH\_DXtrain.lst*

*D:\Alize\lst\_ch\CH\_LLtrain.lst*

*D:\Alize\lst\_ch\CH\_LNCtrain.lst*

*D:\Alize\lst\_ch\CH\_NTQTtrain.lst*

*D:\Alize\lst\_ch\CH\_QGtrain.lst*

*D:\Alize\lst\_ch\CH\_QMtrain.lst*

*D:\Alize\lst\_ch\CH\_QTVDtrain.lst*

*D:\Alize\lst\_ch\CH\_SLCtrain.lst*

*D:\Alize\lst\_ch\CH\_SQCtrain.lst*

*D:\Alize\lst\_ch\CH\_TVtrain.lst*

*D:\Alize\lst\_ch\CH\_XXtrain.lst*

* ***world.weight***: file trọng số, giá trị bằng 1/SoLanDieu

Ví dụ với 25 làn điệu chèo thì nội dung file world.weight trong thư mục lst\_ch sẽ có nội dung:

*0.04*

*0.04*

*0.04*

*….*

*0.04*

*0.04*

*0.04*

* 1. Nhận dạng người nói sử dụng ALIZE

Các bước thực hiện nhận dạng người nói sử dụng công cụ ALIZE được mô tả như hình dưới đây:

Tạo thư mục làm việc

Tính tham số MFCC

Dò tìm năng lượng

Phát hiện tiếng nói trong tín hiệu

Chuẩn hóa các tham số của tín hiệu

Chuẩn hóa TrainWorldInit

Chuẩn hóa TrainWorldFinal

Huấn luyện GMM cho từng người nói

Tính điểm số của từng người nói

Hình 3.3. Các bước xây dựng hệ nhận dạng người nói sử dụng SPro & ALIZE

*Sau đây là hướng dẫn chi tiết các bước thực hiện lệnh chạy của chương trình:*

* + 1. Bước 1: Tạo thư mục làm việc

Tạo các thư mục làm việc để chương trình chạy chứa các file dữ liệu trong quá trình làm, các file có tên như sau: *lst, prm, lbl, gmm, cfg, ndx và input*. File: “input”chứa các file .wav có thể đặt cùng thư mục hoặc ngoài thư mục này.

* + 1. Bước 2: Tính tham số MFCC

Tạo ra các vector đặc trưng từ các tín hiệu tiếng nói, sử dụng công cụ *sfbcep* để tính các tham số MFCC. Điều kiện cần thiết để tạo được tham số MFCC là phải chuẩn hóa các file .wav thành các file .prm. Tạo thư mục có tên là “prm“ để chương trình chạy ghi lại các file .prm

* Đầu vào: file tiếng nói có định dạng \*.wav
* Đầu ra: file prm có chứa tham số MFCC

Sử dụng câu lệnh:

*sfbcep -F PCM16 -f 16000 -p 19 -e -D -A input/AnhDT\_09\_01.wav prm/AnhDT\_09\_01.prm*

Ý nghĩa của câu lệnh như sau:

*-F PCM16*: sử dụng định dạng hiển thị sóng đầu vào của file là : PCM16

*-f 16000*: sử dụng tần số cho sóng đầu vào là: 16000F

*-p 19*: có hệ số Cepstral là 19

*-e*: sử dụng thêm năng lượng cho vector đặc trưng

*-D*: Thêm các dẫn xuất đầu tiên đến các vector đặc trưng

*-A*: Thêm các dẫn xuất thứ hai với vector đặc trưng

🡺 Kết quả : sinh ra file *AnhDT\_09\_01.prm*

* + 1. Bước 3: Dò tìm năng lượng
* Đầu vào: file *all1.lst*
* Đầu ra: file *.enr.prm* được lưu trong thư mục *prm*

Sử dụng câu lệnh:

*NormFeat --config ./cfg/NormFeat\_energy.cfg --inputFeatureFilename ./lst/all1.lst*

🡺 Kết quả : sinh ra file *AnhDT\_09\_01.enr.prm, AnhDT\_09\_02.enr.prm…*

* + 1. Bước 4: Phát hiện tiếng nói trong tín hiệu
* Đầu vào: file *all1.lst*
* Đầu ra: file *.lbl* được lưu trong thư mục *lbl*

Sử dụng câu lệnh:

*EnergyDetector --config ./cfg/EnergyDetector.cfg --inputFeatureFilename ./lst/all1.lst --verbose true --debug false*

🡺 Kết quả : sinh ra file *AnhDT\_09\_01.lbl, AnhDT\_2Tu\_67\_4.lbl…*

Nội dung của file *AnhDT\_2Tu\_67\_4.lbl*:

*0 0 speech*

*0.27 0.34 speech*

*0.37 0.63 speech*

*1.46 1.67 speech*

* + 1. Bước 5: Chuẩn hóa các tham số của tín hiệu
* Đầu vào: file *all1.lst*
* Đầu ra: file *.norm.prm* được lưu trong thư mục prm

Sử dụng câu lệnh:

*NormFeat --config ./cfg/NormFeat.cfg --inputFeatureFilename ./lst/all1.lst*

🡺 Kết quả : sinh ra file *AnhDT\_09\_01.lbl, AnhDT\_2Tu\_67\_4.norm.prm…*

* + 1. Bước 6: Chuẩn hóa TrainWorldInit
* Đầu vào: file *world.lst* và file *world.weight*.
* Đầu ra: file *worl\_init.gmm, world\_initinit.gmm.*

*Sử dụng câu lệnh:*

*TrainWorld --config ./cfg/TrainWorldInit.cfg --inputStreamList ./lst/world.lst --weightStreamList ./lst/world.weight --outputWorldFilename world\_init --debug false --verbose true*

* + 1. Bước 7: Chuẩn hóa TrainWorldFinal
* Đầu vào: file *world.lst* và file *world.weight*
* Đầu ra: file *world.gmm*

Sử dụng câu lệnh:

*TrainWorld --config ./cfg/TrainWorldFinal.cfg --inputStreamList ./lst/world.lst --weightStreamList ./lst/world.weight --outputWorldFilename world --inputWorldFilename world\_init --debug false --verbose true*

* + 1. Bước 8: Huấn luyện GMM cho từng người nói

Đầu vào: file *TenNguoiNoi.ndx*

Đầu ra: file *TenNguoiNoi\_gmm.gmm*

Câu lệnh thực hiện:

*TrainTarget --config ./cfg/target\_dc.cfg --targetIdList ./ndx/AnhDT.ndx --inputWorldFilename world --debug false --verbose true*

🡺 Kết quả : sinh ra file *AnhDT\_gmm, BayNVN\_gmm, CaoNV\_gmm, ChienDT\_gmm…*

* + 1. Bước 9: Nhận dạng người nói
* Đầu và: file *testSpeaker\_TenNguoiNoi.ndx*
* Đầu ra là: file *TenNguoiNoi.res*

Câu lệnh thực hiện:

*ComputeTest --config cfg/target\_seg\_male.cfg --ndxFilename ndx/testSpeakerAnhDT.ndx --outputFilename AnhDT.res --inputWorldFilename world --channelCompensation AnhDT*

🡺 Kết quả : sinh ra file *AnhDT.res, BayNVN.res, CaoNV.res, ChienDT.res..*

Ví dụ: file kết quả *AnhDT.res* như sau:

*M AnhDT\_gmm 1 AnhDT\_2Tu\_01\_1 0.66308*

*M BayNVN\_gmm 0 AnhDT\_2Tu\_01\_1 -1.15561*

*M CaoNV\_gmm 0 AnhDT\_2Tu\_01\_1 -0.328454*

*M ChienVC\_gmm 0 AnhDT\_2Tu\_01\_1 -0.331204*

*M ChiLT\_gmm 0 AnhDT\_2Tu\_01\_1 -0.939304*

*M DiepDT\_gmm 0 AnhDT\_2Tu\_01\_1 -0.70246*

*M DungDT\_gmm 0 AnhDT\_2Tu\_01\_1 -0.309592*

*M DungPTN\_gmm 0 AnhDT\_2Tu\_01\_1 -1.4819*

*M DuongDD\_gmm 0 AnhDT\_2Tu\_01\_1 -1.61751*

*M HaiPN\_gmm 1 AnhDT\_2Tu\_01\_1 0.0911932*

*M HangDT\_gmm 0 AnhDT\_2Tu\_01\_1 -0.890401*

*M HanhTT\_gmm 1 AnhDT\_2Tu\_01\_1 0.199216*

*M HiepTV\_gmm 0 AnhDT\_2Tu\_01\_1 -0.659496*

*……*

Mỗi một file âm thanh sẽ ghi ra điểm số (score) tương ứng với N người nói. Điểm số của người nào lớn nhất trong N người nói chính là người mà hệ thống cho ra kết quả cần tìm.

* 1. Nhận dạng người nói phụ thuộc từ khóa tiếng Việt
     1. Nhận dạng người nói sử dụng ALIZE và Sphinx3
* *PAlize:* lần lượt là 48 giá trị điểm số được lưu ở các file tổng hợp là kết quả chạy qua bộ công cụ ALIZE
* *alpha:* giá trị thử nghiệm
* *PSphinx:* lần lượt là 48 giá trị điểm số được lưu ở các file tổng hợp là kết quả chạy tính xác suất qua bộ công cụ Sphinx3

***Các thử nghiệm lần lượt được thực hiện như sau:***

* **Test 1:** Thử nghiệm với dữ liệu lấy từ xác suất tiên nghiệm lấy từ mô hình GMM (chạy qua bộ công cụ ALIZE ) và mô hình HMM (chạy qua bộ công cụ Sphinx) của từng người nói

***Công thức:***

*SpeakerMax = PAlize+alpha\*PSphinx* (3.1)

***Phương pháp thực hiện:***

Chương trình chạy thử nghiệm ra file kết quả tổng hợp gồm kết quả điểm số chạy qua bộ công cụ ALIZE và Sphinx. Sau đó sẽ thử nghiệm các giá trị alpha khác nhau để tìm ra kết quả nhận dạng người nói đạt tỷ lệ cao nhất theo công thức trên.

* **Test 2:** Thử nghiệm với dữ liệu lấy từ xác suất tiên nghiệm lấy từ mô hình GMM (chạy qua bộ công cụ ALIZE) và xác suất hậu nghiệm lấy từ mô hình HMM (chạy qua bộ công cụ Sphinx) của từng người nói

***Công thức:***

** (3.2)

***Phương pháp thực hiện:***

Chương trình chạy thử nghiệm ra file kết quả tổng hợp gồm kết quả điểm số chạy qua bộ công cụ ALIZE và Sphinx. Hệ số alpha sẽ được chạy trong khoảng [0,1], trong trường hợp này, thực hiện tính toán như sau: lấy xác suất tiên nghiệm của ALIZE và xác suất hậu nghiệm của Sphinx nhân với hệ số alpha theo công thức trên. Xác suất tiên nghiệm của ALIZE chính là kết quả chạy chương trình, giá trị xác suất hậu nghiệm Sphinx được tính = xác suất của từng từ khóa/ tổng hợp của N người nói từ khóa đó.

1. Thử nghiệm dữ liệu lấy từ xác suất của ALIZE và Sphinx:

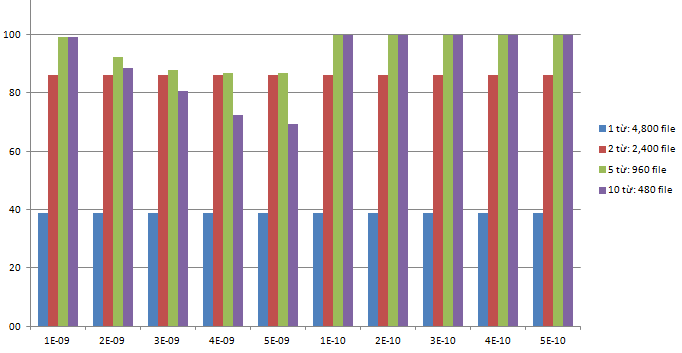
*Chạy theo xác suất: pSpeakerMax = PAlize+ alpha\*PSphinx*

Bảng 3.2: Kết quả nhận dạng người nói thử nghiệm với Test 1: sử dụng xác suất tiên nghiệm lấy từ mô hình HMM và GMM của người nói

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tập Test** | **Tỷ lệ nhận dạng đúng % (alpha = value)** | | | | | | | | | |
| **1E-9** | **2E-9** | **3E-9** | **4E-9** | **5E-9** | **1E-10** | **2E-10** | **3E-10** | **4E-10** | **5E-10** |
| **1 từ khóa** | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 | 38,7 |
| **2 từ khóa** | 86,3 | 86,2 | 86,1 | 86 | 86 | 86,3 | 86,3 | 86,3 | 86,3 | 86,3 |
| **5 từ khóa** | 99,3 | 92,4 | 87,9 | 87 | 86,8 | 100 | 100 | 100 | 100 | 100 |
| **10 từ khóa** | 92,2 | 88,5 | 80,6 | 72,3 | 69,4 | 100 | 100 | 100 | 100 | 100 |

% Chính

xác



alpha

Hình 3.5. Biểu đồ thể hiện kết quả nhận dạng với Test 1

***Nhận xét:*** Nhìn vào bảng tổng hợp và biểu đồ cho thấy kết quả nhận dạng người nói với sự thay đổi giá trị alpha với bộ từ khóa ngắn (gồm 1 từ khóa và 2 từ khóa) không thay đổi nhiều. Với bộ từ khóa dài( gồm: 5 và 10 từ khóa) tỷ lệ nhận dạng đạt cao hơn và đạt đến độ chính xác tuyệt đối 100%, tuy nhiên khi thay đổi giá trị alpha=5E-9 thì tỷ lệ đạt đã giảm hẳn, chỉ còn 86,8%. Tỷ lệ nhận dạng người nói tăng nên gấp đôi khi thay đổi từ khóa từ 1 lên 2 từ khóa và với người nói từ 5 từ khóa trở nên, khi giá trị alpha nhỏ, tỷ lệ kết quả đạt độ chính xác tuyệt đối 100%.

1. Thử nghiệm dữ liệu lấy từ xác suất tiên nghiệm của ALIZE và xác suất hậu nghiệm Sphinx

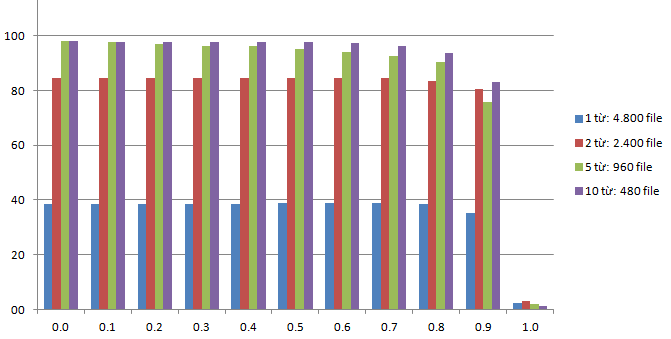
*Công thức: *

Bảng 3.3: Kết quả nhận dạng người nói thử nghiệm với Test 2: Xác suất tiên nghiệm lấy từ mô hình GMM và xác suất hậu nghiệm lấy từ mô hình HMM của từng người nói

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tập Test** | **Tỷ lệ nhận dạng đúng % (với giá trị alpha=)** | | | | | | | | | | |
| **0.0** | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** | **1** |
| **1 từ khóa:** | 38,7 | 38,7 | 38,6 | 38,6 | 38,7 | 38,8 | 39,0 | 38,8 | 38,4 | 35,1 | 2,5 |
| **2 từ khóa:** | 84,6 | 84,5 | 84,6 | 84,5 | 84,5 | 84,5 | 84,5 | 84,6 | 83,6 | 80,5 | 3,1 |
| **5 từ khóa:** | 97,9 | 97,5 | 97,1 | 96,4 | 96,1 | 95,0 | 94,2 | 92,7 | 90,2 | 75,8 | 2,2 |
| **10 từ khóa**: | 97,9 | 97,7 | 97,7 | 97,7 | 97,7 | 97,3 | 96,3 | 97,9 | 93,7 | 83,1 | 1,5 |

% Chính

xác



alpha

Hình 3.6. Biểu đồ thể hiện kết quả nhận dạng với Test 2

***Nhận xét:*** Nhìn vào bảng tổng hợp và biểu đồ cho thấy kết quả nhận dạng người nói không thay đổi nhiều khi giá trị alpha ở khoảng 0.0 đến 0.8. Từ giá trị alpha =0.9 và alpha=1 tỷ lệ nhận dạng giảm đi rõ rệt. Với giá trị alpha=0.9: nhận dạng 1 từ khóa giảm 3-4%, nhận dạng 2 từ khóa giảm khoảng 4%, nhận dạng 5 từ khoán giảm từ 15-22%, nhận dạng 10 từ khóa giảm trên 10%. Với giá trị alpha=1, tỷ lệ nhận dạng giảm xuống rất nhiều đối với 1,2,5,10 từ khóa chỉ còn khoảng 1,5 đến 3,1% . Như vậy tỷ lệ nhận dạng giảm khi giá trị alpha tăng lên

* + 1. Phân tích và đánh giá kết quả

1. Thời gian chạy chương trình thử nghiệm:

Bảng 3.4: Kết quả tổng hợp các thử nghiệm theo thời gian

|  |  |  |
| --- | --- | --- |
| **STT** | **Tập Test** | **Thời gian chạy (tính theo phút)** |
| **Test 1, Test 2** |
| 1 | 1 từ khóa: 4800 file | 480 |
| 2 | 2 từ khóa: 2400 file | 265 |
| 3 | 5 từ khóa: 960 file | 204 |
| 4 | 10 từ khóa: 480 file | 150 |

**Nhận xét:** Nhìn qua bảng thời gian chạy thử nghiệm chương trình cho ta thấy, kết quả chạy chương trình của 2 test là không khác nhau do trong lập trình viết đưa ra file dữ liệu tổng hợp dùng chung cho cả 2 test. Kết quả của 2 test chỉ thay công thức chạy (3.1) hay (3.2). Bảng tổng hợp theo thời gian chạy trên cũng cho ta thấy rằng, đối với cùng 10 file dài thử nghiệm (gồm 5 file của 48 người nói số từ “không” đến “chín” và 5 file của 48 người nói số từ “chín” đến “không”) thì số file dữ liệu càng chia nhỏ ra thì chương trình chạy càng mất nhiều thời gian hơn.

1. Kết quả chương trình chạy tối ưu:

Lấy giá trị nhận dạng cho ra kết quả nhận dạng tốt nhất của các chương trình nhận dạng kết hợp của ALIZE và Sphinx3 được bảng 3.4:

Bảng 3.5: Bảng tổng hợp kết quả nhận dạng các chương trình đã thử nghiệm

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tập Test** | **Tỷ lệ nhận dạng đúng (%)** | |
| **Test 1** | **Test 2** |
| 1 | 1 từ khóa | 38,7 | 39 |
| 2 | 2 từ khóa | 86,3 | 84,6 |
| 3 | 5 từ khóa | 100 | 97,9 |
| 4 | 10 từ khóa | 100 | 97,9 |

Hình 3.7. Biểu đồ thể hiện kết quả nhận dạng với các thử nghiệm

***Qua số liệu thống kê được biểu diễn bằng bảng và biểu đồ, ta có nhận xét chung như sau:***

Nhận dạng người nói với từ khóa dài có tỷ lệ nhận dạng người nói tốt hơn hẳn, đạt độ chính xác 100% với test 1 của 5 từ khóa và 10 từ khóa, với test 2, tỷ lệ nhận dạng thấp hơn 1 chút, chỉ đạt khoảng 97%. Với độ dài từ khóa giảm còn 2 từ khóa, tỷ lệ nhận dạng đạt 86,3% với test 1 và 84,6% với test 2. Khi độ dài từ khóa giảm chỉ còn 1 từ khóa, tỷ lệ nhận dạng giảm chỉ còn <40%. Trong thực tế thì việc nhận dạng người nói 1 từ khóa với thời gian của mỗi từ khóa chỉ là (0,03-0,05 s) thì tương đối khó có thể cải thiện để độ chính xác nhận dạng tăng lên cao. Khi chạy thử nghiệm với 2 từ khóa mỗi file .wav có thời gian là (1-3s) thì tỷ lệ nhận dạng cũng đã tăng lên cao >2 lần so với 1 từ khóa. Với 5, 10 từ khóa thời gian cho mỗi file .wav chạy thử nghiệm là (4-13s) đều có thể đạt được đến tỷ lệ nhận dạng 100%.

Kết quả của 2 test có sự chênh lệch về tỷ lệ nhận dạng. Trong khi test 2 với 1 từ khóa có kết quả nhận dạng tốt hơn 0,3% nhưng đối với bộ từ khóa dài từ 2-10 từ khóa thì tỷ lệ nhận dạng của test 1 lại tốt hơn khi giá trị alpha của test 1 nhỏ

Kết quả ở bảng 3.3 cũng cho thấy rằng thời gian nhận dạng giảm khi người nói thể hiện một câu dài hơn (5 từ khóa và 10 từ khóa so với câu ngắn gồm 1 và 2 từ khóa).

Chương 4. KẾT LUẬN VÀ KIẾN NGHỊ

* 1. Những vấn đề đã giải quyết trong luận văn

Trong luận văn, tác giả đã sử dụng 2 phần mềm mã nguồn mở để áp dụng cho bài toán nhận dạng là công cụ Alize và Sphinx. 2 phần mềm này đã được thử nghiệm trên một số nước: Pháp, Anh, Thụy Điển, Đức, Thái …. Nhưng với đặc thù tiếng nói tiếng Việt, tác giả cũng gặp phải 1 số vấn đề khó khăn: gói công cụ Alize không dễ sử dụng do nó có rất ít tài liệu hỗ trợ. Hơn nữa, có những phần tài liệu kèm theo không phản ánh được việc cài đặt hiện tại.

Tuy chưa có đóng góp quan trọng nào vào trong hai công cụ này nhưng tôi đã mất rất nhiều thời gian đọc và tìm hiểu để có thể chạy hoàn chỉnh ứng dụng trên môi trường Linux.

Với đề tài: “ ***Nhận dạng người nói phụ thuộc vào từ khóa tiếng Việt***” tác giả đã giải quyết một số vấn đề sau:

* Tìm hiểu tổng quan bài toán nhận dạng tiếng nói nói chung và nhận dạng người nói phụ thuộc từ khóa nói riêng.
* Nghiên cứu các công cụ thích hợp để lựa chọn, áp dụng vào bài toán cụ thể và giới thiệu các chức năng, cách cài đặt của gói công cụ Alize và Sphinx
* Xây dựng được kho dữ liệu là các từ khóa tiếng Việt các số từ 0 - > 9 với 48 người nói ở các độ tuổi khác nhau ở phương ngữ Bắc của Việt Nam
* Phát triển và kết hợp 2 chương trình nhận dạng với bộ dữ liệu thử nghiệm người nói đạt độ chính xác cao, tuy thu âm với số lượng người chưa nhiều, chủ yếu là ở các tỉnh, huyện lân cận nhau Hưng Yên, Hải Dương, Bắc Ninh, Hà Nam với giọng nói chưa khác biệt nhau nhiều lắm nhưng với bộ từ khóa đủ lớn tỷ lệ nhận dạng đạt độ chính xác tuyệt đối 100%. Đây cũng là một bước nghiên cứu đóng góp vào phát triển thực tiễn cho lĩnh vực từ động nhận dạng người nói với từ khóa tiếng Việt, ứng dụng rất lớn trong thực tế để đăng nhập vào các tài khoản cá nhân thay password hay mã pin … như hiện nay.
  1. Hướng phát triển của đề tài

Với những kết quả đã đạt được, hướng phát triển của đề tài là:

- Tiếp tục xây dựng kho dữ liệu người nói đa dạng hơn: bao gồm số lượng người nói lớn và các phương ngữ khác nhau để đưa vào thử nghiệm trên ứng dụng.

- Phát triển ứng dụng người nói với bộ từ khóa ngắn làm sao để đem lại kết quả nhận dạng cao hơn.

- Kết hợp với ứng dụng người nói không phụ thuộc từ khóa và văn bản để thêm tính đa dạng của ứng dụng trong thực tế.

TÀI LIỆU THAM KHẢO

[1] Bonastre J.-F., Scheffer N., Fredouille C., Matrouf D. (2004), “Nist’04 Speaker Recognition Evaluation Campaign: New Lia Speaker Detection Plateform Based On Alize Toolkit”, *2004 NIST SRE 04 Workshop: speaker detection evaluation campaign*, Toledo, Spain.

[2] Bonastre J-F, Scheffer N., Matrouf D., Fredouille C., Larcher A., Preti A., Pouchoulin G., Evans N., Fauve B. and Mason J. (2008), “ALIZE/SpkDet: a state-of-the-art open source software for speaker recognition”, *Proc. Odyssey: the Speaker and Language Recognition Workshop*.

[3] Dương Tử Cường (2003), *Xử lý tín hiệu số*, Nxb Quân đội nhân dân, Hà

Nội

[4] Đinh Lê Thành, Nguyễn Quốc Linh, Trịnh Văn Loan, “Định danh người nói Tiếng việt sử dụng mô hình hỗn hợp Gaussian “, *Hội nghị khoa học lần thứ 20 – ĐH Bách khoa Hà Nội,* tr. 40-45.

[4] Jean-Franc¸ois Bonastre, Fr´ed´eric Wils (2005), “ALIZE, A FREE TOOLKIT FOR SPEAKER RECOGNITION”, [*IEEE International Conference*](http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=9711), pp. I 737 - I 740

[5] Ngô Minh Dũng, Đặng Văn Chuyết (2007), “ Xây dựng và khảo sát độ dài từ khóa trong nhận dạng người nói phụ thuộc vào từ khóa tiếng Việt theo mô hình Markov ẩn”, *Tạp chí BCVT & CNTT* ), tr. 93-99.

[6] Ngô Minh Dũng, Đặng Văn Chuyết (2004), “Khảo sát tính ổn định của một số đặc trưng ngữ âm trong nhận dạng người nói”, *Tạp chí BCVT & CNTT,* (kỳ 3 10/2004), tr. 12-18.

[7] Viện công nghệ thông tin Việt Nam (2004), *Tổng hợp và nhận dạng tiếng* *Việt*, Đề tài nghiên cứu cấp nhà nước KC01- 03D, Hà Nội

Website

8. <http://arxiv.org/ftp/arxiv/papers/1205/1205.3316.pdf>

9. <http://cmusphinx.sourceforge.net/wiki/tutorialpocketsphinx>

10. <http://cmusphinx.sourceforge.net/>

11. <http://en.wikipedia.org/wiki/Speech_recognition>

12. <http://home.iitb.ac.in/~jigar23/documents/Sphinx_Guideline_v1.pdf>

13. <http://mistral.univ-avignon.fr/index_en.html>

14. <http://ronaldramdhan.wordpress.com/2010/03/11/sphinxtrain/>

15. <http://www.irisa.fr/metiss/guig/spro/spro-4.0.1/spro_1.html#SEC1>

16. <http://www.speech.cs.cmu.edu/sphinx/tutorial.html>

17. <http://www.fon.hum.uva.nl/praat/>

PHỤ LỤC

1. Mô hình hỗn hợp Gauss

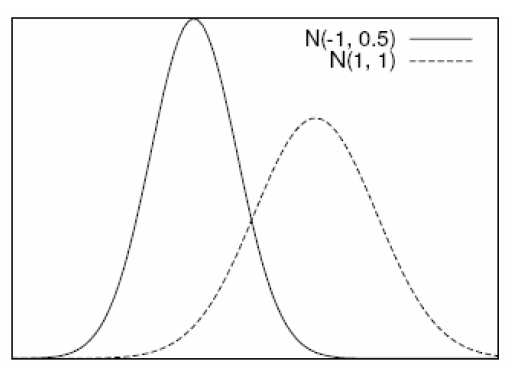
A.1. Đặc tả mô hình

Mô hình hợp Gauss (Gaussian Mixture Model - GMM) là một dạng mô hình thống

kê được xây dựng từ việc huấn luyện các tham số thông qua dữ liệu học. Mô hình

GMM còn có một số tên gọi khác như Weighted Normal Distribution Sums hay

Radial Basis Function Approximations…



Hình A.1. Hàm mật độ Gauss

Về cơ bản, mô hình GMM xấp xỉ một hàm mật độ xác suất bằng hợp các hàm mật

độ Gauss. Hình A.1 minh họa hai hàm mật độ Gauss với các tham số khác nhau.

Một cách hình thức, hàm mật độ xác suất của phân phối Gauss fN(x, μ, σ2) được cho bởi công thức:

 ( 1.1)

trong đó, μ là giá trị trung bình, σ là độ lệch chuẩn. Trong trường hợp x là vector

gồm D thành phần, hàm mật độ xác suất của phân phối Gauss fN(x, μ, Σ) được cho

bởi công thức:



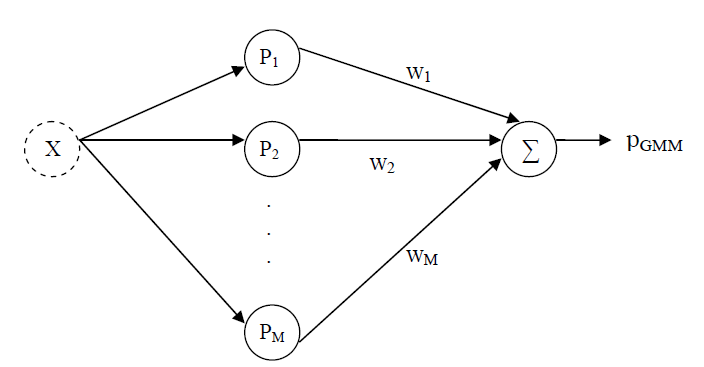
khi đó, μ là vector trung bình, Σ là ma trận hiệp phương sai. Nếu chọn μ=0 và σ=1,

công thức (1.1) sẽ trở thành hàm mật độ chuẩn Gauss:



Từ “Gauss” được đặt theo tên của nhà toán học người Đức Carl Friedrich Gauss.

Ông đã định nghĩa hàm mật độ Gauss và áp dụng trong phân tích dữ liệu thiên văn.



Hình A.2. Mô hình GMM

Cho trước M phân phối Gauss p1, p2, …, pM, hàm mật độ xác suất của mô hình

GMM được minh họa trong hình 3.2 chính là tổng trọng của M phân phối Gauss

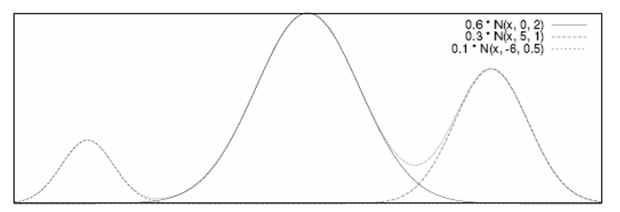
theo công thức:



trong đó, wi là trọng số của phân phối Gauss thứ i, thỏa ràng buộc 0≤ wi ≤1 và

 Các trọng số này thể hiện mức độ ảnh hưởng của mỗi phân phối Gaussđối với mô hình GMM. Như vậy, phân phối Gauss có phương sai và trọng số lớn bao nhiêu thì có mức độ ảnh hưởng lớn bấy nhiêu đối với kết xuất của mô hình.

Hình 3.3 cho thấy mức độ ảnh hưởng của từng phân phối Gauss lên GMM.



Hình A.3. Hàm mật độ của GMM có 3 phân phối Gauss

số λ = { wi, μi, Σi }, i ∈ [1, M]. Trong hướng tiếp cận GMM giải quyết bài toàn định danh người nói, mỗi người nói sẽ được mô hình hóa bằng một mô hình GMM mà bộ tham số λ của nó sẽ được xác định thông qua việc huấn luyện trên tập mẫu học của người nói tương ứng.

Tùy thuộc vào cách tổ chức của ma trận hiệp phương sai (Covariance Matrix), GMM có thể có một số biến thể khác nhau:

- Nodal covariance matrices GMM: mỗi phân phối Gauss trong GMM có một

ma trận hiệp phương sai riêng.

- Grand Covariance Matrix GMM: mọi phân phối Gauss trong một GMM dùng

chung một ma trận hiệp phương sai.

- Global Covariance Matrix GMM: mọi phân phối Gauss trong tất cả các GMM

dùng chung một ma trận hiệp phương sai.

Ngoài ra, xét về dạng thức, ma trận hiệp phương sai gồm hai loại: Full (dạng đầy đủ) và Diagonal (dạng ma trận đường chéo). Thông thường, dạng Nodal-Diagonal

Covariance Matrices GMM được sử dụng phổ biến nhất.

A.2. Bài toán ước lượng mật độ

Trong bộ phân loại dựa trên mô hình thống kê, việc ước lượng các tham số của mô hình được thực hiện thông qua huấn luyện trên một số lượng lớn các dữ liệu học. Mục tiêu của bước huấn luyện là nhằm tổng quát hóa, mô hình hóa những đặc điểm chung nhất của tập dữ liệu học. Đối với mô hình GMM, một trong những kỹ thuật xác định bộ tham số λ của nó được áp dụng khá phổ biến là thuật toán Expectation-Maximization (EM). Bản thân EM là một thuật toán tổng quát, đem lại các kết quả khác nhau đối với các mô hình khác nhau. Ngoài ra, có hai tiêu chí ước lượng khác nhau trong EM:

- Maximum Likelihood (ML): ước lượng tham số theo hướng cực đại hóa độ tương tự p(X | λ).

- Maximum A Posteriori Probability (MAP): ước lượng tham số theo hướng cực đại hóa xác suất quyết định p(λ | X).

Cho trước vector đặc trưng X trích được từ dữ liệu âm thanh, ta có thể dễ dàng tính được độ tương tự p(X | λ). Tuy nhiên, trong định danh người nói, vai trò quyết định lại nằm ở xác suất p(λ | X). Sử dụng công thức Bayes, ta có tương quan giữa p(X|λ) và p(λ|X):



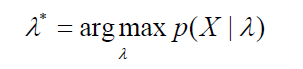
trong đó, p(X) là xác suất xuất hiện của vector đặc trưng X, p(λ) là tần suất xuất

hiện của người nói được mô hình hóa bởi GMM tương ứng. Trong luật quyết định

Bayes, p(X) độc lập; như vậy nếu giả định p(λ) là đồng nhất cho mọi người nói, ta

có thể quy vai trò quyết định từ p(λ | X) về p(X | λ) và áp dụng EM ước lượng λ

theo hướng maximum likelihood:



Như vậy, với tập dữ liệu huấn luyện X gồm T mẫu X  các trọng số, trung bình và phương sai của GMM ở mỗi bước lặp sẽ là:

Trọng số: 

Trung bình: 

Phương sai: 

trong đó,  là các thành phần tương ứng trong các vector  . Xác suất cho bởi công thức:



Trong quá trình xây dựng GMM, có hai vấn đề phát sinh là: số phân phối Gauss M

của mô hình, và bộ tham số khởi đầu λ0 trước khi tiến hành thuật toán EM. Hiện tại,

vẫn chưa có giải pháp tối ưu trên lý thuyết cho việc chọn M và λ0. Thông thường, M sẽ được chọn qua thực nghiệm, còn λ0 sẽ được khởi tạo bằng thuật toán K-means

nhằm đem lại khả năng cao hơn cho việc đạt tối ưu toàn cục, đồng thời đẩy nhanh

tốc độ hội tụ trong huấn luyện.

B. Ba bài toán cơ bản của HMM

B.1. Bài toán thứ nhất: Đánh giá xác suất

Cho dãy quan sát O = (o1o2…oT) và mô hình Markov ẩn λ = (A, B, π). Hãy tính P(O|λ) là xác suất của dãy quan sát theo mô hình.

Đây là bài toán đánh giá, tức là cho trước mô hình và dãy quan sát, ta phải tính toán xác suất mô hình tạo ra dãy quan sát đó như thế nào? Có thể xem đây là một kiểu đánh giá xem mô hình được cho có tốt với dãy quan sát đó hay không. Trong trường hợp ta phải chọn một trong số nhiều mô hình thì giải pháp cho bài toán 1 sẽ cho ta sự lựa chọn được mô hình thích hợp nhất đối với dãy quan sát.

Để giải quyết bài toán 1 ta sử dụng thủ tục tiến và thủ tục lùi được mô tả như sau:

Thủ tục tiến (Forward Procedure)

Xét biến tiến αt(i) được định nghĩa như sau:

αt(i) = P(o1o2…ot , qt=i|λ) (2.10)

Đó là xác suất của dãy quan sát bộ phận o1o2…ot và trạng thái i đạt được tại thời điểm t.

Ta có thể dùng phương pháp quy nạp để tính αt(i) như sau:

Bước 1. Khởi tạo:

α1(i) = πibi(o1) 1 ≤ i ≤ N (2.11)

Bước 2. Quy nạp:

 1 ≤ t ≤ T-1, 1 ≤ j ≤ N (2.12)

Bước 3. Kết thúc:

 (2.13)

Thủ tục lùi (Backward Procedure)

Tương tự ta có thể xét biến lùi βt(i) được định nghĩa như sau:

βt(i) = P(ot+1, ot+2 , oT , qt=i|λ) (2.14)

βt(i) là xác suất của dãy quan sát bộ phận từ thời điểm t+1 đến thời điểm cuối, căn cứ vào trạng thái i tại thời điểm t và mô hình λ. Có thể tính βt(i) bằng phương pháp quy nạp như sau:

**Bước 1. Khởi đầu:**

**βT(i) = 1 1 ≤ i ≤ N (2.15)**

**Bước 2. Quy nạp:**

**** (2.16)

**t = T – 1, T – 2, …, 1 1 ≤ i ≤ N**

**Bước 3. Kết thúc:**

**** (2.17)

Ở các phần sau ta sẽ thấy cách áp dụng các thủ tục tiến và lùi trong việc giải quyết vấn đề 3 của mô hình Markov ẩn.

B.2. Thuật toán thứ hai: Tìm dãy trạng thái tối ưu

Cho dãy quan sát O=(o1o2…oT) và mô hình λ. Làm thế nào lựa chọn được 1 dãy trạng thái q=(q1q2…qT) là tối ưu theo một nghĩa nào đó.

Đây là bài toán trong đó ta cố gắng tháo gỡ phần “ẩn” của mô hình để tìm ra dãy trạng thái “đúng” ẩn bên dưới. Không giống như bài toán 1 với kết quả cho ra là chính xác, bài toán 2 có nhiều cách giải quyết, tức là có nhiều cách để tìm ra dãy trạng thái “tối ưu” ứng với dãy quan sát cho trước. Khó khăn nằm ở chỗ định nghĩa dãy trạng thái thế nào là tối ưu – đó là vì có nhiều tiêu chuẩn tối ưu. Một tiêu chuẩn tối ưu là làm cực đại P(q|O,λ) cũng là làm cực đại P(q,O|λ). Phương pháp quy hoạch động để tìm chuỗi trạng thái đơn tốt nhất chính là giải thuật Viterbi sau đây.

Thuật toán Viterbi

Để tìm dãy trạng thái đơn tốt nhất q = (q1q2…qT) đối với dãy quan sát O = (o1o2…oT), ta định nghĩa đại lượng sau:



*δ­t(i)* là xác suất lớn nhất dọc theo một đường đi đơn tại thời điểm t của t quan sát đầu tiên trong đó có quan sát cuối cùng là ở trạng thái i.

Bằng quy nạp ta có:



Ta sử dụng mảng *ψt(j)* để giữa lại các đối số làm cực đại biểu thức trên ứng với mỗi t và j.

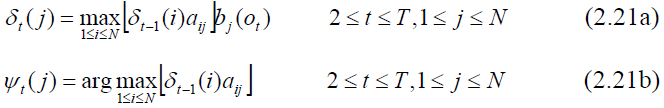
Bước 1. Khởi tạo:

*δ1(i) = πibi(o1­) 1 ≤ i ≤ N*

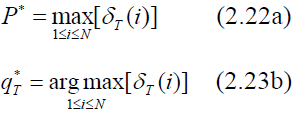
**(2.20**)

*ψ1(i) = 0*

Bước 2. Vòng lặp:



Bước 3. Kết thúc:



Bước 4. Quay ngược lại tìm đường đi (dãy trạng thái):



B.3. Thuật toán thứ ba: Ước lượng tham số của mô hình

Cần phải điều chỉnh các tham số của mô hình như thế nào để P(O|λ) đạt cực đại. Đây là bài toán tối ưu tham số mô hình để mô tả tốt nhất dãy quan sát cho trước. Dãy quan sát dùng để hiệu chỉnh các tham số của mô hình được gọi là dãy huấn luyện. Bài toán huấn luyện là bài toán cốt lõi nhất trong 3 bài toán vì nó cho phép ta tối ưu các tham số mô hình với dữ liệu huấn luyện, tức là tạo ra mô hình tốt nhất cho hiện tượng thực tế.

Không có cách nào giải quyết tập tham số mô hình để làm cực đại xác suất của dãy quan sát một cách chặt chẽ. Tuy nhiên, ta có thể hiệu chỉnh các tham số của mô hình λ = (A, B, π) để xác suất P(O|λ) đạt cực đại địa phương bằng cách dùng phương pháp ước lượng Baum-Welch, còn gọi là phương pháp cực đại kỳ vọng (expectation maximization – EM).

Để mô tả thủ tục ước lượng lại tham số mô hình, đầu tiên ta định nghĩa *ξt(i,j)* là xác suất có trạng thái i tại thời điểm t và trạng thái j tại thời điểm t+1 với mô hình λ và dãy quan sát O đã cho, tức là:

*ξt(i,j) = P(q­t = i, qt+1 = j|O,λ)* (2.25)

Từ định nghĩa các biến tiến và biến lùi trong bài toán 1, ta có thể viết lại *ξt(i,j)* dưới dạng như sau:

 (2.26)





Định nghĩa *γt(i)* là xác suất trạng thái i ở thời điểm t với dãy quan sát O và mô hình λ, vì vậy ta có mối liên hệ giữa *γt(i)* với *ξt(i,j)* như sau:

 (2.27)

Nếu lấy tổng *γt(i)* theo thời gian t, ta nhận được đại lượng chính là kỳ vọng số lần hệ thống ở trạng thái i, hay tương đương với kỳ vọng số chuyển đổi trạng thái được thực hiện từ trạng thái i (nếu bỏ qua t = T trong phép lấy tổng). Tương tự, lấy tổng *ξt(i,j)*theo t (từ t = 1 đến t = T-1) là kỳ vọng số lần chuyển đổi từ trạng thái i sang trạng thái j, tức là :

 kỳ vọng số chuyển trạng thái từ trạng thái i trong O (2.28)

kỳ vọng số chuyển đổi từ trạng thái i sang trạng thái j trong O (2.29)

Từ các công thức trên ta có thể đưa ra một phương pháp đánh giá lại tham số của một mô hình Markov ẩn như sau:

*Ước lượng tham số π :*

* =* kỳ vọng số lần ở trạng thái i tại thời điểm t=1

=  (2.30)

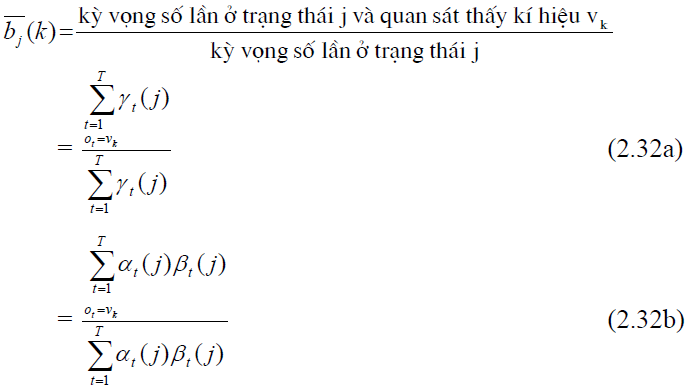


*Ước lượng tham số A :*

** (2.31a)

** (2.31b)

*Ước lượng tham số B :*

**

C. Cấu trúc các gói trong thư viện LIA-RAL

C.1. Cấu trúc file NormFeat.cfg

*\* NormFeat config File*

*mode norm*

*bigEndian false*

*loadFeatureFileFormat SPRO4*

*saveFeatureFileFormat SPRO4*

*loadFeatureFileExtension .prm*

*saveFeatureFileExtension .norm.prm*

*featureServerBufferSize ALL\_FEATURES*

*featureFilesPath ./prm/*

*sampleRate 100*

*saveFeatureFileSPro3DataKind FBCEPSTRA*

*labelSelectedFrames speech*

*segmentalMode false*

*writeAllFeatures true*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*featureServerMode FEATURE\_WRITABLE*

*featureServerMemAlloc 50000000*

C.2. Cấu trúc file NormFeat\_energy.cfg

*\* NormFeat config File*

*mode norm*

*bigEndian false*

*loadFeatureFileFormat SPRO4*

*saveFeatureFileFormat SPRO4*

*loadFeatureFileExtension .prm*

*saveFeatureFileExtension .enr.prm*

*featureServerBufferSize ALL\_FEATURES*

*featureFilesPath ./prm/*

*sampleRate 100*

*saveFeatureFileSPro3DataKind FBCEPSTRA*

*labelSelectedFrames all*

*addDefaultLabel true*

*defaultLabel all*

*segmentalMode false*

*writeAllFeatures true*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*vectSize 60*

*featureServerMode FEATURE\_WRITABLE*

*featureServerMemAlloc 50000000*

### C.3. Cấu trúc file EnergyDetector.cfg

*\*\*\* EnergyDetector Config File*

*\*\*\**

*loadFeatureFileExtension .enr.prm*

*minLLK -200*

*maxLLK 1000*

*bigEndian false*

*loadFeatureFileFormat SPRO4*

*saveFeatureFileFormat SPRO4*

*saveFeatureFileSPro3DataKind FBCEPSTRA*

*featureServerBufferSize ALL\_FEATURES*

*featureServerMemAlloc 50000000*

*featureFilesPath ./prm/*

*mixtureFilesPath ./*

*lstPath ./lst/*

*labelOutputFrames speech*

*labelSelectedFrames all*

*addDefaultLabel true*

*defaultLabel all*

*saveLabelFileExtension .lbl*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*segmentalMode file*

*nbTrainIt 8*

*varianceFlooring 0.0001*

*varianceCeiling 1.5*

*alpha 0.0*

*mixtureDistribCount 3*

*featureServerMask 19*

*vectSize 1*

*baggedFrameProbabilityInit 0.001*

*thresholdMode weight*

*Cấu trúc file TrainWorldInit.cfg*

*distribType GD*

*mixtureDistribCount 20*

*loadMixtureFileExtension .gmm*

*saveMixtureFileExtension .gmm*

*loadFeatureFileExtension .norm.prm*

*maxLLK 200*

*minLLK -200*

*bigEndian false*

*saveMixtureFileFormat RAW*

*loadMixtureFileFormat RAW*

*loadFeatureFileFormat SPRO4*

*featureServerBufferSize ALL\_FEATURES*

*featureServerMemAlloc 100000000*

*featureFilesPath ./prm/*

*mixtureFilesPath ./gmm/*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*lstPath ./lst/*

*labelSelectedFrames speech*

*baggedFrameProbability 0.2*

*baggedFrameProbabilityInit 0.5*

*initVarianceFlooring 0.5*

*initVarianceCeiling 1*

*finalVarianceFlooring 0.5*

*finalVarianceCeiling 5.0*

*normalizeModel true*

*nbTrainIt 10*

*use01 true*

*verbose true*

*featureServerMask 0-18,20-50*

*baggedMinimalLength 1*

*baggedMaximalLength 1*

C.4. Cấu trúc file TrainWorldInit.cfg

*distribType GD*

*mixtureDistribCount 20*

*loadMixtureFileExtension .gmm*

*saveMixtureFileExtension .gmm*

*loadFeatureFileExtension .norm.prm*

*maxLLK 200*

*minLLK -200*

*bigEndian false*

*saveMixtureFileFormat RAW*

*loadMixtureFileFormat RAW*

*loadFeatureFileFormat SPRO4*

*featureServerBufferSize ALL\_FEATURES*

*featureServerMemAlloc 100000000*

*featureFilesPath ./prm/*

*mixtureFilesPath ./gmm/*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*lstPath ./lst/*

*labelSelectedFrames speech*

*baggedFrameProbability 0.2*

*baggedFrameProbabilityInit 0.5*

*initVarianceFlooring 0.5*

*initVarianceCeiling 1*

*finalVarianceFlooring 0.5*

*finalVarianceCeiling 5.0*

*normalizeModel true*

*nbTrainIt 10*

*use01 true*

*verbose true*

*featureServerMask 0-18,20-50*

*baggedMinimalLength 1*

*baggedMaximalLength 1*

C.5. Cấu trúc file TrainWorldFinal.cfg

*distribType GD*

*mixtureDistribCount 20*

*loadMixtureFileExtension .gmm*

*saveMixtureFileExtension .gmm*

*loadFeatureFileExtension .norm.prm*

*maxLLK 200*

*minLLK -200*

*bigEndian false*

*saveMixtureFileFormat RAW*

*loadMixtureFileFormat RAW*

*loadFeatureFileFormat SPRO4*

*featureServerBufferSize ALL\_FEATURES*

*featureServerMemAlloc 100000000*

*featureFilesPath ./prm/*

*mixtureFilesPath ./gmm/*

*labelFilesPath ./lbl/*

*frameLength 0.01*

*lstPath ./lst/*

*labelSelectedFrames speech*

*baggedFrameProbability 1*

*initVarianceFlooring 0.5*

*initVarianceCeiling 5.0*

*finalVarianceFlooring 0.5*

*finalVarianceCeiling 5.0*

*featureServerMask 0-18,20-50*

*normalizeModel true*

*use01 true*

*nbTrainIt 2*

*baggedMinimalLength 1*

*baggedMaximalLength 1*

C.6. Cấu trúc gói TrainTarget

*\*\*\* TrainTarget Configuration File*

*\*\*\**

*maxLLK 200*

*minLLK -200*

*bigEndian false*

*saveMixtureFileFormat RAW*

*loadMixtureFileFormat RAW*

*loadFeatureFileFormat SPRO4*

*featureServerBufferSize ALL\_FEATURES*

*loadMixtureFileExtension .gmm*

*saveMixtureFileExtension .gmm*

*loadFeatureFileExtension .norm.prm*

*featureFilesPath ./prm/*

*labelFilesPath ./lbl/*

*mixtureFilesPath ./gmm/*

*nbTrainIt 1*

*labelSelectedFrames speech*

*normalizeModel true*

*normalizeModelMeanOnly true*

*normalizeModelNbIt 5*

*MAPAlgo MAPOccDep*

*meanAdapt true*

*MAPRegFactorMean 14*

*frameLength 0.01*

*featureServerMemAlloc 100000000*

*featureServerMask 0-18,20-50*

C.7. Cấu trúc gói ComputeTest

*\*\*\* ComputeTest Config File*

*\*\*\**

*distribType GD*

*loadMixtureFileExtension*

*saveMixtureFileExtension .gmm*

*loadFeatureFileExtension .prm*

*mixtureDistribCount 1024*

*maxLLK 200*

*minLLK -200*

*bigEndian false*

*saveMixtureFileFormat RAW*

*loadMixtureFileFormat RAW*

*loadFeatureFileFormat SPRO3*

*featureServerBufferSize ALL\_FEATURES*

*featureFilesPath ./*

*mixtureFilesPath ./*

*labelSelectedFrames male*

*labelFilesPath ./*

*frameLength 0.01*

*segmentalMode segmentLLR*

*topDistribsCount 10*

*computeLLKWithTopDistribs COMPLETE*

*ndxFileName ndx*

*worldModelName wld*

*outputFile test1.res*

*gender M*

*debug true*

*verbose true*

*featureServerMask 0-15,17-32*

*vectSize* 32

D. Code chương trình nhận dạng người nói tiếng Việt

D.1. Code chương trình tổng hợp xác suất của ALIZE và Sphinx

#!/usr/bin/python

import os, sys, math

#print "Hello World"

# Chuong trinh nhan dang nguoi noi su dung Sphinx3 va Alize

# + Dau vao:

# + File List : danh sach cac file am thanh can nhan dang (danh sach cac file Wav)

# + He so Alpha

# + Dau ra:

# + File ket qua nhan dang

# + Ty le nhan dang

# + Giai thuat:

# + Cau hinh cho Alize

# + Cau hinh cho Sphinx: duong dan den HMM cua 48 nguoi noi, file dict

def docFileText(fileText):

lines = []

try:

f = open(fileText, "r")

lines = f.readlines()

for i in range(len(lines)):

lines[i] = lines[i].strip()

f.close()

except:

print "File ", fileText, " khong co !!!"

finally:

return lines

def taoFileTranscription(fileWav):

# Dau vao: AnhDT\_2Tu\_01\_1.wav

# Dau ra: file mySphinx.transcriptions.ar

# Format : xau ky tu (tenfile)

# Vi du: NAM (abc)

number = fileWav.split("\_")[1]

print "Number : ", number

# Tao xau ky tu

myDict = {"09":"KHONG MOT HAI BA BON NAM SAU BAY TAM CHIN", "90":"CHIN TAM BAY SAU NAM BON BA HAI MOT"}

xaukytu = myDict[number]

xaukytu = xaukytu + " (abc)"

print "Xau ky tu: ", xaukytu

# Tao file mySphinx.transcription

try:

f = open("mySphinx.transcription", "w")

f.write(xaukytu)

except:

print "Khong ghi duoc vao file mySphinx.transcription"

f.close()

def taoFileChiSoNguoiNoi(idSpeaker):

print "Chi so nguoi noi", idSpeaker

myName = {"0": "AnhDT", "1": "BayNVN", "2": "CaoNV", "3": "ChienVC", "4": "ChiLT", "5": "DiepDT", "6": "DungDT", "7": "DuongPTN", "8": "DuongDD", "9": "HaiPN", "10":"HangDT", "11": "HanhTT", "12": "HiepTV", "13": "HoaVTT", "14": "HueNT", "15": "HungPM", "16": "HuongVK" ,"17": "HuyenNT" ,"18": "HuyLT", "19": "HuyLV", "20": "KhanhTV", "21": "KhoaDD", "22": "KhoeDD", "23": "KhuongNT", "24": "KienPT", "25": "KiPH", "26": "LanAnhDT", "27":"LienPTN", "28": "MinhNV","29": "NangNTH","30": "NhiNX","31": "QuynhDT","32": "QuynhNT","33": "SonVV","34":"SuotNTTK","35": "TamTT","36": "ThucBD","37": "ThuongND", "38": "ThanhLT","39": "ThuyNTTK","40": "ToPA","41": "TrangBTH","42": "TrangPT","43": "TuPT","44": "VanTT","45": "VuiLT","46":"YenNT", "47":"YenNTN"}

speaker = myName[str(idSpeaker)]

return speaker

# Tao fileNdx trong thu muc ./TestAlize/

def taoFileNdx(fileWav):

lst = fileWav.split(".")[0]

fNdx = open("./TestAlize/testSpeaker" + lst +".ndx","w")

str = lst + " AnhDT\_gmm BayNVN\_gmm CaoNV\_gmm ChienVC\_gmm ChiLT\_gmm DiepDT\_gmm " +\

"DungDT\_gmm DungPTN\_gmm DuongDD\_gmm HaiPN\_gmm HangDT\_gmm HanhTT\_gmm " +\

"HiepTV\_gmm HoaVTT\_gmm HueNT\_gmm HungPM\_gmm HuongVK\_gmm HuyenNT\_gmm " +\

"HuyLT\_gmm HuyLV\_gmm KhanhTV\_gmm KhoaDD\_gmm KhoeDD\_gmm KhuongNT\_gmm " +\

"KienPT\_gmm KiPH\_gmm LanAnhDT\_gmm LienPTN\_gmm MinhNV\_gmm NangNTH\_gmm " +\

"NhiNX\_gmm QuynhDT\_gmm QuynhNT\_gmm SonVV\_gmm SuotNTTK\_gmm TamTT\_gmm " +\

"ThucBD\_gmm ThuongND\_gmm ThanhLT\_gmm ThuyNTTK\_gmm ToPA\_gmm " +\

"TrangBTH\_gmm TrangPT\_gmm TuPT\_gmm VanTT\_gmm VuiLT\_gmm YenNT\_gmm YenNTN\_gmm"

fNdx.write(str)

fNdx.close()

# ghi ket qua vao file TONGHOP1TUAS

# Dinh dang file:

# + ten\_file\_wav cac\_gia\_tri\_Alize cac\_gia\_tri\_Sphinx

# + cac\_gia\_tri\_Alize: 48 gia tri xac suat ung voi 48 nguoi noi tinh voi Alize

# + cac\_gia\_tri\_Sphinx: 48 gia tri xac suat ung voi 48 nguoi noi tinh voi Sphinx

def ghi\_ket\_qua\_1\_file(fileWav):

# Dau vao: file Wav AnhDT\_5\_01.wav

# ==> mySphinx.transcription : abc

taoFileTranscription(fileWav)

"---------------------------- CHAY ALIZE -----------------------------------------"

# Tao ra file ket qua nhan dang boi Alize : AnhDT\_5\_01.res

lst = fileWav.split(".")[0]

fileText = "./TestAlize/" + lst + ".res"

taoFileNdx(fileWav)

tinhXacSuatAlize(fileWav)

"----------------------------KET THUC ALIZE---------------------------------------"

#for i in range(48):

fAS = open("TONGHOP10TUAS","a+")

fAS.write(fileWav)

fAS.write(" ")

# Ghi cac xac suat tinh bang Alize

for i in range(48):

print "################# Alize, Dang tinh xac suat voi nguoi noi: ", i

pAlize = docXacSuatAlize(fileText,i)

strAlize = "%.2f" % pAlize

print strAlize

fAS.write(strAlize)

fAS.write(" ")

# Ghi cac xac suat tinh bang Sphinx

for i in range(48):

print "################# Sphinx, Dang tinh xac suat voi nguoi noi: ", i

pSphinx = tinhXacSuatSphinx(fileWav, i)

strSphinx = "%.2f" % pSphinx

print strSphinx

fAS.write(strSphinx)

fAS.write(" ")

fAS.write("\n")

fAS.close()

def nhandang1file(fileWav):

# Dau vao: file Wav AnhDT\_5\_01.wav

# Dau ra: nguoi noi duoc nhan dang

# Giai thuat: Tinh P(Ai, W), tim Max voi i tu 1==>48 nguoi noi

# logP(Ai, W) = logP(Ai) + alpha \* logP(W|Ai)

# + logP(Ai): xac dinh nho Alize

# + logP(W|Ai): xac dinh nho sphinx3\_align

# ==> mySphinx.transcription : abc

taoFileTranscription(fileWav)

# Thu nghiem voi tung nguoi noi

pSpeakerMax = (-1) \* sys.float\_info.max

idSpeaker = -1

print "pSpeakerMax: ", pSpeakerMax

"---------------------------- CHAY ALIZE -----------------------------------------"

# Tao ra file ket qua nhan dang boi Alize : AnhDT\_5\_01.res

lst = fileWav.split(".")[0]

fileText = "./TestAlize/" + lst + ".res"

taoFileNdx(fileWav)

tinhXacSuatAlize(fileWav)

"----------------------------KET THUC ALIZE---------------------------------------"

#for i in range(48):

fAS = open("TONGHOP1TUAS","a+")

fAS.write(fileWav)

fAS.write(" ")

for i in range(48):

print "################# Dang thu nghiem voi nguoi noi: ", i

pAlize = docXacSuatAlize(fileText,i)

strAlize = "%.2f" % pAlize

print strAlize

#break

# Ghi ca alize va sphinx ra file moi

speaker1 = taoFileChiSoNguoiNoi(i)

fAS.write(speaker1)

#fAS.write(" Alize: ")

fAS.write(strAlize)

fAS.write(" ")

fAS.close()

def docXacSuatTuFileWDSEG(fileWdseg):

pWdseg = 0.0

lines = docFileText(fileWdseg)

print lines

if len(lines) == 0:

return 0.0

else:

# Total score: -564326 ==> pWdseg = -564326

pWdseg = lines[-1].split(":")[1]

print "pWdseg : ", pWdseg

pass

return float(pWdseg)

def docXacSuatAlize(fileText,i):

pAlize1 = 0.0

lines = docFileText(fileText)

if len(lines) == 49:

return 0.0

else:

# Total score: -564326 ==> pWdseg = -564326

pAlize1 = lines[i].split(" ")[4]

print "pAlize : ", pAlize1

pass

return float(pAlize1)

####################### XU LY BANG SPHINX: ==> P(W|Ai) ###############################

# Tinh xac suat P(W|Ai)

# Dau vau:

# + Ten File Wav

# + So thu tu nguoi noi

# Dau ra: gia tri xac suat tim duoc

def tinhXacSuatSphinx(fileWav, stt):

pSphinx = 0.5

# Tinh file MFCC

fileMFCC = "abc.mfc"

try:

cmd = "sphinx\_fe -i /home/diep/input/" + fileWav + " -o " + fileMFCC + " 2>logSphinxFe"

os.system(cmd)

except:

print "Loi chay lenh sphinx\_fe"

return pSphinx

# Xoa file abc.wdseg

try:

os.system("rm -f abc.wdseg")

except:

print "Khong xoa duoc file abc.wdseg"

# Chay forced\_alignment

try:

cmd = "/root/Install/NewInstall/sphinx3/src/programs/sphinx3\_align -insent mySphinx.transcription -outsent abc.out -hmm " + \

hmmSphinxLst[stt] + " -dict mySphinx.dic -ctl mySphinx.ctl -cepdir . -cepext .mfc -wdsegdir . 2>logSphinx3\_Align 1>logSphinx3Align"

#print cmd

os.system(cmd)

except:

print "Loi chay lenh sphinx3\_align"

# Doc xac suat tu file abc.wdseg

pSphinx = docXacSuatTuFileWDSEG("abc.wdseg")

return pSphinx

####################### XU LY BANG ALIZE: ==> P(Ai) ###############################

# Tinh xac suat P(Ai) bang cong cu Alize

# Dau vau:

# + Ten File Wav

# + So thu tu nguoi noi

# Dau ra: gia tri xac suat tim duoc

def tinhXacSuatAlize(fileWav):

# fileWav : AnhDT\_0\_05.wav

# lst: AnhDT\_0\_05

lst = fileWav.split(".")[0]

# Kiem tra neu da co file ./lbl/AnhDT\_0\_05.lbl : xoa file

try:

cmd = "rm -f ./TestAlize/" + lst + ".lbl"

#print cmd

os.system(cmd)

except Exception:

pass

pAlize = 0.5

#---CHAY RA FILE PRM .prm : BayNVN\_0\_03.prm

try:

sfbcep = "sfbcep -F PCM16 -f 16000 -p 19 -e -D -A /home/diep/input/" +\

lst + ".wav" +" TestAlize/" + lst + ".prm"

os.system(sfbcep)

except:

print "Loi chay lenh sfbcep"

#---HOAN THANH FILE PRM .prm

# Tao file .enr.prm: BayNVN\_0\_03.enr.prm

try:

NormFeat = "NormFeat --config ./TestAlize/cfg/NormFeat\_energy.cfg " +\

"--inputFeatureFilename " + lst

os.system(NormFeat)

except:

print "Loi chay lenh NormFeat"

# Tao file .lbl : BayNVN\_0\_03.lbl

try:

Energy = "EnergyDetector --config ./TestAlize/cfg/EnergyDetector.cfg " +\

"--inputFeatureFilename " + lst +\

" --verbose true --debug false 2>Energy 1>Energy"

os.system(Energy)

except:

print "Loi chay lenh Energy"

# Tao file .norm.prm : BayNVN\_0\_03.norm.prm

try:

NormFeatcfg = "NormFeat --config ./TestAlize/cfg/NormFeat.cfg " +\

"--inputFeatureFilename " + lst +\

" 2>logNormFeatcfg 1>logNormFeatcfg"

os.system(NormFeatcfg)

except:

print "Loi chay lenh NormFeatcfg"

# Tao file .res : BayNVN\_0\_03.res

try:

ComputeTest = "ComputeTest --config ./TestAlize/cfg/target.cfg " +\

"--ndxFilename ./TestAlize/testSpeaker" + lst + ".ndx " +\

"--outputFilename " + "./TestAlize/" + lst + ".res " +\

"--inputWorldFilename world --channelCompensation ABC"

print ComputeTest

os.system(ComputeTest)

except:

print "------------Loi chay lenh ComputeTest"

# xoa cac file khong can thiet

try:

# xoa file PRM: BayNVN\_0\_03.prm

cmd = "rm -f ./TestAlize/" + lst + ".prm"

os.system(cmd)

# xoa file .enr.prm: BayNVN\_0\_03.enr.prm

cmd = "rm -f ./TestAlize/" + lst + ".enr.prm"

os.system(cmd)

# xoa file LBL: BayNVN\_0\_03.lbl

cmd = "rm -f ./TestAlize/" + lst + ".lbl"

os.system(cmd)

# xoa file .norm.prm: BayNVN\_0\_03.norm.prm

cmd = "rm -f ./TestAlize/" + lst + ".norm.prm"

os.system(cmd)

# xoa file ndx: testSpeakerBayNVN\_0\_03.ndx

cmd = "rm -f ./TestAlize/testSpeaker" + lst + ".ndx"

os.system(cmd)

except Exception:

pass

# Chuong trinh nay chi nhan dang file dau tien trong danh sach

# File list : chua danh sach cac file Wav de thu nghiem nhan dang nguoi noi

fileLst = "lstTest10Tu"

# Danh sach file Wav can xu ly

lstWav = []

# Doc tu file "mySphinxModels.txt"

fileSphinxModels = "mySphinxModels.txt"

#fileAlizeModels = "./lst/all1.lst"

# Danh sach duong dan toi cac thu muc chua HMM (48), huan luyen boi Sphinx

hmmSphinxLst = []

# Tu dien phat am cua cac mo hinh HMM huan luyen boi Sphinx (48 nguoi noi)

dict = "mySphinx.dic"

hmmSphinxLst = docFileText(fileSphinxModels)

#print "File Sphinx Models: ", hmmSphinxLst

print "So mo hinh Sphinx: ", len(hmmSphinxLst)

lstWav = docFileText(fileLst)

print "Danh sach file Wav: ", lstWav

# File chua danh sach ten cua 48 nguoi noi

nametest="allten.txt"

# Danh sach ten cua 48 nguoi noi

filenametest = docFileText(nametest)

print filenametest

# File ket qua

fileKetqua = "ketqua1"

try:

os.system("rm -f " + fileKetqua)

except Exception:

pass

#lstAlize = docFileText(fileAlizeModels)

#print "Danh sach file Wav: ", lstAlize

fileWav = lstWav[0]

print "Tong so file Wav duoc nhan dang: ", len(lstWav)

# Thuc hien nhan dang tung file Wav

i=0

while i<len(lstWav):

fileWav = lstWav[i]

print "File Wav duoc nhan dang: ", fileWav

# Khong xet cac dong tro^'ng trong fileLst

if fileWav.strip() == "":

i = i+1

continue

#print "OK"

#nhandang1file(fileWav)

ghi\_ket\_qua\_1\_file(fileWav)

#break

i=i+1

#if i >= 2:

# break

D.2. Code chương trình chạy xác suất tiên nghiệm với ALIZE và Sphinx

#!/usr/bin/python

print "Hello World"

fileKq = "TONGHOP10TUAS"

alpha = 0.000000004

def taoFileChiSoNguoiNoi(idSpeaker):

#print "Chi so nguoi noi", idSpeaker

myName = {"0": "AnhDT", "1": "BayNVN", "2": "CaoNV", "3": "ChienVC", "4": "ChiLT", "5": "DiepDT", "6": "DungDT", "7": "DungPTN", "8": "DuongDD", "9": "HaiPN", "10":"HangDT", "11": "HanhTT", "12": "HiepTV", "13": "HoaVTT", "14": "HueNT", "15": "HungPM", "16": "HuongVK" ,"17": "HuyenNT" ,"18": "HuyLT", "19": "HuyLV", "20": "KhanhTV", "21": "KhoaDD", "22": "KhoeDD", "23": "KhuongNT", "24": "KienPT", "25": "KiPH", "26": "LanAnhDT", "27":"LienPTN", "28": "MinhNV","29": "NangNTH","30": "NhiNX","31": "QuynhDT","32": "QuynhNT","33": "SonVV","34":"SuotNTTK","35": "TamTT","36": "ThucBD","37": "ThuongND", "38": "ThanhLT","39": "ThuyNTTK","40": "ToPA","41": "TrangBTH","42": "TrangPT","43": "TuPT","44": "VanTT","45": "VuiLT","46":"YenNT", "47":"YenNTN"}

speaker = myName[str(idSpeaker)]

return speaker

# Doc fileKq

try:

fKq = open(fileKq, "r")

lines = fKq.readlines()

fKq.close()

except Exception:

print "Khong duoc duoc file ket qua"

exit(0)

print "Tong so file test: ", len(lines)

tong\_so\_file = len(lines)

so\_file\_dung = 0

f=open("Result " + fileKq,"a+")

for line in lines:

fileWav = line.split()[0]

pAlizeArr = line.split()[1:49]

pSphinxArr = line.split()[49:]

#print fileWav, " pAlizeArr: ", pAlizeArr

#print fileWav, " pSphinxArr: ", pSphinxArr

# Nhan dang nguoi noi

idSpeaker = 0

pSpeakerMax = float(pAlizeArr[0])+ float(pSphinxArr[0])\* alpha

#print "pAlizeArr[0] : ", pAlizeArr[0], "pSphinxArr[0]", pSphinxArr[0],

#print "pSpeakerMax : ", pSpeakerMax

for i in range(1, 48):

#print i

pSpeaker = float(pAlizeArr[i]) + float(pSphinxArr[i]) \* alpha

if pSpeaker > pSpeakerMax:

pSpeakerMax = pSpeaker

idSpeaker = i

print fileWav

nguoi\_noi = fileWav.split('\_')[0]

print nguoi\_noi,

nguoi\_noi\_kq = taoFileChiSoNguoiNoi(idSpeaker)

print nguoi\_noi\_kq

# f.write(nguoi\_noi + " " + nguoi\_noi\_kq)

#f.write("\n")

if nguoi\_noi == nguoi\_noi\_kq:

so\_file\_dung = so\_file\_dung + 1

# break

#else:

#f.write(" nguoi noi khong dung:" + nguoi\_noi +" ==> " +nguoi\_noi\_kq + " sao co " +"\n")

print "Tong so file : ", tong\_so\_file

print "So file dung : ", so\_file\_dung

print "Ty le:", (1.0\*so\_file\_dung/tong\_so\_file)\*100

alpha1 = "%.10f" % alpha

f.write("alpha: ")

f.write(alpha1)

#str = lst + " ==> " + speaker + " " + repr(pSpeakerMax) + "\n"

str1 = " Tong so file: " + repr(tong\_so\_file) + " ==> So file nhan dang dung " + repr(so\_file\_dung)

str2= " ==> Phan tram: " + repr((1.0\*so\_file\_dung/tong\_so\_file)\*100)+"%"

f.write(str1)

f.write(str2)

f.write("\n")