|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  BỘ MÔN KỸ THUẬT ĐO VÀ TIN HỌC CÔNG NGHIỆP  Description: C:\Users\Nguyen-Pro\Downloads\index.jpg  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  NGÀNH TIN HỌC CÔNG NGHIỆP  **Đề tài:**  **TƯƠNG TÁC VỚI ĐIỆN THOẠI THÔNG MINH BẰNG TIẾNG NÓI**  Sinh viên thực hiện: **NGUYỄN DUY BÌNH**  Lớp KSCLC-K53  Giảng viên hướng dẫn: TS. **TRẦN ĐỖ ĐẠT**  HÀ NỘI 06-2013 |

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. **Thông tin về sinh viên**

* Họ và tên sinh viên: Nguyễn Duy Bình
* Điện thoại liên lạc: 01689.928.600 Email: [hero.nguyenduybinh@gmail.com](mailto:hero.nguyenduybinh@gmail.com)
* Lớp: Tin học công nghiệp-KSCLC-K53 Hệ đào tạo : Chính quy
* Đồ án tốt nghiệp được thực hiện tại: Viện nghiên cứu quốc tế MICA
* Thời gian làm ĐATN: Từ ngày 22/02/2013 đến 14/06/2013

1. **Nhiệm vụ của đồ án**

Nhiệm vụ chung của nhóm VIVA là xây dựng chương trình VIVA làm thư ký ảo cho người dùng trên điện thoại Android với khả năng:

* Nghe được: khối nhận dạng tiếng nói.
* Nói được: khối tổng hợp tiếng Việt.
* Hiểu được yêu cầu của người dùng: khối hiểu câu lệnh.
* Thực hiện được chức năng người dùng yêu cầu: các khối chức năng.

Nhiệm vụ của cá nhân em, thành viên trong nhóm VIVA:

* Phân tích, thiết kế kiến trúc cho chương trình VIVA.
* Áp dụng kết quả từ chương trình Hoa Súng để xây dựng module tổng hợp tiếng của VIVA, chạy trên điện thoại.
* Xây dựng module chức năng tin nhắn bao gồm hai chức năng: gửi tin nhắn và đọc tin nhắn đến.
* Chuyển nội dung văn bản tiếng Việt không dấu thành có dấu phục vụ cho chức năng đọc tin nhắn.

1. **Lời cam đoan của sinh viên**

Tôi – **Nguyễn Duy Bình** - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của **TS Trần Đỗ Đạt**

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 tháng năm 2013

Tác giả ĐATN **Nguyễn Duy Bình**

1. **Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ.**  
   …………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Hà Nội, ngày tháng năm 2013

Giáo viên hướng dẫn

LỜI NÓI ĐẦU

Tương tác người – máy nói chung và tương tác người - máy bằng tiếng nói là một đề tài rất hay, đã được đưa ra và nghiên cứu từ lâu và vẫn đang tiếp tục phát triển với những thành tựu mới. Trong thời buổi hiện nay, với sự phát triển nhanh chóng của các thiết bị di động thông minh như điện thoại, máy tính bảng thì đề tài tương tác với điện thoại thông minh đang tạo nên một xu hướng công nghệ mới. Các công ty công nghệ lớn như Apple, Google, Samsung đã đưa chức năng này lên các thiết bị di động hàng đầu của mình như thư ký ảo Siri trong Iphone, công cụ Google Now trong hệ điều hành Android 4, SVoice trong dòng điện thoại Galaxy, cùng với nhiều ứng dụng khác cho phép người dùng tương tác với điện thoại bằng tiếng nói trên nhiều ngôn ngữ.

Tuy nhiên qua khảo sát, vẫn chưa có một chương trình nào cho phép người dùng tương tác với điện thoại thông minh bằng tiếng Việt mặc dù số lượng người dùng tiếng Việt sử dụng smartphone lên tới hàng triệu và đang tăng không ngừng. Chính vì thế trong quá trình thực tập tại viện nghiên cứu quốc tế MICA, nhóm VIVA đã được thành lập với 5 sinh viên khóa 53 dưới sự hướng dẫn của các thầy Ts.Trần Đỗ Đạt, Ts.Nguyễn Quốc Cường, Ts.Mạc Đăng Khoa với mục tiêu nghiên cứu cùng với sử dụng một số kết quả khoa học sẵn có của viện để xây dựng lên chương trình VIVA – thư ký ảo tiếng Việt trên điện thoại Android.

Trước khi trình bày báo cáo đồ án, em xin gửi lời cảm ơn tới các cán bộ MICA đã giúp đỡ nhóm VIVA và cá nhân em hoàn thành đồ án tốt nghiệp, đặc biệt là Ts.Trần Đỗ Đạt người trực tiếp hướng dẫn đồ án của em cùng với Ts.Mạc Đăng Khoa, Ts.Đỗ Thị Ngọc Diệp đã giúp đỡ em rất nhiều trong cả quá trình thực tập, làm đồ án tại viện MICA.

MỤC LỤC

[CHƯƠNG 1. GIỚI THIỆU 10](#_Toc358334121)

[1.1 Đặt vấn đề. 10](#_Toc358334122)

[1.2 Khảo sát các thống có chức năng tương tự. 11](#_Toc358334123)

[1.3 Dự án VIVA 13](#_Toc358334124)

[1.3.1 Giới thiệu 13](#_Toc358334125)

[1.3.2 Mục tiêu 13](#_Toc358334126)

[1.3.3 Chức năng 13](#_Toc358334127)

[1.4 Kế hoạch xây dựng VIVA 15](#_Toc358334128)

[1.5. Nội dung trình bày trong báo cáo đồ án 16](#_Toc358334129)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT & KIẾN TRÚC HỆ THỐNG VIVA. 17](#_Toc358334130)

[2.1 Sơ đồ khối hệ thống 17](#_Toc358334131)

[2.1.1 Sơ đồ tổng quan 17](#_Toc358334132)

[2.1.2 Mô tả các khối 18](#_Toc358334133)

[2.2 Khối tổng hợp tiếng Việt 20](#_Toc358334134)

[2.2.1. Cơ sở lý thuyết 20](#_Toc358334135)

[2.2.1.1.Lý thuyết tổng hợp tiếng 20](#_Toc358334136)

[2.2.1.2.Tổng hợp tiếng theo phương pháp ghép nối 20](#_Toc358334137)

[2.2.2 Chương trình Hoa Súng – MICA 23](#_Toc358334138)

[2.2.2.1 Giới thiệu chương trình 23](#_Toc358334139)

[2.2.2.2 Biểu đồ lớp của chương trình 23](#_Toc358334140)

[2.2.2.3 Quá trình tổng hợp tiếng nói của chương trình 24](#_Toc358334141)

[2.2.3 Chuyển đổi sang thư viện dùng cho Android 25](#_Toc358334142)

[2.2.3.1 Quá trình chuyển đổi 25](#_Toc358334143)

[2.2.3.1 Sử dụng trong Android project 27](#_Toc358334144)

[2.2.3.3 API 27](#_Toc358334145)

[2.3. Module nhận dạng tiếng Việt. 28](#_Toc358334146)

[2.3.1. Cơ sở lý thuyết. 28](#_Toc358334147)

[2.3.1.1 Phương pháp trích chọn đặc trưng MFCC 28](#_Toc358334148)

[2.3.1.2 Mô hình Markov ẩn liên tục 30](#_Toc358334149)

[2.3.2. Triển khai module nhận dạng trong VIVA. 31](#_Toc358334150)

[2.3.2.1 Xây dựng mô hình nhận dạng trên Sphinx4 31](#_Toc358334151)

[2.3.2.2 Xây dựng chương trình nhận dạng 31](#_Toc358334152)

[2.4. Module hiểu ngữ nghĩa. 33](#_Toc358334153)

[2.4.1 Cơ sở lý thuyết. 33](#_Toc358334154)

[2.4.1.1 Tách từ tiếng Việt 33](#_Toc358334155)

[2.4.1.1 Automat 34](#_Toc358334156)

[2.4.1.2 Automat hữu hạn đơn định – DFA 35](#_Toc358334157)

[2.4.2 Triển khai module hiểu ngữ nghĩa trong VIVA. 36](#_Toc358334158)

[2.5. Kiến trúc VIVA trên android. 36](#_Toc358334159)

[2.5.1. Giới thiệu kiến trúc Android. 36](#_Toc358334160)

[2.5.2. Kiến trúc ứng dụng VIVA. 37](#_Toc358334161)

[2.5.2.1. Biểu đồ lớp 37](#_Toc358334162)

[2.5.2.2. Mô tả lớp 39](#_Toc358334163)

[CHƯƠNG 3. KHỐI CHỨC NĂNG TIN NHẮN 41](#_Toc358334164)

[3.1 Chức năng gửi tin nhắn. 41](#_Toc358334165)

[3.1.1 Mô tả. 41](#_Toc358334166)

[3.1.2 Sơ đồ hoạt động. 42](#_Toc358334167)

[3.1.3 Biểu đồ tuần tự. 43](#_Toc358334168)

[3.2 Chức năng đọc tin nhắn đến. 45](#_Toc358334169)

[3.2.1 Mô tả. 45](#_Toc358334170)

[3.2.2 Sở đồ hoạt động. 46](#_Toc358334171)

[3.2.3 Biểu đồ tuần tự. 47](#_Toc358334172)

[CHƯƠNG 4. THÊM DẤU CHO VĂN BẢN TIẾNG VIỆT KHÔNG DẤU 48](#_Toc358334173)

[4.1. Lý do cần giải quyết bài toán. 48](#_Toc358334174)

[4.2. Hiện trạng bài toán. 48](#_Toc358334175)

[4.2.1. Các hướng tiếp cận đang có. 49](#_Toc358334176)

[4.2.2. Một số kết quả trong bài toán thêm dấu. 50](#_Toc358334177)

[4.2.2.1 Cho các ngôn ngữ khác 50](#_Toc358334178)

[4.2.2.2 Bài toán thêm dấu trong tiếng Việt 51](#_Toc358334179)

[4.2.3. Hướng tiếp cận bằng dịch máy (Machine translation). 52](#_Toc358334180)

[4.3. Hệ dịch máy Moses. 52](#_Toc358334181)

[4.3.1. Giới thiệu hệ dịch máy Moses 52](#_Toc358334182)

[4.3.2. Cơ sở lý thuyết dịch máy thống kê. 53](#_Toc358334183)

[4.4. Sử dụng Moses cho bài toán thêm dấu tiếng Việt. 56](#_Toc358334184)

[4.4.1. Bước 1: Cài đặt Moses Toolkit. 56](#_Toc358334185)

[4.4.2. Bước 2: Tạo corpus. 56](#_Toc358334186)

[4.4.3. Bước 3: Huấn luyện mô hình ngôn ngữ. 58](#_Toc358334187)

[4.4.4. Bước 4: Huấn luyện hệ thống. 58](#_Toc358334188)

[4.4.5. Bước 5: Kiểm tra. 59](#_Toc358334189)

[4.5. Triển khai module thêm dấu văn bản cho VIVA. 59](#_Toc358334190)

[4.5.1. Tạo server dịch. 60](#_Toc358334191)

[4.5.2. Xây dựng module thêm dấu trong VIVA. 61](#_Toc358334192)

[4.6 Đánh giá hệ thống. 62](#_Toc358334193)

[4.6.1 Đánh giá bằng so sánh Word-By-Word. 62](#_Toc358334194)

[4.6.2. Đánh giá bằng điểm BLEU. 63](#_Toc358334195)

[CHƯƠNG 5. CHƯƠNG TRÌNH VIVA PHIÊN BẢN DEMO 65](#_Toc358334196)

[5.1 Ghép nối các module 65](#_Toc358334197)

[5.2 Demo chương trình VIVA 65](#_Toc358334198)

[CHƯƠNG 6. TỔNG KẾT, ĐÁNH GIÁ 68](#_Toc358334199)

[6.1 Kết luận chung 68](#_Toc358334200)

[6.2 Hướng phát triển 69](#_Toc358334201)

DANH MỤC HÌNH VẼ

[Hình 1.1 Số thiết bị Android được kích hoạt mỗi ngày [1] 10](#_Toc358334202)

[Hình 1.2 Các nước có tốc độ tăng trưởng thiết bị Android & iOS cao nhất [2] 11](#_Toc358334203)

[Hình 1.3 Mô phỏng sơ đồ hoạt động của chương trình Siri [4] 12](#_Toc358334204)

[Hình 2.1 Sơ đồ khối hệ thống VIVA 17](#_Toc358334205)

[Hình 2.2 Mô hình tổng hợp tiếng nói 20](#_Toc358334206)

[Hình 2.3 Biểu đồ lớp của chương trình Hoa Súng 24](#_Toc358334207)

[Hình 2.4 Biểu đồ hoạt động tổng quát của chương trình Hoa Súng 25](#_Toc358334208)

[Hình 2.5 Triển khai module tổng hợp tiếng trên Android 26](#_Toc358334209)

[Hình 2.6 Copy thư viện, dữ liệu module tổng hợp tiếng vào project Android 27](#_Toc358334210)

[Hình 2.7 Sơ đồ thuật toán MFCC 29](#_Toc358334211)

[Hình 2.8 Mô hình Markov ẩn liên tục 31](#_Toc358334212)

[Hình 2.9 Sơ đồ khối chương trình nhận dạng 32](#_Toc358334213)

[Hình 2.10 Mô hình của một automat 34](#_Toc358334214)

[Hình 2.11 Đồ thị chuyển trạng thái của một automat hữu hạn đơn định 35](#_Toc358334215)

[Hình 2.12 Kiến trúc tổng thể Android 37](#_Toc358334216)

[Hình 2.13 Biểu đồ lớp của chương trình VIVA 39](#_Toc358334217)

[Hình 3.1 Sơ đồ hoạt động của chức năng gửi tin nhắn 42](#_Toc358334218)

[Hình 3.2 Biểu đồ tuần tự của chức năng gửi tin nhắn 44](#_Toc358334219)

[Hình 3.3 Sơ đồ hoạt động của chức năng đọc tin nhắn 46](#_Toc358334220)

[Hình 3.4 Biểu đồ tuần tự của chức năng đọc tin nhắn 47](#_Toc358334221)

[Hình 4.1 Mô hình hệ thống dịch máy thống kê 54](#_Toc358334222)

[Hình 4.2 Gióng hàng từ giữa hai câu trong hai ngôn ngữ 54](#_Toc358334223)

[Hình 4.3 Mô hình áp dụng dịch máy cho bài toán thêm dấu 56](#_Toc358334224)

[Hình 4.4 Chương trình tạo corpus tiếng Việt không dấu từ corpus tiếng Việt có dấu 57](#_Toc358334225)

[Hình 4.5 Các file tạo ra sau bước huấn luyện 59](#_Toc358334226)

[Hình 4.6 Chạy kiểm tra hệ dịch Moses 59](#_Toc358334227)

[Hình 4.7 Mô hình Client-Server cho bài toán thêm dấu 60](#_Toc358334228)

[Hình 4.8 Script khởi chạy server Moses 60](#_Toc358334229)

[Hình 4.9 Kiểm tra hoạt động của server Moses bằng công cụ NetCat 61](#_Toc358334230)

[Hình 4.10 Nội dung tin nhắn không dấu được thêm dấu trong chức năng đọc SMS 62](#_Toc358334231)

[Hình 4.11 Chạy Moses để thêm dấu cho file 10.000 câu tiếng Việt không dấu 62](#_Toc358334232)

[Hình 4.12 Chương trình so sách file kết quả thêm dấu và file nguồn 63](#_Toc358334233)

[Hình 4.13 File thông tin so sánh 63](#_Toc358334234)

[Hình 4.14 Tính điểm BLEU 64](#_Toc358334235)

[Hình 5.1Màn hình khởi động & chào hỏi của VIVA 65](#_Toc358334236)

[Hình 5.2 Màn hình thực hiện chức năng gửi tin nhắn 66](#_Toc358334237)

[Hình 5.3 Màn hình đọc tin nhắn đến 67](#_Toc358334238)

[Hình 5.4 Màn hình chức năng trả lời thời tiết 67](#_Toc358334239)

[Hình 5.5Màn hình chức năng trả lời giá vàng 68](#_Toc358334240)

[Hình 5.6 Màn hình chức năng trả lời tỉ giá 68](#_Toc358334241)

DANH SÁCH BẢNG BIỂU

[Bảng 1.1 Bảng phân công công việc cho các thành viên nhóm VIVA 15](#_Toc358334242)

[Bảng 2.1 API module tổng hợp tiếng trên Android 27](#_Toc358334243)

[Bảng 4.1 Dấu trong các ngôn ngữ viết trên bảng chữ Latin trong khối Châu Âu[10] 48](#_Toc358334244)

[Bảng 4.2 Thống kê tỉ tệ từ có dấu và tỉ lệ nhập nhằng của từ không dấu trong một số ngôn ngữ 49](#_Toc358334245)

[Bảng 4.3 Luật bỏ dấu cho các ký tự trong tiếng Việt 57](#_Toc358334246)

DANH SÁCH CÁC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| ASR | Automatic Speech Recognition | DFT | Discrete Fourier Transform |
| HMI | Human Machine Interface | IDFT | [Inverse Discrete Fourier Transform](https://www.google.com.vn/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&sqi=2&ved=0CEEQFjAC&url=http%3A%2F%2Fwww.mathworks.com%2Fhelp%2Fmatlab%2Fref%2Fifft.html&ei=2YulUbPHBLGujALugoHwBQ&usg=AFQjCNFwNcZQirnqABnx3vCGFgoiTPzsuw&sig2=6sdckD5nJ3GJHTF1_OWe2Q&bvm=bv.47008514,d.cGE) |
| WER | Word Error Rate | IFFT | [Inverse Fast Fourier Transform](https://www.google.com.vn/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&sqi=2&ved=0CEEQFjAC&url=http%3A%2F%2Fwww.mathworks.com%2Fhelp%2Fmatlab%2Fref%2Fifft.html&ei=2YulUbPHBLGujALugoHwBQ&usg=AFQjCNFwNcZQirnqABnx3vCGFgoiTPzsuw&sig2=6sdckD5nJ3GJHTF1_OWe2Q&bvm=bv.47008514,d.cGE) |
| LPC | Linear Prediction Coding | VAD | Voice Automatic Detection |
| MFCC | Mel – Frequency Cepstrum Coefficients | PSOLA | Pitch Synchronous Overlap-Add |
| PCA | Principle Components Analysis | TTS | Text To Speech |
| DTW | Dynamic Time Warping | MFC | Microsoft Foundation Class |
| VQ | Vector Quantization | DFA | Deterministic Finite Automata |
| HMM | Hidden Markov Modeling | MT | Machine Translattion |
| ANN | Artifitial Neural Network | POS | Part Of Speech |
| DCT | Discrite Cosin Transform | BLEU | Bilingual Evaluation Understudy |

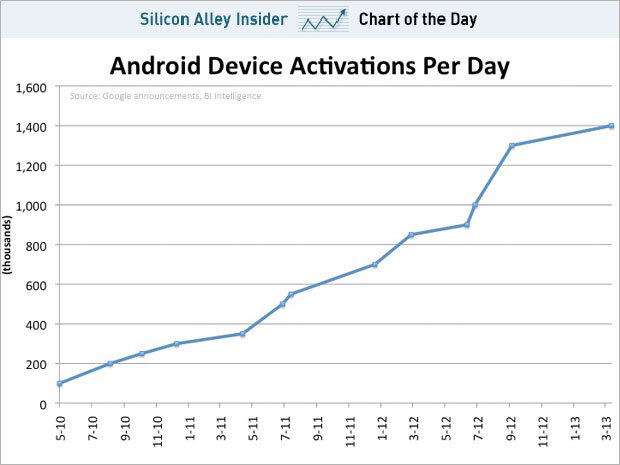
# GIỚI THIỆU

## 1.1 Đặt vấn đề.

Tương tác người – máy bằng tiếng nói là mảng đề tài rộng, mặc dù không mới nhưng luôn phát triển từng ngày về công nghệ, với mục tiêu chung là làm cho máy móc nghe được, hiểu được yêu cầu của con người và thực hiện.

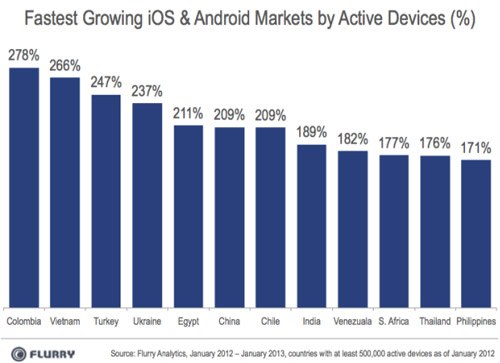
Các thiết bị di động thông minh như Smartphone, Tablet đang phát triển với tốc độ vũ bão và dần trở thành một phần không thể thiếu trong cuộc sống của con người hiện đại. Chiếc điện thoại giờ đây không chỉ là một thiết bị nghe gọi, những chiếc điện thoại thông minh có thể phục vụ con người trong tất cả các lĩnh vực giải trí, công việc, học tập, cuộc sống thường nhật. Giao tiếp với điện thoại thông minh bằng tiếng nói là một đề tài thú vị, một xu hướng công nghệ hiện nay, với mục tiêu biến chiếc điện thoại thành thư ký ảo cho người dùng, giúp đỡ người dùng thực hiện các thao tác mà không cần chạm vào màn hình, tạo ra sự tiện lợi và trải nhiệm thú vị.

Theo thống kê [1], chỉ tính riêng hệ điều hành Android, tính đến tháng 03/2013 đã có tổng cộng 1.230.000.000 thiết bị Android được kích hoạt, và hiện tại mỗi ngày có thêm 1.400.000 thiết bị Android được kích hoạt.



Hình 1.1 Số thiết bị Android được kích hoạt mỗi ngày [1]

Tính đến tháng 11/2013 Việt Nam đang là nước có tốc đột tăng trưởng về thiết bị Android & iOS đứng thứ hai thế giới [2]



Hình 1.2 Các nước có tốc độ tăng trưởng thiết bị Android & iOS cao nhất [2]

Rất nhiều hãng sản xuất điện thoại thông minh, công ty công nghệ phần mềm hàng đầu trên thế giới đang đầu tư, nghiên cứu phát triển các hệ thống tương tác với smartphone bằng tiếng nói như Apple, Google, Nuance, Samsung… với các sản phẩm Siri, Google Now, Nuance Talks, S Voice…

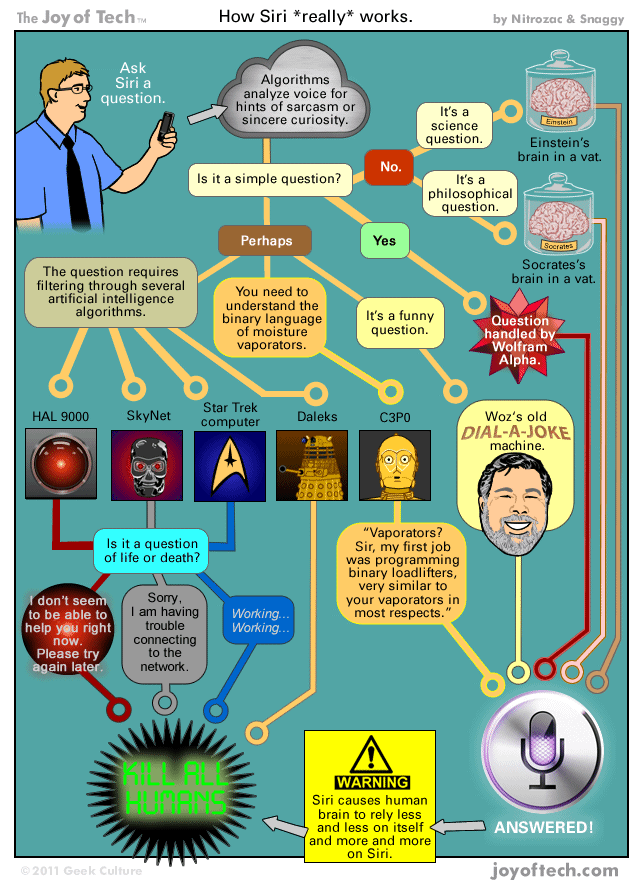
Tuy nhiên chưa hề có một hệ thống giúp người dùng tương tác với điện thoại bằng tiếng Việt, chính vì vậy chúng em đã lựa chọn đề tài này làm đồ án tốt nghiệp.

## 1.2 Khảo sát các thống có chức năng tương tự.

Trên thế giới các hãng công nghệ đã xây dựng các hệ thống tương tác với điện thoại qua tiếng nói như thế nào? Trước hết chúng ta hãy khảo sát các hệ thống hiện có trên thế giới để trả lời câu hỏi này.

a. Siri [3]

* Công bố vào tháng 11/2011, do Apple phát triển, chạy trên nền tảng hệ điều hành iOS.
* Ngôn ngữ hỗ trợ: English, French, German, Japanese, Spanish, Italian, Korean, Madarin, Cantonese.
* Các chức năng chính: bản đồ, tìm kiếm trên web, ghi chú, gửi Twitter tweet , tìm liên hệ trong danh bạ, mở ứng dụng, tìm địa danh, gọi điện, gửi email, SMS, đặt báo thức, truy vấn thông tin: thời tiết, thể thao, phim, kiến thức (Wolfram Alpha).
* Sơ đồ hoạt động:



Hình 1.3 Mô phỏng sơ đồ hoạt động của chương trình Siri [4]

b. Google Now [5]

Phát triển bởi GooglFigure 1e, được tích hợp với Android OS 4.1, công bố vào tháng 07/2012.

* Ngôn ngữ hỗ trợ: English.
* Chức năng chính: thống kê thông số sức khỏe người dùng khi chạy bộ, đạp xe, tìm kiếm ngày sinh, sự kiện, lịch bay, thao tác với Gmail, tìm kiếm thông tin: phim, tin tức, cuộc họp, hình ảnh, địa điểm, đề tài nghiên cứu, thể thao, chứng khoán, thông tin giao thông, du lịch, dịch máy, thông tin thời tiết.

c. Một số hệ thống khác: TellMe (Microsoft), S-Voice (Samsung), Iris (Dexetra), Speaktoit Assistant (Speaktoit, Inc), Vlingo (Vlingo, Inc), Evi (Evi Technologies Ltd).

## 1.3 Dự án VIVA

### 1.3.1 Giới thiệu

Để thực hiện đề tài tương tác với smartphone bằng tiếngViệt, dự án VIVA (**VI**etnamese **V**oice **A**ssistant) được thành lập tại viện nghiên cứu MICA. Dự án được thực hiện bởi các sinh viên: Nguyễn Duy Bình, Nguyễn Hữu Hiển, Nguyễn Thanh Tùng, Nguyễn Văn Hiếu, Nguyễn Văn Bảo, dưới sự dẫn dắt của TS.Trần Đỗ Đạt, TS.Nguyễn Quốc Cường và TS.Mạc Đăng Khoa.

### 1.3.2 Mục tiêu

Dự án VIVA được thành lập với mục tiêu xây dựng hệ thống có thể:

* Nghe được: nhận dạng tiếng nói (tiếng Việt).
* Hiểu được: hiểu được câu lệnh của người dùng.
* Nói được: phản hồi lại người dùng bằng tiếng nói.
* Làm thư ký: giúp người dùng thực hiện các thao tác trên điện thoại mà không cần chạm vào bàn phím hay mèn hình để điều khiển, các tương tác sẽ được yêu cầu, trả lời kết quả, thông tin qua tiếng nói.

Dựa trên sự hướng dẫn của các giảng viên trên MICA cùng với một số kết quả khoa học đã được nghiên cứu trên MICA, nhóm VIVA có nhiệm vụ:

* Xây dựng mới khối nhận dạng tiếng nói tiếng Việt.
* Xây dựng mới khối hiểu câu lệnh tiếng Việt.
* Áp dụng kết quả tổng hợp tiếng nói từ chương trình Hoa Súng để xây dựng module tổng hợp tiếng.
* Xây dựng ứng dụng thư ký ảo VIVA trên điện thoại thông minh, kết nối ba khối trên cùng với các module thực thi chức năng.
* Xây dựng VIVA theo hướng mở: có thể mở rộng được miền nhận dạng, khối hiểu câu lệnh, thêm mới các module chức năng VIVA có thể thực hiện.

### 1.3.3 Chức năng

Dựa trên những khảo sát từ các hệ thống tương tự đang có, VIVA sẽ được phát triển trước tiên trên hệ điều hành Android với những chức năng sau:

1. Giao tiếp đơn giản.
   1. Thư ký ảo có khả năng giao tiếp với người dùng các câu tiếng Việt thông dụng.
      * Chào hỏi
      * Trả lời tên, tuổi
      * Trả lời tâm trạng...
   2. Các câu giao tiếp có ý nghĩa phù hợp với ngữ cảnh giao tiếp.
   3. Cùng một câu nói của người dùng, thư ký ảo có thể đưa ra 1 hoặc nhiều câu trả lời dạng khác nhau để việc giao tiếp không trở nên nhàm chán.
   4. Thư ký ảo có thể chủ động giao tiếp với người dùng:
      1. Chào, hỏi người người dùng khi mở ứng dụng.
         * Hôm nay tâm trạng bạn thế nào? Bạn có muốn nghe một bản nhạc không?
         * Tôi có thể giúp gì cho bạn?
         * Bạn đang có 1 email, tin nhắn chưa đọc.Bạn có muốn tôi mở email/ trả lời tin nhắn không?
      2. Đặt câu hỏi khi không nhận dạng được tiếng nói hay không thể phân tích chính xác nghĩa của từ mà người dùng nói.
         * Ý của bạn là .... phải không ?
         * Tôi không nghe rõ, bạn vui lòng nhắc lại
   5. Hệ thống có khả năng tiếp nhận, mở rộng thêm dữ liệu về các câu giao tiếp để có thể làm cho khả năng giao tiếp của thư ký ảo ngày càng phong phú, chuẩn xác hơn.
2. Tương tác với ứng dụng.
   1. Đóng, mở ứng dụng theo tên.
   2. Tương tác với các ứng dụng nền tảng có sẵn trên hệ điều hành Android.
      1. Ứng dụng nhắn tin:
         1. Soạn, gửi tin nhắn tới số điện thoại hoặc tên người nhận trong danh bạ.
         2. Thông báo, đọc, trả lời tin nhắn tới.
      2. Ứng dụng gọi điện:
         1. Gọi điện tới số điện thoại.
         2. Gọi điện tới tên liên lạc trong danh bạ
         3. Tra cứu cước viễn thông, số tiền trong tài khoản
      3. Ứng dụng Email:
         1. Soạn, gửi email
         2. Thông báo, đọc, trả lời email đến
      4. Ứng dụng bản đồ
         1. Tìm một địa danh theo tên trên bản đồ
         2. Trả lời khoảng cách giữa hai địa điểm
         3. Tìm các địa điểm theo từ khóa gần người dùng: cây ATM, quán cà phê, nhà hàng, khách sạn, sân bóng…
      5. Ứng dụng báo thức
         1. Đặt báo thức
         2. Hủy báo thức
      6. Ứng dụng Google Search
         1. Mở Google tìm kiếm theo từ khóa
      7. Ứng dụng chơi nhạc, video
         1. Mở bài hát trong playlist
         2. Mở video trong playlist
      8. Notification.
         1. Đọc các thông báo của hệ điều hành.(Nếu ngôn ngữ hệ điều hành đang sử dụng khác tiếng Việt thì qua một bước dịch sang tiếng Việt).
   3. Các ứng dụng phổ biến có cung cấp API (Application Programming Interface)
      1. Mạng xã hội Facebook
         1. Đăng bài viết
         2. Chụp hình và đăng lên trang cá nhân
      2. Mạng xã hội Twiter
         1. Tweet - đăng tin nhắn lên Twiter
      3. Mạng xã hội Instagram
         1. Chụp hình và đăng lên trang cá nhân
3. Truy vấn thông tin: người dùng hỏi, thư ký ảo tìm & trả lời các thông tin.
   1. Nguồn thông tin trên internet:
      1. Tỉ giá tiền tệ.
      2. Thời tiết.
      3. Giá vàng.
      4. Chứng khoán.
      5. RSS tin tức từ các trang báo mạng tiếng Việt.
      6. Thông tin về con người, sự vật, sự việc từ nguồn Wikipedia tiếng Việt.
   2. Nguồn thông tin trên điện thoại:
      1. Thời gian, ngày tháng.

Trong khuôn khổ đồ án tốt nghiệp, nhóm đặt mục tiêu và hoàn thành các khối nhận dạng tiếng nói, khối hiểu câu lệnh, khối tổng hợp, thiết kế và xây dựng cở sở hạ tầng cho ứng dụng VIVA trên nền Android 2.3 cùng với các chức năng cơ bản: **2.2.1, 3.1.1, 3.1.2, 3.1.3**, việc xây dựng thêm các chức năng còn lại sẽ được triển khai trong thời gian tới.

## 1.4 Kế hoạch xây dựng VIVA

Mỗi thành viên trong VIVA được phân công một mảng, học tập và nghiên cứu một mặt để xây dựng VIVA, một mặt phát triển làm đồ án tốt nghiệp. Nhiệm vụ của mỗi thành viện được liệt kê trong bảng sau:

Bảng 1.1 Bảng phân công công việc cho các thành viên nhóm VIVA

|  |  |
| --- | --- |
| **Thành viên** | **Công việc** |
| Nguyễn Duy Bình | 1. Phân tích, thiết kế cơ sở hạ tầng cho ứng dụng VIVA. 2. Áp dụng kết quả từ chương trình Hoa Súng để xây dựng module tổng hợp tiếng của VIVA, chạy trên điện thoại. 3. Xây dựng module chức năng tin nhắn. 4. Chuyển nội dung văn bản tiếng Việt không dấu thành có dấu phục vụ cho chức năng đọc tin nhắn. 5. Viết chương trình thu âm để tạo cơ sở dữ liệu huấn luyện khối nhận dạng. |
| Nguyễn Hữu Hiển | 1. Xây dựng module thu tiếng nói, tiền xử lý làm đầu vào cho module nhận dạng tiếng nói. 2. Xây dựng module phát hiện tiếng nói, tự động bắt đầu thu và ngắt thu âm khi người dùng nhập câu lệnh bằng tiếng nói. |
| Nguyễn Thanh Tùng | 1. Xây dựng module nhận dạng tiếng nói dựa trên Sphinx 4. |
| NguyễnVăn Hiếu | 1. Xây dựng khối hiểu câu lệnh. 2. Ghép nối các module vào ứng dụng VIVA. |
| Nguyễn Văn Bảo | 1. Nâng cao chất lượng module tổng hợp tiếng. 2. Xây dựng module chức năng truy vấn thông tin. |

## 1.5. Nội dung trình bày trong báo cáo đồ án

Đồ án bao gồm 6 chương:

* *Chương 1:* Giới thiệu

*Trình bày tổng quan về đề tài tương tác với điện thoại thông minh bằng tiếng nói, giới thiệu dự án VIVA, giới thiệu các thành viên và công việc của từng người.*

* *Chương 2:* Cơ sở lý thuyết, thiết kế hệ thống VIVA.

*Trình bày sơ đồ khối hệ thống VIVA, tổng quan cơ sở lý thuyết và cấu trúc từng khối của hệ thống. Nhấn mạnh khối tổng hợp tiếng nói và việc áp dụng kết quả từ chương trình Hoa Súng.*

* *Chương 3:* Module chức năng tin nhắn.

*Trình bày thiết kế, quá trình triển khai module chức năng tin nhắn, bao gồm hai chức năng: gửi tin nhắn (tới số điện thoại hoặc tên người trong danh bạ), thông báo và đọc tin nhắn đến.*

* *Chương 4:* Thêm dấu cho văn bản tiếng Việt không dấu.

*Trình bày bài toán thêm dấu cho văn bản tiếng Việt không dấu: ngữ cảnh, hiện trạng của bài toán, hướng giải quyết, quá trình triển khai, kết quả và đánh giá.*

* *Chương 5:* Chương trình VIVA phiên bản demo.

*Trình bày quá trình ghép nối, chạy thử chương trình VIVA phiên bản Demo.*

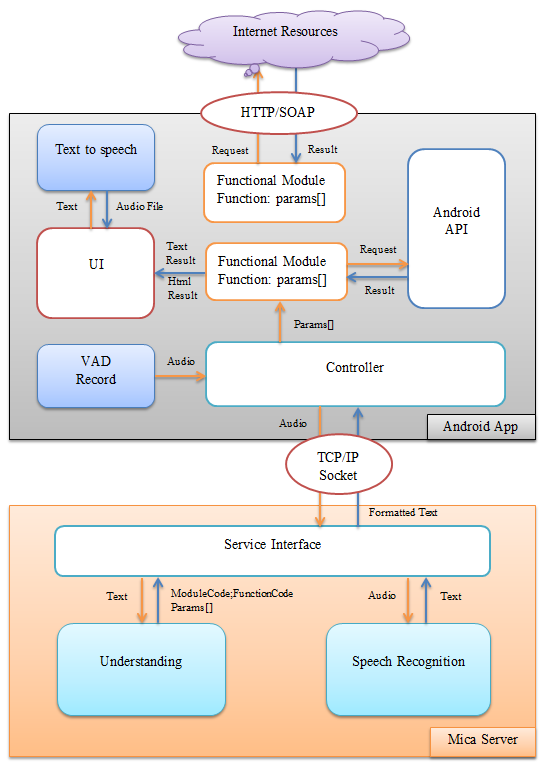
* *Chương 6:* Tổng kết.

*Tổng kết lại quá trình xây dựng VIVA, những việc đã làm được và chưa làm được, đánh giá kết quả và đưa ra hướng phát triển tiếp theo.*

# CƠ SỞ LÝ THUYẾT & KIẾN TRÚC HỆ THỐNG VIVA.

## 2.1 Sơ đồ khối hệ thống

### 2.1.1 Sơ đồ tổng quan



Hình 2.1 Sơ đồ khối hệ thống VIVA

### 2.1.2 Mô tả các khối

Hệ thống VIVA bao gồm các khối chính sau :

1. ***Khối VAD :***

* **Mô tả :** Tiếng nói của người dùng được thu âm qua khối VAD, khối VAD tự động phát hiện tiếng nói để bắt đầu ghi âm, tự động ngắt ghi âm khi người dùng ngừng nói quá một khoảng thời gian đặt trước (0.5giây). Sau các bước lọc nhiễu, nén file, khối VAD gửi file âm thanh san khối điều khiển chính để gửi lên server nhận dạng.
* **Đầu vào :** Tiếng nói
* **Đầu ra :** File thu âm

1. ***Khối Controller (khối điều khiển):***

* **Mô tả:** Khối Controller làm nhiệm vụ giao tiếp với Server chạy khối nhận dạng và khối hiểu qua TCP/IP socket. Khối Controller gửi file ghi âm đã được thu và xử lý bởi khối VAD lên Server MICA, sau khi nhận được kết quả là đoạn text có định dạng, chứa thông tin nội dung câu nói hệ thống nhận dạng được, mã module chức năng, mã lệnh và các giá trị tham số sẽ thực hiện. Từ kết quả trả về, khối Controller thực hiện tìm tới module chức năng và gọi thực thi lệnh tương ứng với các tham số đầu vào nếu có.
* **Đầu vào:** File ghi âm từ khối VAD.
* **Đầu ra:** Kích hoạt module chức năng và lệnh tương ứng.

1. ***Khối Service Interface (Khối giao diện dịch vụ đặt trên Server MICA):***

* **Mô tả:** Khối làm nhiệm vụ tiếp nhận các request TCP/IP Socket từ các client do khối Controller gửi lên. Từ các file ghi âm gửi lên, khối gọi tới khối nhận dạng để thu về câu nói dạng text tương ứng, tiếp theo khối gọi tới khối hiểu để phân tích yêu cầu của người dùng và trả lại kết quả về client.
* **Đầu vào:** File ghi âm.
* **Đầu ra:** Đoạn text chứa câu nhận dạng được và kết quả phân tích bởi khối hiểu.

1. ***Khối nhận dạng tiếng nói:***

* **Mô tả:** Khối làm nhiệm vụ nhận dạng các file ghi âm và trả về câu nói dạng text tương ứng.
* **Đầu vào:** File ghi âm.
* **Đầu ra:** Text.

1. ***Khối hiểu câu lệnh:***

* **Mô tả:** Khối làm nhiệm vụ phân tích ngữ nghĩa câu lệnh của người dùng, từ đó xác định xem hệ thống phải gọi tới module chức năng nào, thực thi lệnh với các tham số trích xuất được trong câu lệnh để đáp ứng lại người dùng.

Thí dụ: người dùng yêu cầu “tôi muốn xem thông tin thời tiết ngày mai tại hà nội”, khối hiểu sẽ phân tích và trả về:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *ModuleCode* | QUERYINFO | *FunctionCode* | WEATHER | *Param* | Hà nội | *Param* | Ngày mai |

* **Đầu vào:** Text.
* **Đầu ra:** Text có định dạng.

1. ***Các khối chức năng:***

* **Mô tả:** Các khối chức năng bao gồm các function làm nhiệm vụ thức thi yêu cầu của người dùng, các khối này được xây dựng theo danh sách các chức năng của VIVA liệt kê tại mục **1.3.3**.

Các chức năng sau khi được kích hoạt bởi khối Controller sẽ chạy theo một kịch bản được lập trình sẵn, trong quá trình thực thi có thể sẽ phải gọi tới API của hệ điều hành để thực hiện thao tác (nhắn tin, gọi điện, gửi email…) hoặc truy vấn tới nguồn thông tin trên Internet để lấy thông tin trả lời người dùng (thời tiết, tỉ giá, giá vàng, chứng khoán…)

Sau khi thực thi xong, sẽ gọi tới khối UI để hiển thị kết quả dạng text hoặc dạng Html view.

* **Đầu vào:** Tham số.
* **Đầu ra:** Text, Html result.

1. ***Khối UI (giao diện):***

* **Mô tả:** Khối làm nhiệm vụ hiển thị kết quả text – câu trả lời của VIVA và các dữ liệu trực quan dạng Html (giá vàng, thời tiết…) cho người dùng. Đồng thời với việc hiển thị câu trả lời của VIVA lên màn hình, khối UI sẽ gọi tới khối tổng hợp tiếng nói để thu về file tiếng và phát trả lời người dùng.
* **Đầu vào:** Câu trả lời dạng text, dữ liệu hiển thị dạng Html.
* **Đầu ra:** Kết quả hiện lên màn hình.

1. ***Khối tổng hợp tiếng nói:***

* **Mô tả:** Khối làm nhiệm vụ tổng hợp câu nói dạng text đưa vào và trả về file tiếng nói dạng âm thanh.
* **Đầu vào:** Text.
* **Đầu ra:** File tiếng nói.
* Khối nhận dạng tiếng nói và khối hiểu ngữ nghĩa phức tạp về cấu trúc, khi thực thi chiếm dụng tài nguyên (không gian ổ cứng, bộ nhớ, xử lý của CPU) rất lớn nên chưa thể tích hợp vào ứng dụng trên điện thoại mà được triển khai trên server của MICA, cho các client kết nối tới bằng TCP/IP socket. Chính vì lý do này, hệ thống VIVA khi chạy luôn luôn cần kết nối mạng internet.

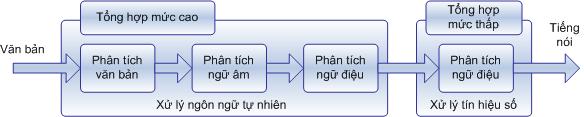
## 2.2 Khối tổng hợp tiếng Việt

### 2.2.1. Cơ sở lý thuyết

#### 2.2.1.1.Lý thuyết tổng hợp tiếng

Một nhu cầu rất quan trọng trong lĩnh vực tổng hợp tiếng nói là tổng hợp tiếng nói từ văn bản (Text To Speech – TTS). Quá trình này được chia làm hai mức xử lý:

* High Level Synthesis: Tổng hợp mức cao.
* Low Level Synthesis: Tổng hợp mức thấp.



Hình 2.2 Mô hình tổng hợp tiếng nói

* [**Tổng hợp mức cao**](#_Toc60680202)

Tổng hợp mức cao là giai đoạn đầu của quá trình tổng hợp, giai đoạn chuyển đổi các văn bản text thành các đơn vị tiếng nói (ví dụ như diphone). Văn bản được nhập hoặc sao chép vào, sau đó qua tổng hợp mức thấp sẽ thành tiếng nói.

Tổng hợp mức cao gồm 3 bước:

* Xử lý trước văn bản với các chữ số, các ký tự đặc biệt, chữ viết tắt, và những từ viết tắt được ghép bằng các chữ đầu của các từ đầy đủ...
* Phân tích cách phát âm của từ, kể cả từ đồng âm khác nghĩa và các tên riêng.
* Phân tích ngữ điệu của tiếng nói.

Sau khi tổng hợp mức cao, thông tin được cung cấp cho hệ thống mức thấp để điều khiển. Chẳng hạn, với bộ tổng hợp formant thì cần các thông tin như tần số cơ bản, tần số formant, khoảng thời gian, và biên độ của mỗi đoạn âm thanh.

* **[Tổng hợp mức thấp](#_Toc60680206)**

Tổng hợp mức thấp là quá trình tạo ra tín hiệu tiếng nói từ các thông tin được đưa tới từ tổng hợp mức cao (chuỗi các đơn vị âm và thông tin ngữ điệu).

Có 3 phương pháp cơ bản được thực hiện ở mức tổng hợp mức thấp:

* Phương pháp mô phỏng hệ thống phát âm.
* Phương pháp tổng hợp formant.
* Phương pháp tổng hợp ghép nối .

#### 2.2.1.2.Tổng hợp tiếng theo phương pháp ghép nối

Trong tổng hợp tiếng nói theo kiểu ghép nối, một đơn vị âm được tổng hợp đơn giản bằng cách lấy ra đoạn sóng âm tương ứng với âm vị tương ứng. Một lời phát biểu được tổng hợp bằng cách ghép nối đồng thời nhiều đơn vị âm lại với nhau. Tuy nhiên, nếu ta ghép nối 2 đoạn âm không liền kề nhau sẽ dẫn đến việc không liên tiếp về phổ và không liên tiếp về ngôn điệu. Sự không liên tiếp về phổ xảy ra khi formant tại điểm nối không hợp với nhau. Sự không liên tiếp về ngôn điệu xảy ra khi cao độ tại điểm nối là không hợp với nhau. Chất lượng âm thanh tổng hợp sẽ tồi khi mà có nhiều điểm không liên tiếp trong ghép nối, cho dù các đơn vị âm là hoàn toàn tự nhiên.

Thường thì, các hệ thống tổng hợp tiếng nói theo kiểu kết nối hay có sự thay đổi về chất lượng, có thể một hệ thống có chất lượng tốt, và hệ thống khác có thể là chất lượng tồi. Nếu có đủ lượng đơn vị âm, chất lượng âm thanh tổng hợp có thể tốt như âm thanh lúc thu. Tuy nhiên nếu xuất hiện một vài sự không liên tục sẽ gây cho chất lượng âm thành tổng hợp sẽ kém đi nhiều.

Chính vì vậy, khi thiết kế một hệ thống tổng hợp tiếng nói, ta cần quan tâm đến các vấn đề quan trọng sau:

* Lựa chọn loại đơn vị âm tối ưu
* Xây dựng cơ sở dữ liệu của các đơn vị âm
* Tìm kiếm đơn vị âm tối ưu để ghép nối
* Ghép nối các đơn vị âm
* **Lựa chọn loại âm vị tối ưu:**

Tiếng Việt là ngôn ngữ đơn âm tiết. Âm tiết là đơn vị âm thanh nhỏ nhất có thể cảm nhận được. Cấu trúc của âm tiết trong tiếng Việt gồm 5 phần: âm đầu, âm đệm, âm chính, âm cuối và thanh điệu.

Tổng quan về các loại đơn vị âm trong tiếng Việt, các nghiên cứu trên Viện MICA cho thấy rằng, đối với tiếng Việt tồn tại bốn kiểu đơn vị âm (âm vị kép, bán âm tiết, âm đầu/vần, âm tiết) có thể sử dụng cho quá trình tổng hợp. Tuy nhiên kiểu đơn vị âm âm đầu/vần sẽ không phải là loại đơn vị âm tối ưu do sự không liên tục giữa âm đầu (initial part) và vần (final part). Còn đối với trường hợp đơn vị âm là âm tiết, để một hệ thống có thể tổng hợp được bất kỳ đoạn văn bản nào thì đòi hỏi phải có một cơ sở dữ liệu rất lớn. Do vậy mà âm vị kép và bán âm tiết hiện vẫn đang là những loại đơn vị âm được sử dụng phổ biến nhất trong các hệ thống tổng hợp tiếng nói bằng phương pháp ghép nối. Viện nghiên cứu MICA cũng đang nghiên cứu và sử dụng chủ yếu hai loại đơn vị âm này trong chương trình tổng hợp tiếng nói được phát triển tại Viện và trong đó, cơ sở dữ liệu cho loại đơn vị âm là bán âm tiết được xây dựng và phát triển đầy đủ nhất.

* **Xây dựng cơ sở dữ liệu của các đơn vị âm**

Tiếng Việt là ngôn ngữ có thanh điệu, ngoài việc ảnh hưởng của khớp nối giữa các đơn vị âm cạnh nhau, đơn vị âm cũng bị ảnh hưởng bởi thanh điệu của âm tiết. Trong ngôn ngữ có thanh điệu đơn vị âm được chia thành hai loại: đơn vị âm không thanh điệu và đơn vị âm có thanh điệu. Đơn vị âm không thanh điệu là đơn vị âm thu được nhờ việc tách ra từ các âm tiết có thanh ngang. Đơn vị âm có thanh điệu thu được từ các âm tiết chứa thông tin về thanh điệu. Với đơn vị âm không thanh điệu, sau khi ghép nối để tổng hợp nên các âm tiết không thanh điệu, thanh điệu được thêm vào bằng cách điều khiển trực tiếp sự thay đổi của F0 của âm tiết. Trong khi đó, với đơn vị âm có thanh điệu, những thông tin về thanh điệu được lưu trong đơn vị âm, vì vậy ta không cần phải điều khiển đường cong F0.

* **Tìm kiếm đơn vị âm tối ưu để ghép nối**

Bước tìm kiếm đơn vị âm tối ưu nhằm tìm kiếm trong cơ sở dữ liệu các đơn vị âm tốt nhất sao cho sau khi ghép nối ta thu được chất lượng tiếng nói là tốt nhất có thể. Các đơn vị âm tốt nhất là các đơn vị thỏa mãn sao cho độ méo tiềm tàng (về ngữ điệu, về phổ) giữa chúng là thấp nhất.

Có hai phương pháp để chọn lựa ra các đơn vị âm tối ưu cho quá trình tổng hợp:

* *Chọn lựa dựa trên mô hình cây quyết định:* dữ liệu đọc được nhóm lại trong một cây bằng cách phân đoạn mỗi nút thành các nút cấp dưới dựa trên dữ liệu âm học, bằng cách sử dụng các tiêu chuẩn được đề xuất bởi nhãn ngữ cảnh của dữ liệu. Điều này tạo ra một số lượng nhóm, mỗi nhóm chứa các phân đoạn giống nhau ở cùng một mức ngữ cảnh và âm học. Nhóm được sử dụng cho một ngữ cảnh nào đó trong quá trình tổng hợp có thể lấy ra từ cây tương ứng hoặc sử dụng các kết quả giống về ngữ cảnh nhằm mục đích so sánh nhãn của ngữ cảnh yêu cầu với nhãn của các nhóm có thể sử dụng được.
* *Chọn lựa dựa trên việc tối ưu hóa hàm giá trị:* phương pháp này sẽ tìm ra các units dựa trên một lượng các instances cho trước với mục đích là tối thiểu hóa hàm giá trị, hay tương ứng với việc các loại méo và nhiễu xảy ra trong quá trình ghép nối là nhỏ nhất.
* **Ghép nối các đơn vị âm**

Vấn đề lớn nhất đối với kỹ thuật tổng hợp ghép nối là ghép nối các đơn vị âm. Chất lượng của tiếng nói tổng hợp được nhờ kỹ thuật này phụ thuộc rất nhiều vào phương pháp hay thuật toán được sử dụng để ghép nối các đơn vị âm. Thông thường, việc ghép nối các đơn vị âm dễ dẫn đến hiện tượng không liên tục tại điểm ghép nối giữa các đơn vị âm (về cao độ, về phổ, về pha). Những sự không liên tục này gặp phải có thể là do có sự khác nhau về ngữ cảnh của các đơn vị âmhay do chính chất lượng không tốt của các đơn vị âm gây ra bởi quá trình thu âm, quá trình phân đoạn…

Đối với kỹ thuật tổng hợp ghép nối của tổng hợp tiếng nói, mô hình tổng hợp tiếng nói trong miền thời gian là mô hình cho phép thu được tiếng nói với chất lượng cao mà chi phí tính toán lại thấp hơn rất nhiều. Sự phát triển của tổng hợp tiếng nói trong miền thời gian gắn liền với sự phát triển của phương pháp PSOLA, một phương pháp cho phép điều khiển các tham số ngữ điệu của tiếng nói và ghép nối các đơn vị âm.

***Phương pháp TD-PSOLA***

PSOLA (Pitch Synchronous Overlap and Add) còn được gọi là kỹ thuật chồng đồng bộ cao độ tần số cơ bản. Nó thực hiện việc biến đổi trường độ và cao độ [âm thanh](http://vi.wikipedia.org/wiki/%C3%82m_thanh) hay là quá trình thay đổi cao độ [tần số âm cơ bản](http://vi.wikipedia.org/wiki/T%E1%BA%A7n_s%E1%BB%91_%C3%A2m_c%C6%A1_b%E1%BA%A3n) F0 và độ dài tiếng nói ở miền thời gian trực tiếp trên dạng sóng tiếng nói.Kỹ thuật này được sử dụng trong [tổng hợp giọng nói](http://vi.wikipedia.org/wiki/T%E1%BB%95ng_h%E1%BB%A3p_gi%E1%BB%8Dng_n%C3%B3i).

Kỹ thuật PSOLA được diễn ra qua ba bước là:

* Phân tích tín hiệu gốc thành các chuỗi âm ngắn theo các cửa sổ đồng bộ với các điểm đánh dấu pitch (còn gọi là epoch).
* Biến đổi các âm phân tích được ở trên thành các âm ngắn mới theo các biến đổi độ dài và tần số âm cơ bản.
* Tổng hợp lại các âm ngắn này theo phương pháp cộng chồng, đồng bộ lại tín hiệu.

Giải thuật PSOLA có nhiều phiên bản như FD-PSOLA, LP-PSOLA, TD-PSOLA. Trong các phiên bản này TD-PSOLA được sử dụng rộng rãi do chi phí tính toán thấp và chất lượng âm thanh tổng hợp đem lại.

TD-PSOLA là phiên bản trên miền thời gian của giải thuật PSOLA.Với PSOLA, tín hiệu tổng hợp được tạo nên bằng cách cộng xếp chồng (Overlap-Add) các đoạn tín hiệu thành phần. Giải thuật này cho phép thao tác trực tiếp với tín hiệu tiếng nói trên miền thời gian, thay đổi tần số cơ bản và độ dài của tín hiệu.

### 2.2.2 Chương trình Hoa Súng – MICA

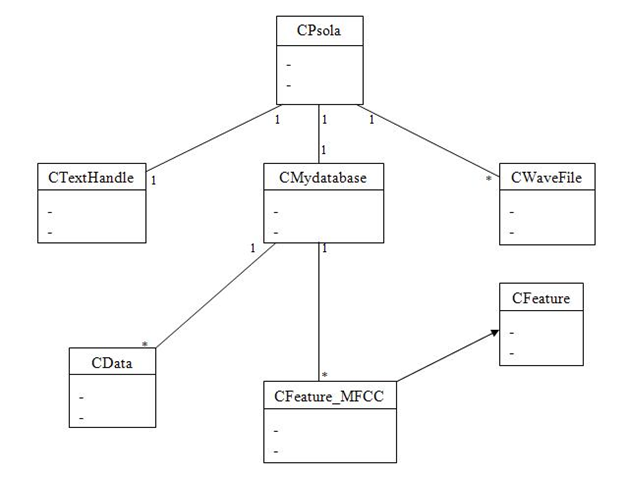
#### 2.2.2.1 Giới thiệu chương trình

Chương trình tổng hợp tiếng nói Hoa Súng là chương trình tổng hợp tiếng Việt có chất lượng khá cao, dễ hiểu và tương đối tự nhiên trên PC được phát triển bởi các chuyên gia nghiên cứu của Viện MICA. Chương trình được viết dựa trên nền tảng lập trình hướng đối tượng bằng công cụ lập trình Microsoft Visual C++ 6.0. Phiên bản đầu tiên của chương trình (phiên bản demo 1.0) được hoàn thành vào năm 2007 với sự tham gia của Trần Đỗ Đạt và Lê Xuân Hùng.

Chương trình cho phép điều khiển các tham số ngữ điệu (tần số cơ bản, trường độ) của tiếng nói cần tổng hợp theo ý muốn đồng thời còn cho phép người dùng có thể lựa chọn loại đơn vị âm để tổng hợp như: âm vị kép (Diphone), bán âm tiết (Half-Syllable), âm tiết (Base Syllable), âm đầu/vần (initial/final part) và dùng kết hợp các loại đơn vị âm để tổng hợp.

#### 2.2.2.2 Biểu đồ lớp của chương trình

Chương trình tổng hợp tiếng nói Hoa Súng bao gồm các lớp chính (các lớp liên quan đến các module tổng hợp tiếng nói) như trên biểu đồ sau đây:



Hình 2.3 Biểu đồ lớp của chương trình Hoa Súng

#### 2.2.2.3 Quá trình tổng hợp tiếng nói của chương trình

Chương trình tổng hợp tiếng nói HOA SÚNG sẽ có nhiệm vụ nhận văn bản đầu vào (gõ trực tiếp từ bàn phím hoặc từ file văn bản) và tạo ra file tiếng nói ở đầu ra là file .wav. Quá trình tổng hợp tiếng nói theo chương trình sẽ chia làm hai giai đoạn (block) lớn sau đây:

* Giai đoạn 1: đây là giai đoạn phân tích văn bản đầu vào, sử dụng các mô hình ngữ điệu và mô hình thanh điệu để lấy ra các thông tin cần thiết (các đơn vị âm cấu thành và các tham số ngữ điệu) về các âm tiết cần tổng hợp. Bước tiếp theo của giai đoạn này là tìm kiếm trong cơ sở dữ liệu các đơn vị âm tối ưu (sử dụng phương pháp tối ưu hóa hàm chi phí) để phục vụ cho việc ghép nối.
* Giai đoạn 2: đây là giai đoạn điều khiển các tham số ngữ điệu thích hợp theo thuật toán TD - PSOLA và ghép nối các đơn vị âm đã được chọn lựa ở giai đoạn 1. Các đơn vị âm được ghép nối để tạo ra từng âm tiết và từng âm tiết sẽ được ghi lần lượt vào file .wav, mỗi khi ghi một âm tiết mới vào thì làm trơn mức năng lượng giữa âm tiết đó với âm tiết trước nó.



Hình 2.4 Biểu đồ hoạt động tổng quát của chương trình Hoa Súng

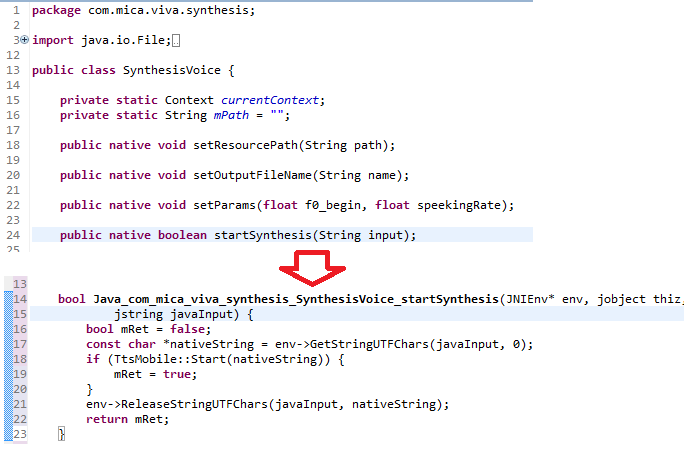
### 2.2.3 Chuyển đổi sang thư viện dùng cho Android

#### 2.2.3.1 Quá trình chuyển đổi

Việc triển khai trên hệ điều hành Android khá phức tạp, do các ứng dụng Android được viết bằng mã nguồn Java, được biên dịch thành bytecode rồi chuyển thành các file thực thi do Davik Virtual Machine nên ta không thể gọi trực tiếp tới mã nguồn C++ của thư viện tổng hợp Tiếng Việt ở trên được. Đế có thể sử dụng lại mã nguồn C++ trong chương trình Hoa Súng cho khối tổng hợp trên VIVA, chúng ta cần thực hiện các bước sau:

**Bước 1:** Chuẩn hóa các thư viện dùng trong mã nguồn C++ của Hoa Súng: thay thế thư viện xử lý chuỗi CString, CArray của thư viện MFC (Microsoft Foundation Class) chỉ hộ trợ trên nền Windows bằng thư viện mã nguồn mở Standard Lib C (STL C).

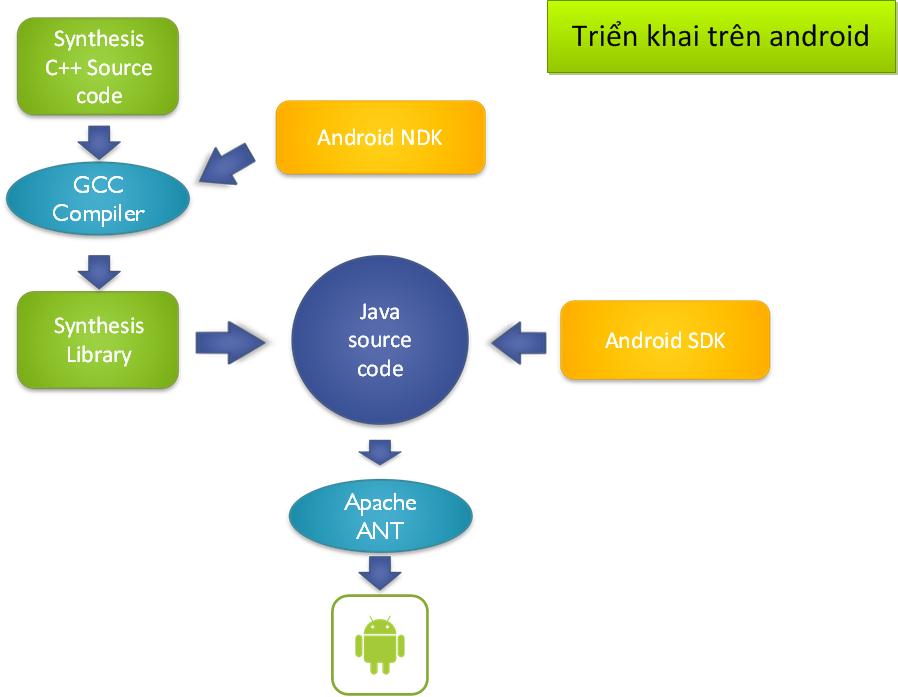
**Bước 2:** Khai báo các phương thức dạng native trong lớp SynthesisVoice và viết các phương thức thực thi tương ứng trong file main.cpp trong mã nguồn C++.



**Bước 3:** Biên dịch mã nguồn C++ thành thư viện liên kết động ( lib \*.so) rồi trong mã nguồn Java của chương trình , các thư viện này được nạp vào qua lời gọi **System.LoadLibrary().**

**E:\Study\study\DoAnTiengNoi\MyWorks\Resource\load-library.png**

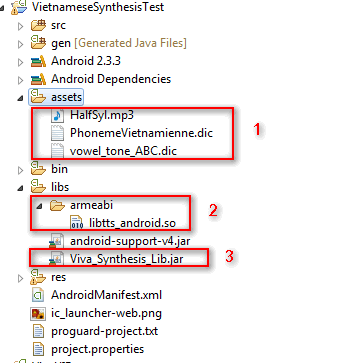
**Bước 4:** Đóng gói module tổng hợp tiếng com.mica.viva.synthesis thành thư viện \*.jar có thể dùng trong các project android.



Hình 2.5 Triển khai module tổng hợp tiếng trên Android

#### 2.2.3.1 Sử dụng trong Android project

**Bước 1:** thêm thư viện & dữ liệu.



Hình 2.6 Copy thư viện, dữ liệu module tổng hợp tiếng vào project Android

1. Copy các ***file dữ liệu*** vào thư mục ***assets***.
2. Copy thư viện ***libtts\_android.so*** vào thư mục ***libs/armeabi***.
3. Copy thư viện ***VIVA\_Synthesis\_Lib.jar*** vào thư mục ***libs***.

**Bước 2:** sử dụng module tổng hợp tiếng.

Import package:

**import** com.mica.viva.synthesis.SynthesisVoice;

Khởi tạo đối tượng, gọi hàm để cấu hình (tự động) các tham số module tổng hợp:

SynthesisVoice syn = **new** SynthesisVoice();

syn.prepareToSynthesis(getApplicationContext());

Tổng hợp & phát câu nói:

syn.readText(text);

#### 2.2.3.3 API

Bảng 2.1 API module tổng hợp tiếng trên Android

|  |  |
| --- | --- |
| **package**  **public** **class** | com.mica.viva.synthesis  SynthesisVoice |
| **Public methods** | **Mô tả** |
| **void** setResourcePath(String path); | Thiết đặt đường dẫn tới các file dữ liệu.  Example:  "/data/data/" + getPackageName() + "/" |
| **void** setOutputFileName(String name ); | Thiết đặt đường dẫn + tên file tiếng nói được tổng hợp.  Example:  "/data/data/" + getPackageName() + "/result.wav" |
| **void** setParams(**float** f0\_begin, **float** speekingRate); | Thiết đặt tham số tổng hợp tiếng nói:  f0\_begin: cao độ (100-350).  speekingRate: trường độ (0.5-2). |
| **boolean** startSynthesis(String input); | Tổng hợp tiếng nói cho câu đưa vào, kết quả tổng hợp lưu tại file thiết đặt trước bởi hàm setOutputFileName  Return: true – tổng hợp thành công  false – tổng hợp không thành công |
| **void** prepareToSynthesis(Context mContext) | Tự động thiết đặt các tham số nêu trên:  Resource path:  "/data/data/" + getPackageName() + "/"  Output file:  "/data/data/" + getPackageName() + "/result.wav"  f0 begin: 210  speaking rate: 1 |
| **void** readText(String message) | Tổng hợp câu nói, đồng thời phát câu nói tổng hợp được. |

## 2.3. Module nhận dạng tiếng Việt.

Module nhận dạng tiếng Việt được xây dựng trên nền tảng Sphinx4 đặt trên server (server MICA) phục vụ cho nhiều client kết nối tới qua internet.

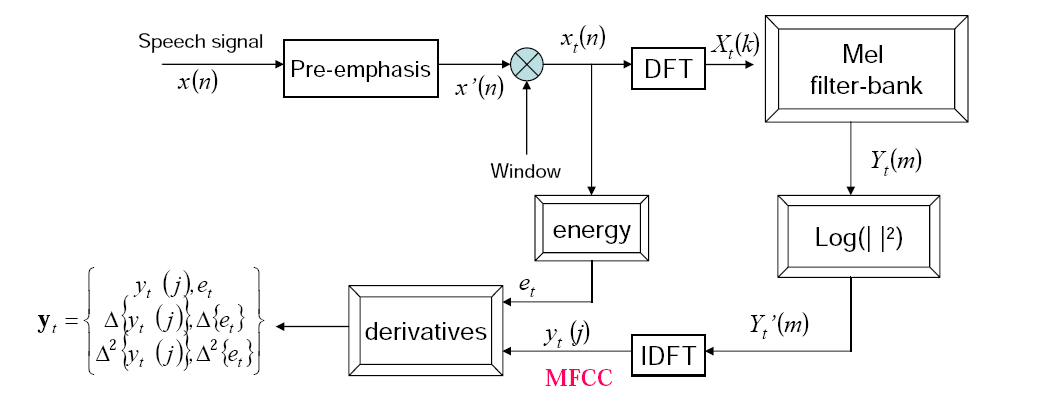
Sphinx4 là công cụ nhận dạng tiếng nói liên tục với bộ từ vựng lớn, không phụ thuộc vào người nói do trường đại học Carnegie Mellon University phát triển. Nó là tập hợp của các công cụ miễn phí và mã nguồn mở cho phép nhà nghiên cứu và phát triển xây dựng các hệ thống nhận dạng cho riêng mình. Sphinx4 là công cụ viết bằng ngôn ngữ lập trình Java, được phát triển trên nền tảng Sphinx2, Sphinx3. Nó cung cấp mô hình linh hoạt hơn trong việc phát triển hệ thống nhận dạng tiếng nói.

Mô hình được chọn để xây dựng hệ thống nhận dạng là mô hình Markov ẩn liên tục (Continous Density HMM). Phương pháp trích chọn đặc trưng được sử dụng là MFCC (Mel-scale Frequency Cepstral Coefficients).

### 2.3.1. Cơ sở lý thuyết.

#### 2.3.1.1 Phương pháp trích chọn đặc trưng MFCC

Phương pháp trích chọn đặc trưng MFCC có quy trình được thể hiện ở mô hình sau:



Hình 2.7 Sơ đồ thuật toán MFCC

* Speech signal: Tín hiệu âm thanh đầu vào, quá trình thu âm lời nói thực chất là quá trình biến đổi một tín hiệu liên tục thành một tín hiệu rời rạc (qua microphone).
* Pre - Emphasis:tăng cường năng lượng của các âm thanh ở tần số cao.
* Window: phân khung và cửa sổ hóa tín hiệu.
* DFT (Discrete Fourier Transform): chuyển đổi tín hiệu thời gian thành tín hiệu tần số.
* Mel filter-bank: dùng những cửa sổ lọc – filterbank, được sắp xếp một cách đồng đều để tính toán mel – spectrum.
* Logarit bình phương biên độ: tính logarit của bình phương biên độ tín hiệu đầu ra sau khi qua bộ lọc tần số Mel.
* IDFT: chuyển đổi logarit của mel spectrum về trục thời gian, kết quả được gọi là "mel- frequency cepstrum coefficients (MFCC)".
* Energy: tính toán hệ số đặc trưng của năng lượng : 
* Derivatives: tính toán hệ số để mô tả sự thay đổi của các hệ số đặc trưng theo thời gian. Các hệ số đó gọi là đặc trưng delta và đặc trưng double-delta

Một vectơ đặc trưng MFCC điển hình có định dạng như sau :

* Độ dài mỗi khung (frame) : 25 ms
* Khoảng cách các frame : 10 ms
* Hệ số pre-emphasis : 0.97
* Số lượng các hệ số đặc trưng :
  + 12 MFCC ( hệ số thang tần số Mel)
  + 12 hệ số delta MFCC
  + 12 hệ số delta bình phương MFCC
  + 1 hệ số đặc trưng năng lượng
  + 1 hệ số delta năng lượng
  + 1 hệ số delta bình phương của năng lượng

Như vậy, tổng cộng có 39 hệ số đặc trưng trong MFCC.

#### 2.3.1.2 Mô hình Markov ẩn liên tục

* **Mô hình Markov ẩn**

Mô hình Markov ẩn là mô hình thống kê trong đó hệ thống được mô hình hóa được cho là một quá trình Markov với các tham số không biết trước và nhiệm vụ là xác định các tham số ẩn từ các tham số quan sát được, dựa trên sự thừa nhận này. Các tham số của mô hình được rút ra sau đó có thể sử dụng để thực hiện các phân tích kế tiếp.

Trong một mô hình Markov điển hình, trạng thái được quan sát trực tiếp bởi người quan sát, và vì vậy các xác suất chuyển tiếp trạng thái là các tham số duy nhất. Mô hình Markov ẩn thêm vào các đầu ra: mỗi trạng thái có xác suất phân bổ trên các biểu hiện đầu ra có thể. Vì vậy, nhìn vào dãy của các biểu hiện được sinh ra bởi HMM không trực tiếp chỉ ra dãy các trạng thái.

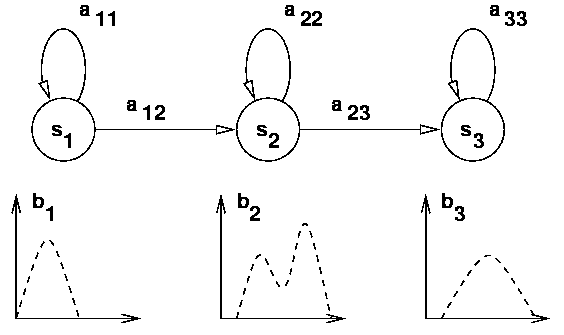
Mô hình Markov ẩn đặc trưng bởi các tham số sau :

* O = {o1, o2,..., oM} : Một bảng kí tự quan sát ở đầu ra. Các kí tự này đại diện cho đầu ra các đại lượng vật lí của hệ thống.
* Ω {1,2,..., N} : Tập hợp các trạng thái để biểu diễn một không gian trạng thái. st biểu diễn trạng thái tại thời điểm t.
* Aij = {aij} : Ma trận xác suất chuyển trạng thái. aij  là xác suất chuyển trạng thái từ i sang j.
* B = {bi(k)} : Ma trận xác suất đầu ra. bi(k) là xác suất đầu ra của kí tự ok khi trạng thái i được gia nhập.
* π = {πi} là phân bố trạng thái khởi tạo. πi = P (s0 = i)
* Chuỗi vecto quan sát X (X1, X2,..., XT).

Áp dụng mô hình HMM vào nhận dạng tiếng nói, ta phải giải quyết 3 vấn đề chính sau :

* Vấn đề đánh giá : Cung cấp  cho mô hình các tham số ( chuỗi quan sát O = (o1,o2, ..., oT) và một mô hình HMM (Φ = (A, B,π)) , tính xác suất của dãy đầu ra cụ thể (P (X | Φ). Giải bằng thuật toán tiến trước.
* Vấn đề giải mã : Cung cấp cho mô hình các tham số Φ và X , tìm dãy các trạng thái (ẩn) S (s0,s1,...,sT) có khả năng lớn nhất mà có thể sinh ra dãy đầu ra đã cung cấp. Giải bằng thuật toán Viterbi.
* Vấn đề học : Cung cấp dãy đầu ra X và mô hình Φ , tìm tập hợp có khả năng nhất của chuyển tiếp trạng thái và các xác suất đầu ra. Giải bằng thuật toán Baum-Welch.
* **Mô hình Markov ẩn liên tục (Continuous density HMM)**

Để đảm bảo cấu trúc ban đầu của tín hiệu nguyên thủy, người ta dùng mô hình Markov liên tục :



Hình 2.8 Mô hình Markov ẩn liên tục

Mô hình HMM liên tục là một mô hình tốt cho lí thuyết HMM. Do khối lượng tính toán khá lớn nên đặt hệ thống nhận dạng trên 1 máy chủ (server) là phù hợp. Mỗi trạng thái trong HMM liên tục cần có M vector riêng đặc trưng. Mỗi vector có thêm 1 ma trận hội tụ tương ứng với nó. Vector có số chiều càng lớn thì kích thước ma trận hội tụ càng lớn, tính toán càng phức tạp.

### 2.3.2. Triển khai module nhận dạng trong VIVA.

Trong phạm vi đề tài, nhóm tập trung xây dựng cơ sở dữ liệu xoay quanh một số chủ đề chính : tin nhắn (soạn tin nhắn bằng tiếng nói, ra lệnh gửi, hủy... ), truy vấn thông tin (thời tiết, giá vàng,ngoại tệ, chứng khoán...) trên Internet. Mô hình nhận dạng dựa trên bộ từ vựng nhỏ (từ điển khoảng 250 từ), các kịch bản huấn luyện được xây dựng từ trước tập trung vào các chủ đề trên.

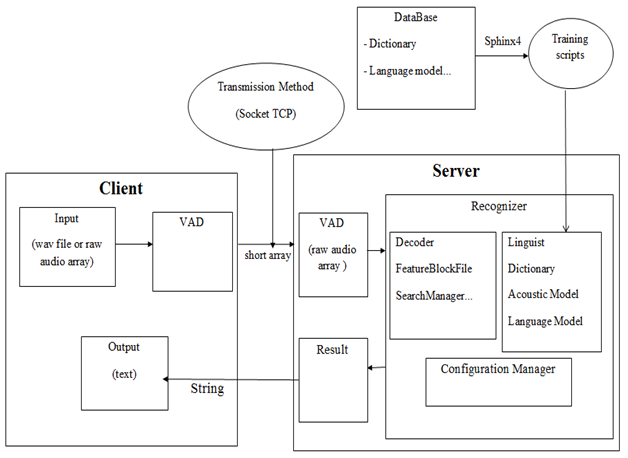
#### 2.3.2.1 Xây dựng mô hình nhận dạng trên Sphinx4

Việc xây dựng mô hình nhận dạng trên Sphinx4 được thực hiện qua 2 bước:

* Chuẩn bị cơ sở dữ liệu
* Xây dựng thư viện cơ sở dữ liệu và các mô hình

#### 2.3.2.2 Xây dựng chương trình nhận dạng

* **Sơ đồ khối chương trình**



Hình 2.9 Sơ đồ khối chương trình nhận dạng

* **Cách thức hoạt động**
* Môi trường truyền là mạng: Internet (mạng dây, 3G), LAN ( mạng dây, wifi )
* Kiểu kết nối: Multi Client – Server
* Client gửi cho server một mảng kiểu short là tín hiệu âm thanh đã được lấy mẫu
* Server nhận dạng và trả lại kết quả dạng text
* Phương thức truyền : Socket TCP

Cách thức hoạt động của cả hệ thống Multi client – server nhận dạng tiếng nói được mô tả như sau :

Sau khi thu âm ở client, client sẽ tự động ngắt thu âm và mở kết nối socket với server, số mẫu tiếng nói ban đầu được và được lưu vào một mảng short và được gửi lên server.

Server khi chạy luôn mở socket để lắng nghe và kết nối khi có một client muốn kết nối. Với mỗi một client đến server tạo một thread riêng để đáp ứng với client đó.

Quá trình nhận dạng qua một số bước chính: sau khi qua bộ VAD để bỏ các khoảng lặng, các file âm thanh sẽ được trích chọn đặc trưng MFCC để tạo ra các hệ số đặc trưng như đã mô tả ở các phần trước, rồi đưa đến bộ giải mã (decoder). Các hệ số đặc trưng đưa vào bộ giải mã để ánh xạ đến mô hình ngôn ngữ (file dict và DMP) và mô hình âm học (folder an4.ci\_cont) đã thu được sau quá trình huấn luyện. Bộ giải mã sẽ hoạt động và đưa ra kết quả nhận dạng.

## 2.4. Module hiểu ngữ nghĩa.

Trong giai đoạn đầu của dự án, xây dựng chương trình VIVA phiên bản demo, khối hiểu ngữ nghĩa được xây dựng đơn giản để hiểu câu lệnh người dùng yêu cầu, và tập trung vào các chức năng: tin nhắn, truy vấn thông tin. Khối hiểu được xây dựng trên mô hình một automat hữu hạn, đơn định, câu đầu vào được tách từ, các từ được gán nhãn theo bộ nhãn quy định trước, danh sách từ đã gán nhãn được đưa vào automat, nếu được đoán nhận, automat sẽ trả về mã khối chức năng, mã chức năng và các tham số.

Phần cơ sở lý thuyết sau đây trình bày tóm lược về phương pháp tách từ và khái niệm automat, automat hữu hạn đơn định sử dụng trong khối hiểu.

### 2.4.1 Cơ sở lý thuyết.

#### 2.4.1.1 Tách từ tiếng Việt

Tách từ là một quá trình xử lý nhằm mục đích xác định ranh giới của các từ trong câu văn, cũng có thể hiểu đơn giản rằng tách từ là quá trình xác định các từ đơn, từ ghép… có trong câu. Đối với xử lý ngôn ngữ tự nhiên, để có thể xác định cấu trúc ngữ pháp của câu, xác định từ loại của một từ trong câu, yêu cầu nhất thiết đặt ra là phải xác định được đâu là từ trong câu. Vấn đề này tưởng chừng đơn giản với con người nhưng đối với máy tính, đây là bài toán rất khó giải quyết.

Chính vì lý do đó tách từ được xem là bước xử lý quan trọng đối với các hệ thống Xử Lý Ngôn Ngữ Tự Nhiên, đặc biệt là đối với các ngôn ngữ thuộc vùng Đông Á theo loại hình ngôn ngữ đơn lập. Ví dụ: tiếng Trung Quốc, tiếng Nhật, tiếng Thái, và tiếng Việt. Với các ngôn ngữ thuộc loại hình này, ranh giới từ không chỉ đơn giản là những khoảng trắng như trong các ngôn ngữ thuộc loại hình hòa kết như tiếng Anh…, mà có sự liên hệ chặt chẽ giữa các tiếng với nhau, một từ có thể cấu tạo bởi một hoặc nhiều tiếng. Vì vậy đối với các ngôn ngữ thuộc vùng Đông Á, vấn đề của bài toán tách từ là khử được sự nhập nhằng trong ranh giới từ.  
3.2.1 Các hướng tiếp cận giải quyết

Có 3 phương pháp tiếp cận chính trong bài toán tách từ :

* Tiếp cận dựa vào từ điển cố định.
* Tiếp cận dựa vào thống kê thuần túy
* Tiếp cận dựa trên cả hai phương pháp trên.

Các phương pháp được sử dụng:

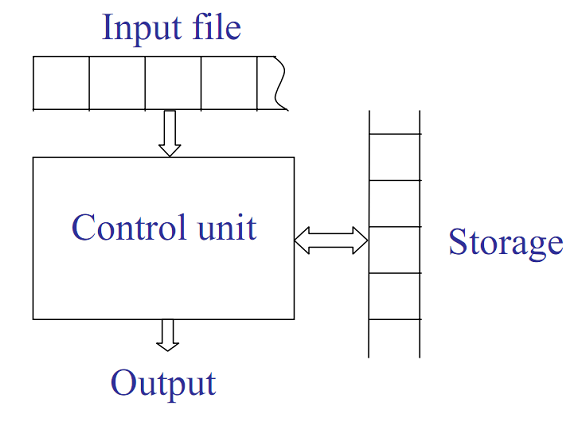
* So khớp từ dài nhất (Longest Matching)
* So khớp cực đại (Maximum Matching)
* Mô hình Markov ẩn (Hidden Markov Models- HMM)
* Học dựa trên sự cải biến (Transformation-based Learning – TBL)
* Chuyển đổi trạng thái trọng số hữu hạn (Weighted Finite State Transducer – WFST)
* Độ hỗn loạn cực đại (Maximum Entropy – ME)
* Máy học sử dụng vectơ hỗ trợ (Support Vector Machines)
* Trường xác xuất có điều kiện (CRFs)

Tài nguyên hỗ trợ:

* Từ điển tiếng Việt
* Ngữ liệu tiếng Việt được tách từ hỗ trợ quá trình huấn luyện

#### 2.4.1.1 Automat

Automat là máy dịch tự động. Đây là một mô hình trừu tượng của một automat bao gồm các thành phần sau :



Hình 2.10 Mô hình của một automat

* Đầu vào (input file) : là nơi mà các chuỗi nhập (input string) được ghi lên và được automat đọc nhưng không thay đổi được nội dung. Nó được chia thành các ô, mỗi ô giữ một ký hiệu.
* Cơ cấu nhập(input mechanism): là bộ phận có thể đọc input file từ trái sang phải, một ký tự tại một thời điểm. Nó cũng có thể dò tìm được điểm kết thúc của chuỗi nhập
* Bộ nhớ tạm(temporary storage) : là thiết bị gồm một số không giới hạn ô nhớ(cell), mỗi ô có thể giữ 1 ký hiệu từ một bảng chữ cái (không cần giống bảng chữ cái nhập). Automat có thể đọc và thay đổi nội dung của các ô nhớ này.
* Đơn vị điều kiển (Control Unit): mỗi automat có một đơn vị điều khiển, cái mà có thể ở trong một trạng thái bất kỳ trong một số hữu hạn các trạng thái nội, và có thể chuyển đổi trạng thái trong một kiểu được định nghĩa sẵn có nào đó.
* **Một số khái niệm:**
* Trạng thái nội: là một trạng thái của đơn vị điều khiển mà nó có thể đạt được
* Trạng thái kế: Là một trạng thái nội của đơn vị điều khiển mà nó sẽ đạt tới ở thời điểm kế tiếp
* Hàm chuyển trạng thái: là hàm đưa ra trạng thái kế của automat dựa trên trạng thái hiện hang, kí hiệu hiện thành được quét và thông tin hiện hành trong bộ nhớ tạm.
* **Hoạt động của automat:**
* Tại một thời điểm bất kỳ đã cho, đơn vị điều khiển đang ở trong một trạng thái nội (internal state) nào đó, và cơ cấu nhập là đang quét (scanning) một ký hiệu củ thể nào đó trên Input file.
* Trạng thái nội của đơn vị điều khiển tại thời điểm kế tiếp được xác định bởi trạng thái kế (next state) hay bởi hàm chuyển trạng thái.
* Trong suốt quá trình chuyển trạng thái từ khoảng thời gian này tới khoảng thời gian kế, kết quả có thể được sinh ra và lưu trong bộ nhớ lưu trữ có thể thay đổi.

#### 2.4.1.2 Automat hữu hạn đơn định – DFA

Otomat hữu hạn đơn định (Deterministic Finite Automata) gồm một tập hữu hạn các trạng thái và một tập các phép chuyển trạng thái tới trạng thái khác trên các ký hiệu nhập (input symbols) được chọn từ một bộ chữ cái ∑ nào đó. Mỗi ký hiệu nhập có đúng một phép chuyển khỏi mỗi trạng thái. Nó cũng có thể chuyển về chính nó. Có trạng thái bắt đầu và một số trạng thái được thiết kế như trạng thái kết thúc hay trạng thái chấp nhận.

Otomat hữu hạn đơn định gồm 5 thành phần:

* Q : là tập hữu hạn các trạng thái
* ∑ : là bộ chữ cái nhập hữu hạn
* δ : Là hàm chuyển ánh xạ từ Q x ∑ 🡪 Q, tức là δ(q,a) là một trạng thái được cho phép bởi phép chuyển từ trạng thái q trên ký hiệu nhập a
* q0 là trạng thái bắt đầu
* F là tập các trạng thái kết thúc

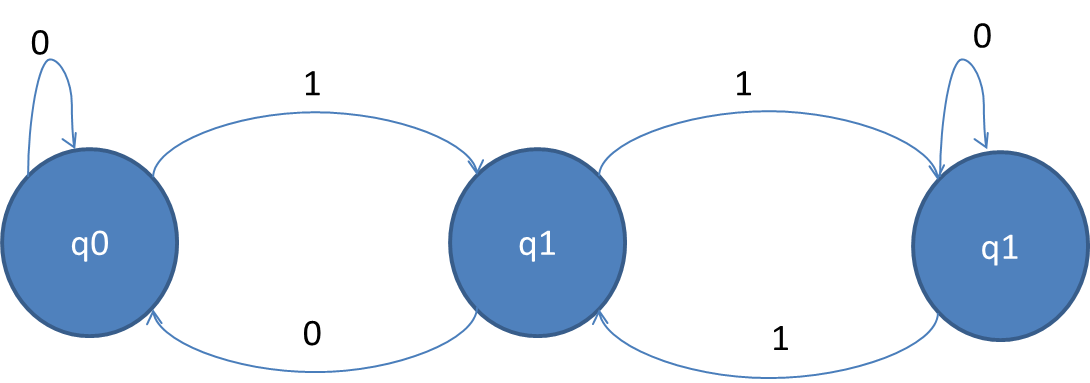
Hoạt động của một DFA:

* Tại thời điểm khởi đầu, nó được giả thiết ở trong **trạng thái khởi đầu q0** với cơ cấu nhập đang ở ký hiệu đầu tiên bên trái của chuỗi nhập
* Mỗi lần di chuyển, cơ cấu nhập tiến về phía phải 1 ký hiệu và lấy ra
* Khi gặp ký hiệu kết thúc, chuỗi được chấp nhận nếu automat đang ở vào 1 trong **các trạng thái chấp nhận được** của nó. Ngược lại thì chuỗi bị từ chối.

Để biểu diễn trực quan cho DFA ta có thể dùng đồ thị chuyển trạng thái:

Ví dụ đồ thị chuyển trạng thái của Một DFA : M = (Q, ∑, δ, q0, F)

Q = {q0,q1,q2}, ∑ = {0,1}, F{q1}



Hình 2.11 Đồ thị chuyển trạng thái của một automat hữu hạn đơn định

### 2.4.2 Triển khai module hiểu ngữ nghĩa trong VIVA.

* Bước 1 : Tách từ

Sử dụng công cụ có sẵn là bộ tách từ  *vnTokenizer 4.1.1c (04/08/2010)* của dự án VLSP.[6]

* Bước 2 : Gán nhãn keyword cho từ

Xác định loại liên quan của từ được tách ra.

Ví dụ: nhắn, nhắn tin 🡪 tomessage ; thời tiết, trời 🡪toweather.

Đây chính là xác định các “ký tự nhập” của automat.

* Bước 3 : Thực hiện dịch automat

Kết quả chạy theo là các tham số Function xác định được.

* Bước 4: Xuất kết quả ra, trả về hàm điều khiển ở hệ thống.

Trong hệ thống hiện tại, gửi về client để xử lý.

## 2.5. Kiến trúc VIVA trên android.

Ứng dụng VIVA được thiết kế và xây dựng trên android, một hệ điều hành mã nguồn mở mạnh mẽ dành cho thiết bị di động. Hệ điều hành android ngày càng phát triển mạnh mẽ và số lượng thiết bị cài đặt hệ điều hành này cũng không ngừng tăng cao.

### 2.5.1. Giới thiệu kiến trúc Android.

Hệ điều hành Android có 4 tầng từ dưới lên trên là tầng hạt nhân Linux(phiên bản 2.6), tầng Libraries & Android runtime, Tầng Application Framework và trên cùng là tầng Application.



Hình 2.12 Kiến trúc tổng thể Android

* **Tầng hạt nhân Linux ( Linux Kernel Layer)**

Hệ điều hành Android được phát triển dựa trên hạt nhân Linux, cụ thể là hạt nhân linux phiên bản 2.6, điều đó được thể hiện ở lớp dưới cùng này. Tất cả mọi hoạt động của điện thoại muốn thi hành được thì đều được thực hiện ở mức cấp thấp ở lớp này bao gồm quản lý bộ nhớ (memory management), giao tiếp với phần cứng ( Driver model ), thực hiện bảo mật ( security),quản lý tiến trình ( process).

* **TầngLibraries**

Android cung cấp một tập các thư viện C/C++ được sử dụng bởi các thành phần khác nhau trong một hệ thống. Bao gồm một số thư viện lõi: thư viện C hệ thống,thư viện media, Surface Manager, LibWebCore,SGL (Skia Graphics Library) cơ chế đồ họa 2D cơ sở, thư viện 3D: OpenGL ES 1.0 APIs, FreeType, SQLite.

* **Tầng Application Framework.**

Đây là nơi chứa các dịch vụ và hệ thống quản lí ứng dụng bao gồm:

Activity Manager,Window Manager, Content Providers, View Systemm, Package, Telephony Manager, Resource Manager, Notification Manager.

* **Tầng Applications**

Đây là các ứng dụng tương tác trực tiếp với người dùng bao gồm các ứng dựng lõi và những ứng dụng của bên thứ ba. Ngoài việc cung cấp đầy đủ các ứng dụng cơ bản của một chiếc điện thoại thông thường như: tạo cuộc gọi, nhắn tin... Android còn có những dịch vụ rất hữu ích khác: trình duyệt web, google maps, các ứng dụng media, camera, games

Chương trình VIVA được xây dựng trên tầng Applications.

### 2.5.2. Kiến trúc ứng dụng VIVA.

#### 2.5.2.1. Biểu đồ lớp

VIVA được phân tích và thiết kế hướng đối tượng. Kiến trúc VIVA gồm các package:

* *com.mica.viva:* package của chương trình
* *com.mica.viva.controller*: khối điều khiển chính.
* *com.mica.via.controller.sms*: khối chức năng tin nhắn.
* *com.mica.viva.controller.queryinformation*: khối chức năng truy vấn thông tin.
* *com.mica.viva.inputting*: khối thu âm, VAD
* *com.mica.viva.entity*: khối thực thể, chứa các entity class chứa thông tin module chức năng, tham số của chức năng…
* *com.mica.viva.resource*: đọc các file data, quản lý các entity.
* *com.mica.viva.utility* : chứa các lớp tiện ích.

Biểu đồ lớp của chương trình:





Hình 2.13 Biểu đồ lớp của chương trình VIVA

#### 2.5.2.2. Mô tả lớp

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller | |
| Class | Đặc tả | |
| BaseModuleController | Lớp cha | android.app.Activity |
| 1. Là lớp cơ sở của các lớp thực thi chức năng (gửi tin, đọc tin nhắn, trả lời thời tiết, giá vàng…) 2. Có thuộc tính **protected** Thread process là luồng thực thi chức năng. Các lớp con sẽ khai báo kịch bản, tiến trình thực hiện trong hàm **public** **void** run() của process | |
| UIController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển giao diện chính của ứng dụng. * Cung cấp các phương thức public để thao tác với giao diện chính: * **void** displayRequestText(String requestText) hiển thị câu nói của người dùng nhập vào. * **void** displayResponseText(String responseText) hiển thị câu trả lời của VIVA * **void** displayHtmlResult(String htmlContent) hiển thị thông tin dạng html | |
| InputController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc nhập câu nói. * Các phương thức public để thực hiện quá trình nhập câu nói: * **void** startInputSentence(Activity caller) bắt đầu nhập vào câu lệnh * **void** startInputParameter(Activity caller) bắt đầu nhập vào tham số (thí dụ: tham số tên ngươi nhận trong chức năng gửi tin nhắn) * **void** startInputChoise(Activity caller) bắt đầu nhập vào lựa chọn có hay không? | |
| ResponseController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc trả lời bằng tiếng nói của VIVA. * Cung cấp các phương thức public thực hiện việc đọc và hiển thị câu trả lời: * **void** responseVoiceMessage(String message) đọc câu trả lời. * **void** responseMessage(String message) đọc câu trả lời đồng thời hiện câu trả lời ra màn hình. * **void** responseRandomMessage(String type) tìm câu ngẫu nhiên theo kiểu định trước trong cơ sở dữ liệu và trả lời người dùng. | |
| FunctionController | Lớp cha | java.lang.Object |
| * Là lớp tiếp nhận dữ liệu trả về từ khối hiểu và kích hoạt chức năng tương ứng. * Các phương thức public quan trọng: * Function getFunctionFromData(String st) nhận vào đoạn text có định dạng trả về từ khối hiểu và đưa ra chức năng tương ứng cùng các tham số nếu có. * **void** startFunction(Context context, Function function) Kích hoạt một chức năng. * **void** startFunctionForResult(Activity activity,Function function) kích hoạt một chức năng và trả về kết quả sau khi thực hiện xong. | |

Các package, lớp còn lại được mô tả chi tiết trong phụ lục 1.

# KHỐI CHỨC NĂNG TIN NHẮN

## 3.1 Chức năng gửi tin nhắn.

### 3.1.1 Mô tả.

Chức năng gửi tin nhắn cho phép người dùng gửi tin nhắn với hai tham số: tên hoặc số điện thoại người nhận và nội dung tin nhắn.

Nếu trong câu lệnh người dùng chưa có hai thông tin này, VIVA sẽ yêu cầu người dùng nhập vào thông tin còn thiếu.

Trước khi gửi tin nhắn, VIVA yêu cầu người dùng xác nhận có muốn gửi tin nhắn hay không, nếu xác nhận có, VIVA thực hiện gửi tin nhắn và thông báo khi gửi thành công.

Khi người yêu cầu gửi tin nhắn cho tên một người trong danh bạ, VIVA sẽ tìm kiếm trong danh bạ theo tên này, các trường hợp xảy ra:

* Không có tên người nhận trong danh bạ: VIVA thông báo và yêu cầu nhập lại.
* Có 1 liên hệ tương ứng: gửi tin nhắn cho người này.
* Có nhiều hơn 1 liên hệ tương ứng: VIVA yêu cầu người dùng chọn theo số thứ tự “một” / “hai” / “ba”…

### 3.1.2 Sơ đồ hoạt động.



Hình 3.1 Sơ đồ hoạt động của chức năng gửi tin nhắn

### 3.1.3 Biểu đồ tuần tự.





Hình 3.2 Biểu đồ tuần tự của chức năng gửi tin nhắn

## 3.2 Chức năng đọc tin nhắn đến.

### 3.2.1 Mô tả.

Android là hệ điều hành mở cho phép các ứng dụng bắt các bản tin Broadcast các sự kiện trong đó có sự kiện có tin nhắn đến. Lớp ReceivingSms trong module tin nhắn của VIVA làm nhiệm vụ bắt bản tin Broadcast này và kích hoạt chức năng đọc tin nhắn mỗi khi có tin nhắn đến.

Chức năng đọc tin nhắn bao gồm 3 thao tác sau:

* Thông báo tin nhắn mới: VIVA thông báo cho người dùng “bạn có tin nhắn từ số điện thoại…”. Nếu số điện thoại người gửi được lưu trong danh bạ, VIVA sẽ tìm trong danh bạ để lấy ra tên liên hệ tương ứng và thông báo cho người dùng.
* Confirm và đọc nội dung tin nhắn: VIVA hỏi người dùng có muốn nghe nội dung tin nhắn không, nếu có, VIVA thực hiện đọc nội dung tin nhắn.
* Confirm trả lời tin nhắn: VIVA hỏi người dùng có muốn trả lời tin nhắn không, nếu có, VIVA chuyển sang chức năng gửi tin nhắn đã trình bày trên.

### 3.2.2 Sở đồ hoạt động.



Hình 3.3 Sơ đồ hoạt động của chức năng đọc tin nhắn

### 3.2.3 Biểu đồ tuần tự.



Hình 3.4 Biểu đồ tuần tự của chức năng đọc tin nhắn

# THÊM DẤU CHO VĂN BẢN TIẾNG VIỆT KHÔNG DẤU

## 4.1. Lý do cần giải quyết bài toán.

Trong chức năng đọc tin nhắn đến của VIVA, có một vấn đề là nội dung tin nhắn thường được người dùng viết ở dạng không dấu để tăng tốc độ soạn thảo cũng như tăng được nội dung tin nhắn (thông thường các nhà mạng Việt Nam giới hạn độ dài tin nhắn 160 ký tự với tiếng Việt không dấu, 70 ký tự với tiếng Việt có dấu). Việc nhắn tin bằng tiếng Việt không dấu rất phổ biến, hầu hết những tin nhắn được gửi đều ở dạng không dấu hoặc có dấu nhưng không đầy đủ, ngay cả tới những tin nhắn thông báo của các nhà mạng. Nội dung bằng tiếng Việt không khi đưa vào khối tổng hợp tiếng nói của VIVA sẽ không tổng hợp được hoặc nội dung khó hiểu, sai lệch, vì thế cần thiết phải chuẩn hóa, tự động thêm dấu cho các nội dung này trước khi đưa vào khối tổng hợp.

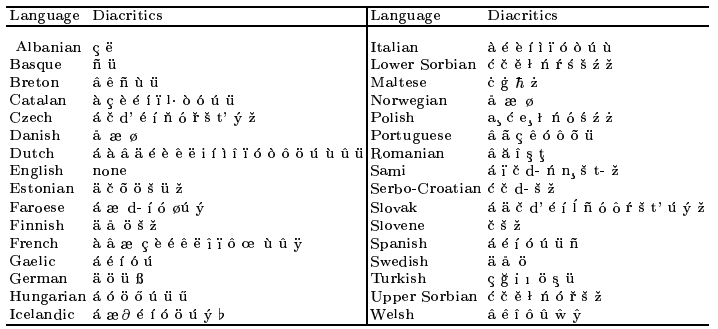
Không chỉ gói hẹp trong phạm vi nội dung tin nhắn SMS, văn bản tiếng Việt không dấu cũng rất phổ biến trong các nội dung trên internet đặc biệt là các nội dung chat, email, comment, hay các ghi chú cá nhân… Việc thêm dấu cho văn bản không dấu hoặc thiếu dấu cũng rất quan trọng trong việc chuẩn hóa văn bản text làm đầu vào hoặc làm dữ liệu huấn luyện cho các hệ thống như hệ thống tổng hợp tiếng nói, hệ thống dịch tự động, máy tìm kiếm, các hệ thống xử lý ngôn ngữ tự nhiên tiếng Việt, cũng như có thể áp dụng vào rất nhiều các chương trình như kiểm tra chính tả, tự động sửa lỗi chính tả cho văn bản, …

Chính vì thế, giải quyết bài toán tự động thêm dấu cho văn bản tiếng Việt với độ chính xác cao là rất quan trọng.

## 4.2. Hiện trạng bài toán.

Bài toán thêm dấu cho văn bản được đặt ra từ khá sớm với rất nhiều ngôn ngữ trên thế giới. Chỉ tính riêng trong khối châu Âu đã có tới 36 ngôn ngữ sử dụng bản chữ cái lating và có dấu giống như tiếng Việt.

Bảng 4.1 Dấu trong các ngôn ngữ viết trên bảng chữ Latin trong khối Châu Âu[10]



Thêm dấu cho văn bản tiếng Việt tự động thực sự là một công việc không hề dễ dàng, khó khăn lớn nhất là có quá nhiều sự lựa chọn khi chuyển từ từ không dấu thành từ có dấu với ý nghĩa khác nhau tạo nên sự nhập nhằng, một từ chỉ được thêm dấu chính xác khi được xem xét trong ngữ cảnh cụ thể, quan hệ ngữ nghĩa với các từ khác trong câu.

Bảng 4.2 Thống kê tỉ tệ từ có dấu và tỉ lệ nhập nhằng của từ không dấu trong một số ngôn ngữ

| **Ngôn ngữ** | **Tỉ lệ từ có dấu** | **Tỉ lệ từ không dấu tạo nên nhập nhằng** |
| --- | --- | --- |
| French [11] | 15% | 50% |
| Romanian [9] | 35% | 25% |
| Vietnamese[12] | 95% | 80% |

Bảng thống kê trên cho thấy, dấu xuất hiện với một tần suất rất lớn trong tiếng Việt, đồng thời tỉ lệ việc từ không dấu tạo nên sự nhập nhằng về nghĩa là rất cao, vì thế việc thêm dấu cho văn bản tiếng Việt là khó khăn hơn nhiều so với các ngôn ngữ khác như tiếng Pháp, tiếng Romani. Một thí dụ đơn giản như, từ “toi” không dấu trong tiếng Việt có thể được thêm dấu để trở thành : “tôi” (chỉ chính mình), “tỏi” (trong củ tỏi), “tồi” (trong tồi tệ), “tội” (trong tội ác) hoặc cũng có thể là “tối” (trong buổi tối)…Sự nhập nhằng khiến cho đôi khi chính con người cũng khó hiểu, hiểu sai với các câu tiếng Việt không dấu.

### 4.2.1. Các hướng tiếp cận đang có.

Bài toán thêm dấu đã và đang được nghiên cứu và giải quyết bởi rất nhiều nhà nghiên cứu trên nhiều ngôn ngữ trên thế giới, có nhiều cách làm khác nhau dựa trên hai hướng tiếp cận chính: hướng tiếp cận dựa vào từ (word-based)và hướng tiếp cận dựa vào âm tiết (character-based).

* Hướng tiếp cận dựa vào từ:
* Hoạt động dựa vào việc phân tích từ vựng và mô hình ngôn ngữ, các phương pháp theo hướng tiếp cận này đòi hỏi phải có một từ điển từ vựng đầy đủ, các corpus với kích thước lớn, cùng với các bước phân tích từ vựng đơn giản như: tách từ, phân tích cú pháp, gán nhãn từ (nhãn từ loại, nhãn cú pháp, nhãn ngữ nghĩa).
* Ưu điểm:
  + Độ chính xác cao.
* Nhược điểm:
  + Vì hoạt động dựa trên mô hình ngôn ngữ nên các phương pháp này sẽ phụ thuộc vào từng ngôn ngữ khác nhau.
  + Mất nhiều công sức xây dựng dữ liệu từ điển, corpus, bên cạnh đó dữ liệu trong corpus phải tương đối chuẩn về ngữ pháp, ngữ nghĩa. Việc này tương đối khó, nhất là với tiếng Việt hiện chưa có một corpus nào hoàn thiện, đầy đủ do các corpus chủ yếu được lấy từ internet bằng các Crawler, mà dữ liệu text trên internet không phải lúc nào cũng chuẩn xác.
  + Mất nhiều thời gian xử lý do phải thực hiện các thao tác như tách từ, phân tích cú pháp, gán nhãn…
* Hướng tiếp cận dựa vào âm tiết:
* Các phương pháp dựa trên những hướng tiếp cận này hoạt động với các thuật toán khác nhau, dựa trên thông tin thống kê từ các n-gram trích xuất ra được từ trong dữ liệu huấn luyện. Dữ liệu huấn luyện chỉ cần ở dạng thô, không cần thiết phải qua các bước xử lý như tách từ, gán nhãn.
* Ưu điểm:
  + Không phụ thuộc vào ngôn ngữ.
  + Dễ dàng triển khai.
  + Tốc độ xử lý cao.
* Nhược điểm:
  + Với những ngôn ngữ mà việc thêm dấu cho từ phụ thuộc vào ngữ cảnh và dễ gặp nhập nhằng như tiếng Việt thì tỉ lệ chính xác của phương pháp khá thấp và thường chỉ ứng dụng cho những ngôn ngữ đơn giản (ít dấu, ít đồng âm dị nghĩa, ...)

Việc chọn hướng tiếp cận nào để giải quyết bài toán thêm dấu cần phải dựa vào nhiều tiêu chí, thí dụ như luật thêm dấu của ngôn ngữ đó theo luật ngữ pháp hay theo luật ngữ nghĩa, dữ liệu huấn luyện, từ điển hiện có có đầy đủ, chuẩn hay không, phụ thuộc vào yêu cầu về tôc độ của hệ thống, yêu cầu về độ chính xác, yêu cầu của người dùng…

### 4.2.2. Một số kết quả trong bài toán thêm dấu.

#### 4.2.2.1 Cho các ngôn ngữ khác

Theo hai hướng tiếp cận trên, trên thế giới đã có khá nhiều các công trình nghiên cứu về vấn đề thêm dấu cho các ngôn ngữ khác nhau.

**Theo hướng tiếp cận dựa trên từ.**

* El-Bèze và các đồng nghiệp (1994) dùng phương pháp POS-tagging (Part Of Speech tagging) trong tiếng Pháp để lấy ra các thông tin về ngữ cảnh, thông tin thống kê N-Gram để quyết định có thêm một dấu vào từ đang xét hay không.
* Spiet và El-Bèze (1997) đã mô tả việc dùng mô hình N-Gram trong POS trong việc thêm dấu cho văn bản tiếng Pháp: họ đánh giá phương pháp của mình bằng việc dùng bộ dữ liệu 19,000 từ trong đó có 2,6 % số từ nhập nhằng, 89,5% số từ không cần dấu, và kết quả thu được khá khả quan với 99,31% chính xác.
* Yarowsky (1998) thực hiện thêm dấu cho tiếng Tây Ban Nha và tiếng Pháp bằng một decision-list framework.
* Simard (1998) cũng sử dụng phương pháp POS-tagging để thêm dấu cho tiếng Pháp.
* Cùng phương pháp tương tự Simard, Tufis và Chitu (1999) thực hiện thêm dấu cho tiếng Romani với độ chính xác cao mặc dù tiếng Romani phức tạp hơn, nhưng cũng như trước đó, phương pháp này bỏ qua, không xử lý được những từ nhập nhằng.

**Theo hướng tiếp cận dựa trên âm tiết:**

* Mihalcea (2002) thực hiện thêm dấu cho tiếng Romani sử dụng mô hình N-Gram và thực nghiệm với một hệ học máy memory-based (TIMBL) và cây quyết định phân loại, cô đưa ra kết quả chính xác tới 99%.
* Bobiceva (2008) đưa ra một phương pháp khác cũng để thêm dấu cho văn bản tiếng Romani, cô áp dụng phương pháp thống kê được sử dụng trong nén dữ liệu văn bản (PPM- prediction by partial matching).
* Zweigenbaum và Grabar (2002) mô tả một hệ thống được thiết kế đặc biệt cho việc khôi phục các phiên bản khác nhau của dấu phụ ‘e’ (é, è, ê, ë) trong từ điển chuyên ngành tiếng Pháp. Họ sử dụng hai phương pháp khác nhau: một bộ chuyển đổi trạng thái hữu hạn và bộ gán nhãn Brill để học quy tắc chuyển đổi theo ngữ cảnh cho chữ 'e' (mỗi từ thích hợp, có chứa một chữ ‘e’ được chia thành các chữ cái cấu thành của nó được coi là các token được gắn thẻ).

#### 4.2.2.2 Bài toán thêm dấu trong tiếng Việt

Việc thêm dấu cho văn bản tiếng Việt nói chung phức tạp hơn các ngôn ngữ khác bởi tỉ lệ từ có dấu và tỉ lệ nhập nhằng cao. Chính vì thế Bài toán thêm dấu cho tiếng Việt đã được đặt ra từ lâu, nhưng những nghiên cứu về bài toán này lại chưa thực sự thu được nhiều thành công, và cũng chưa có công trình, kết quả nghiên cứu nào về bài toán này được công bố chính thức trên các hội nghị khoa học. Dưới đây là danh sách một số chương trình đã có:

* Vietizer:
  + Hình thức: demo trên website
  + Nhóm phát triển: VietLab. (website: http://vietlabs.com/vietizer.html)
  + Độ chính xác: không công bố.
  + Phương pháp, mã nguồn: không công bố.
* Vietmarker:
  + Hình thức: phần mềm, add-on cho Microsoft Office 2003.
  + Nhóm phát triển: Vietmarker thuộc Học Viện Bưu Chính Viễn thông.
  + Độ chính xác: 93% (giới thiệu trên tạp chí E-chip số 458 của báo Vietnamnet).
  + Phương pháp: word-based
  + Mã nguồn: không công bố.

Trước hiện trạng như vậy, cần thiết phải có một phương pháp mới với độ chính xác cao hơn cũng như một hệ thống mở hơn để giải quyết bài toán này.

### 4.2.3. Hướng tiếp cận bằng dịch máy (Machine translation).

Chương này sẽ trình bày một phương pháp dựa trên hướng tiếp cận hoàn toàn mới cho bài toán thêm dấu: thêm dấu cho văn bản tiếng Việt không dấu bằng dịch máy. Ý tưởng cơ bản của phương pháp là: coi tiếng Việt không dấu (hoặc thiếu dấu) là một ngôn ngữ mới và dùng hệ dịch máy để dịch văn bản từ ngôn ngữ mới này sang tiếng Việt chuẩn.

Xây dựng mới một hệ dịch máy là một việc không hề đơn giản, để thực thi phương pháp này em đã sử dụng một hệ dịch máy mã nguồn mở có tên Moses – một hệ dịch máy dựa trên thống kê, được phát triển từ năm 2005 bởi trường đại học Edinburgh.

## 4.3. Hệ dịch máy Moses.

### 4.3.1. Giới thiệu hệ dịch máy Moses

Moses là một hệ dịch máy (Machine Translation - MT) sử dụng phương pháp thống kê. Đây là phương pháp chủ đạo trong lĩnh vực dịch tự động tại thời điểm này, và được sử dụng bởi các hệ thống dịch trực tuyến triển khai bởi những những công ty công nghệ lớn như Google với Google Translate và Microsoft với Bing Translator. Trong dịch máy thống kê (SMT), hệ thống dịch được đào tạo bởi số lượng lớn dữ liệu song song (từ những dữ liệu này, hệ thống học cách dịch những phân đoạn nhỏ.

Dữ liệu song song là một tập hợp các câu thuộc hai ngôn ngữ khác nhau, trong đó mỗi câu trong ngôn ngữ này là câu dịch tương ứng của nó trong ngôn ngữ khác, các câu được dóng hàng từng đôi một.

Quá trình huấn luyện của Moses sử dụng corpus dữ liệu song song và sử dụng đồng thời các từ cũng như các cụm từ để suy ra sự tương ứng dịch giữa hai ngôn ngữ quan tâm. Dịch máy dựa trên cụm từ (Phrase-based MT), sự tương ứng dịch rất đơn giản giữa những dãy các từ liên tục. Tuy nhiên với dịch máy dựa trên cụm từ có phân cấp (Hierarchical Phrase-based MT) hoặc dựa trên cú pháp (Syntax-based MT) nhiều luật cấu trúc phải được thêm vào sự dịch tương ứng này.

Hiện tại Moses hỗ trợ ba mô hình dịch: mô hình dịch dựa trên cụm từ (phrase-based), mô hình dịch dựa trên cú pháp (syntactic-based) và mô hình dịch dựa trên các yếu tố ngôn ngữ (factored-based). Dù sử dụng mô hình nào thì yếu tố then chốt quyết định chất lượng dịch chính là chất lượng của corpus song song dùng để huấn luyện.

### 4.3.2. Cơ sở lý thuyết dịch máy thống kê.

Dịch máy dựa trên phương pháp thống kê là mảng nghiên cứu được bắt đầu từ những năm 1980 bởi Candide trong một dự án của IBM. Phương pháp đầu tiên IBM đưa ra là dịch từng từ riêng rẽ từ ngôn ngữ nguồn tới ngôn ngữ đích, trong đó có cho phép việc thêm hoặc xóa từ.

Sau đó, nhiều nhà nghiên cứu công bố chất lượng bản dịch tốt hơn với phương pháp dịch theo cụm từ. Hiện tại hệ thống dịch máy sử dụng dịch cụm từ đang được sử dụng ở hầu hết những hệ thống dịch máy tên tuổi như CMU, IBM, ISI, và hệ dịch máy của Google.

Bên cạnh đó cũng có một vài phương pháp khác để thực hiện dịch máy thống kê, thí dụ như xây dựng mô hình dựa trên cú pháp bằng cách sử dụng cây cú pháp thực tế được tạo ra bởi bộ phân tích cú pháp, hoặc các phương pháp cây chuyển đổi được hỗ trợ bởi mô hình sắp xếp cú pháp.

Phương pháp dịch máy dựa trên cụm từ đươc trình bày dưới đây được định nghĩa bởi Koehn/Och/Marcu, (2003) [10].

Dịch máy thống kê được hiểu là dịch văn bản từ ngôn ngữ này sang một ngôn ngữ khác dựa trên mô hình được sinh ra một cách tự động từ ngữ liệu.

Với ví dụ cho câu tiếng Anh đầu vào e, tìm câu tiếng Việt v sao cho xác suất điều kiện P(v|e) là lớn nhất. Do v và e phụ thuộc lẫn nhau, nên theo lý thuyết xác suất có điều kiện công thức Bayes:

argmax(P(v|e)) = argmax(P(v) \* P (e|v))

Trong đó :

* P(v) là mô hình ngôn ngữ
* P(e|v) là mô hình dịch

Bộ giải mã (decoder)

argmax(P(v|e)) = argmax(P(v) \* P (e|v))

Mô hình ngôn ngữ P(v)

Mô hình dịch P(e|v)

Ngữ liệu đơn ngữ

Ngữ liệu song ngữ

Câu nguồn e: ‘toi la sinh vien’

Câu đích v: ‘tôi là sinh viên’

Hình 4.1 Mô hình hệ thống dịch máy thống kê

* **Mô hình ngôn ngữ**
* Tính p(v):

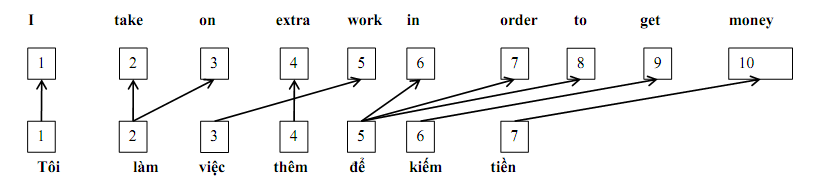
Chuỗi xác suất: P(v)= p(v1,v2,…,vI) = p(v1) p(v2|v1)…p(vI|v1,v2,…,vI-1)

* Giả thuyết Markov:
* N-gram: cụm n-từ liên tiếp
* Xấp xỉ n-gram: P (vI |v1, v2,…, vI-1) ≈ p (vI| vI-n+1, vI-n+2,…, vI-1)
* Tri-gram(n=3): P(v)= p(v1,v2,…,vI) = ∏ p(vk | vk-2 vk-1) với k=1,…,I
* Phương pháp Trigram được làm trơn:

Trong đó: là các hệ số, T là xác suất Tri-gram, B là xác suất Bi-gram, U là xác suaát Uni-Gram.

* **Mô hình dịch**

Tính xác suất P(e|v) dựa trên việc gióng hàng từ (Word-Alignment) giữa hai câu Anh (ngôn ngữ nguồn) – Việt (ngôn ngữ đích).



Hình 4.2 Gióng hàng từ giữa hai câu trong hai ngôn ngữ

Gióng hàng: a {11, 2(2,3), 35, 44, 5(6,7,8), 69, 710} trong đó:

* 11, 35, 44, 69, 710: gióng hàng 1-1
* 2(2,3), 5(6,7,8): gióng hàng 1-nhiều

Đối với một cặp câu (v,e) có rất nhiều cách liên kết từ khác nhau, do đó để đánh giá độ chính xác của từng liên kết thì phải quan tâm đến xác suất liên kết P(A=a|V=v,E=e) gồm bộ ba biến ngẫu nhiên (V,A,E):

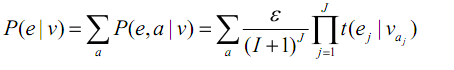
, trong đó

Để chọn được một liên kết tốt nhất thì cần phải chạy mô hình dịch từ 1 đến 5 thông qua một số các vòng lặp của thuật toán ước lượng giá trị cực đại (Estimation Maximization Algorithm) cho từng thông số của mô hình.

* *Mô hình dịch dựa trên đơn vị từ*

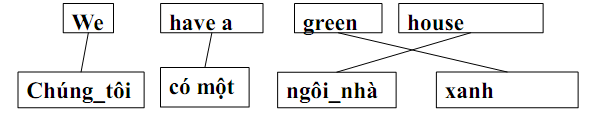
Được đề xuất bởi IBM, với giả thuyết chọn cách gióng hàng đơn giản.

* Đơn vị dịch: từ.
* Phương pháp sinh: chia quá trình gióng hàng thành các bước nhỏ hơn. Mô hình hóa các bước đó bằng các xác suất điều kiện dựa trên các giả thuyết đơn giản.
* Xác suất p(e|v) được tính:



Hạn chế của mô hình này là chỉ giải quyết được gióng hàng 1-1, 1-nhiều, không thể giải quyết được gióng hàng nhiều-nhiều, do đó cần thiết phải phát triển lên gióng hàng theo cụm từ,

* *Mô hình dịch dựa trên đơn vị cụm từ (phrase)*
* Đơn vị dịch: cụm từ.



* Mỗi cụm tiếng Việt tương ứng với một cụm từ tiếng Anh , là xác suất dịch cụm từ.
* Các cụm từ có thể bị dịch chuyển, là xác suất dịch chuyển.
  + : vị trí đầu tiên của cụm từ tiếng Anh ứng với
  + : vị trí đầu tiên của cụm từ tiếng Anh ứng với

Xác suất dịch tốt nhất:

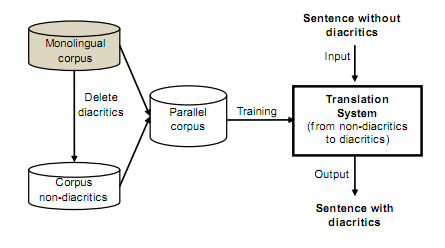
* **Giải mã**

Ngữ liệu sau khi qua 2 mô hình ngôn ngữ và mô hình dịch ta được bảng xác suất cho từng thông số tương ứng. Vấn đề tìm ra tích số P(e)P(v|e) lớn nhất.

Có hai thuật giải và một thuật toán tối ưu cho mô hình tìm kiếm: thuật giải tìm kiếm tham lam, thuật giải tìm kiếm dựa trên ngăn xếp, và thuật toán tìm kiếm theo chu trình Hamilton tối ưu. Hai thuật giải có thời gian nhanh xử lý nhanh hơn thuật toán nhưng kết quả thấp hơn thuật toán.

## 4.4. Sử dụng Moses cho bài toán thêm dấu tiếng Việt.

Để thêm dấu cho văn bản tiếng Việt không dấu, ta coi tiếng Việt không dấu là một ngôn ngữ mới, sau khi huấn luyện hệ dịch tự động Moses với corpus song ngữ (tiếng Việt không dấu – tiếng Việt có dấu) ta đưa văn bản tiếng Việt không dấu làm đầu vào, sau quá trình dịch sẽ thu được văn bản tiếng Việt có dấu.



Hình 4.3 Mô hình áp dụng dịch máy cho bài toán thêm dấu

Việc triển khai hệ dịch máy Moses để thêm dấu tiếng Việt được thực hiện qua các bước sau:

* + Cài đặt Moses.
  + Chuẩn bị dữ liệu huấn luyện.
  + Huấn luyện mô hình ngôn ngữ.
  + Huấn luyện hệ dịch máy.
  + Kiểm tra

### 4.4.1. Bước 1: Cài đặt Moses Toolkit.

Moses toolkit được viết bằng ngôn ngữ C++ và một số tool viết bằng Perl Script. Mặc dù có phiên bản chạy trên nền Windows nhưng Moses chạy tốt nhất trên nền Unix. Để chạy được Moses, những phần mềm cần phải cài đặt:

* Moses.
* GIZA++, để thực hiện gióng hàng cụm từ trong các cặp câu trong corpus song ngữ.
* IRSTLM, để xây dựng mô hình ngôn ngữ.

### 4.4.2. Bước 2: Tạo corpus.

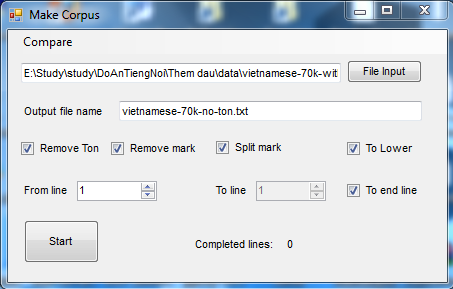
Để huấn luyện cho hệ dịch máy Moses, em đã xây dựng một corpus tiếng Việt không dấu và một corpus tiếng Việt có dấu tương ứng.

Trước tiên, để xây dựng corpus các câu tiếng Việt có dấu, em đã tạo một Clawrer lấy các bài viết online từ các báo mạng tiếng Việt như báo Tin Nhanh Việt Nam - vnexpress.net, báo Việt Nam Net – vietnamnet.vn, báo Vietnam News Agency – vietnamnews.vn, tách nội dung tiếng Việt từ văn bản HTML thu được. Corpus thu được khoảng 70.000 câu tiếng Việt (23Mb text).

Sau khi có được corpus các câu tiếng Việt, việc tạo corpus tiếng Việt không dấu khá dễ dàng bằng việc thay thế các chữ cái có dấu bằng các chữ cái không dấu tương ứng theo luật:

Bảng 4.3 Luật bỏ dấu cho các ký tự trong tiếng Việt

| Chữ cái có dấu | Chữ cái không dấu |
| --- | --- |
| "a", "à", "ả", "ã", "á", "ạ", "ă", "ằ", "ẳ", "ẵ", "ắ", "ặ", "â", "ầ", "ẩ", "ẫ", "ấ", "ậ" | "a" |
| "e", "è", "ẻ", "ẽ", "é", "ẹ", "ê", "ề", "ể", "ễ", "ế", "ệ" | "e" |
| "i", "ì", "ỉ", "ĩ", "í", "ị" | "i" |
| "o", "ò", "ỏ", "õ", "ó", "ọ", "ơ", "ờ", "ở", "ỡ", "ớ", "ợ", "ô", "ồ", "ổ", "ỗ", "ố", "ộ" | "o" |
| "u", "ù", "ủ", "ũ", "ú", "ụ", "ư", "ừ", "ử", "ữ", "ứ", "ự" | "u" |
| "y", "ỳ", "ỷ", "ỹ", "ý", "ỵ" | "y" |
| "đ", "d" | "d" |



Hình 4.4 Chương trình tạo corpus tiếng Việt không dấu từ corpus tiếng Việt có dấu

Trước khi dùng corpus cho huấn luyện, để tăng độ chính xác cho hệ thống, corpus thu được cần được “làm sạch” qua các thao tác đơn giản:

* Tách mỗi câu thành một dòng riêng rẽ.
* Chuyển tất cả các từ trong corpus về dạng viết thường.
* Tách riêng rẽ các dấu câu liền với từ (“xin chào, ” 🡪 “xin chào , ”).

Sau bước này ta thu được 2 corpus song song: vietnamese.ton, Vietnamese.noton, mỗi corpus này được tách làm 2 phần: một phần gồm 60.000 (vietnamese-60k.ton, vietnamese-60k.noton) câu để làm dữ liệu huấn luyện, một phần 10.000 câu còn lại để phục vụ test (vietnamese-10k-source.txt, vietnamese-10k-test.txt).

### 4.4.3. Bước 3: Huấn luyện mô hình ngôn ngữ.

Mô hình ngôn ngữ đích dùng để chắc chắn rằng câu đầu ra chính xác về mặt cú pháp theo ngôn ngữ này.

Công cụ IRSTLM giúp chúng ta tạo mô hình ngôn ngữ cho tiếng Việt có dấu (ngôn ngữ đích) một cách dễ dàng trên tập corpus có sẵn với một vài dòng lệnh:

mkdir ~/lm

cd ~/lm

~/irstlm/bin/add-start-end.sh < ~/corpus/n > vietnamese.ton

export IRSTLM=$HOME/irstlm; ~/irstlm/bin/build-lm.sh -i vietnamese.sb.ton -t ./tmp -p \ -s improved-kneser-ney -o vietnamese.lm.ton

~/irstlm/bin/compile-lm --text yes vietnamese.lm.ton.gz vietnamese.arpa.ton

Sau bước này ta thu được mô hình ngôn ngữ cho tiếng Việt có dấu trong file vietnamese.arpa.ton

### 4.4.4. Bước 4: Huấn luyện hệ thống.

Trong bước này ta sẽ dùng corpus đã chuẩn bị để huấn luyện cho hệ dịch máy. Trong quá trình huấn luyện chức năng word-alignment sẽ được thực thi với công cụ GIZA++, các cụm từ được phân tách và đánh trọng số (scoring), tạo các bảng từ vựng liên kết, bảng hoán đổi vị trí từ. Các dòng lệnh dùng để chạy bước huấn luyện:

mkdir ~/working

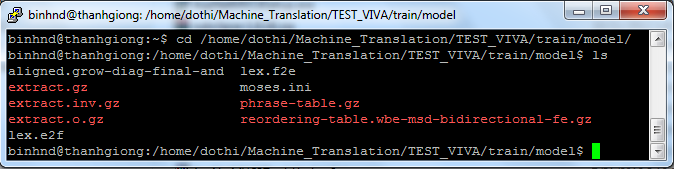
cd ~/working

nohup nice ~/mosesdecoder/scripts/training/train-model.perl -root-dir train -corpus ~/corpus/Vietnamese-60k \

-f noton -e ton -alignment grow-diag-final-and -reordering msd-bidirectional-fe \

-lm 0:3:$HOME/lm/vietnamese.arpa.ton:8 -external-bin-dir ~/mosesdecoder/tools >& training.out &

Sau bước này ta thu đước file *Moses.ini* chứa các thông tin cấu hình của hệ dịch và các file *phrase-table.gz, reodering-table.wbe-msd-bidirectional-fe.gz* chứa thông tin trích xuất được sau quá trình huấn luyện.

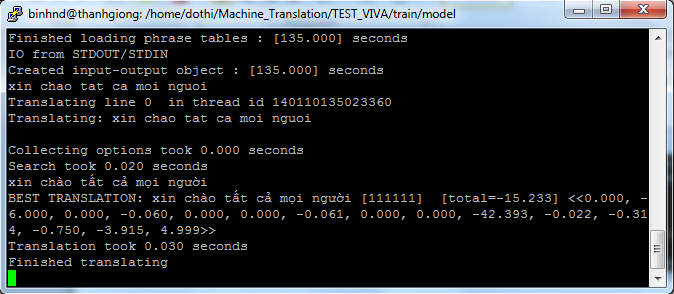


Hình 4.5 Các file tạo ra sau bước huấn luyện

### 4.4.5. Bước 5: Kiểm tra.

Sau bước huấn luyện, ta có thể chạy thử chương trình Moses từ file moses.ini thu được và kiểm tra kết quả.

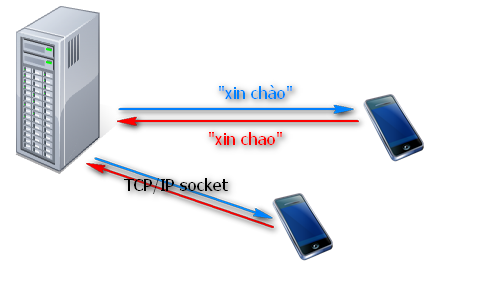
Hệ dịch máy Moses đã hoạt động, và kết quả dịch kiểm tra khá tốt: thêm dấu cho câu “xin chao tat ca moi nguoi” 🡪 “xin chào tất cả mọi người”



Hình 4.6 Chạy kiểm tra hệ dịch Moses

## 4.5. Triển khai module thêm dấu văn bản cho VIVA.

Hệ dịch máy Moses đã được triển khai thành công trên máy chủ LINUX của viện MICA, bài toán thêm dấu đã được giải quyết. Tuy nhiên việc đưa cả hệ thống vào ứng dụng VIVA trên điện thoại là không thể, do hệ thống chạy sử dụng rất nhiều tài nguyên bộ nhớ (khoảng 8Gb RAM), vì vậy để có thể áp dụng hệ thống này vào việc thêm dấu cho nội dung tin nhắn trong ứng dụng VIVA em đã thực hiện xây dựng một server dịch Moses cho phép các client kết nối tới, gửi văn bản có tiếng Việt không dấu qua TCP/IP socket, server Moses sẽ thực hiện dịch (thêm dấu) và trả về văn bản tiếng Việt có dấu.



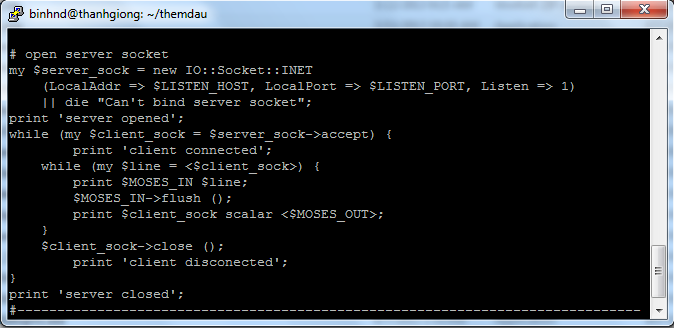
Hình 4.7 Mô hình Client-Server cho bài toán thêm dấu

Quá trình triển khai module thêm dấu cho VIVA sẽ được trình bày trong hai phần dưới đây.

### 4.5.1. Tạo server dịch.

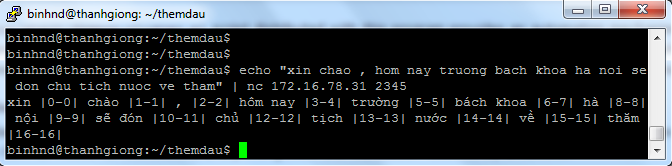
Trong quá trình khởi động Moses, hệ thống sẽ đọc file moses.ini và tìm đến những file thông tin trích xuất được sau khi huấn luyện và load vào bộ nhớ RAM. Quá trình khởi động Moses diễn ra trong khoảng 90s, vì thế để có thể đáp ứng được cácc client tức thì, Moses cần được khởi động sẵn và luôn chạy ở chế độ thường trực, mặc dù điều này có khiến tài nguyên RAM bị chiếm dụng khá nhiều.

Server dịch Moses được khởi chạy bằng một đoạn Perl Script đơn giản, đoạn script này cho phép nhập vào hai tham số khi chạy là tên server và số hiệu cổng.



Hình 4.8 Script khởi chạy server Moses

Sau khi server dịch Moses đã được khởi động, từ một máy bất kỳ, ta có thể kiểm tra sự hoạt động của server dịch bằng công cụ NetCat.



Hình 4.9 Kiểm tra hoạt động của server Moses bằng công cụ NetCat

* **Kết quả:** Server Moses đã hoạt động và trả lại kết quả dịch cho client khá chính xác: “xin chao , hom nay truong bach khoa ha noi se don chu tich nuoc ve tham” 🡪 “xin chào , hôm nay trường bách khoa hà nội sẽ đón chủ tịch nước về thăm”

### 4.5.2. Xây dựng module thêm dấu trong VIVA.

Để kết nối tới server dịch Moses, trong ứng dụng VIVA em đã xây dựng thêm một package com.mica.viva.diacritic, có chứa lớp VietnameseDiacritic được mô tả như dươis đây:

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.diacritic | |
| Class | Đặc tả | |
| VietnameseDiacritic | Lớp cha | java.lang.Object |
| 1. **public** String AddDiacritic(String inputSentence) Kết nối tới server dịch Moses, gửi lên chuỗi cần thêm dấu bằng socket, nhận kết quả dịch từ server và trả về. 2. **private** String standardiseInputSentence(String sentence) chuẩn hóa chuỗi trước khi gửi lên server. 3. **private** String normaliseOutputSentence(String sentence) chuẩn hóa chuỗi trả về từ server. | |

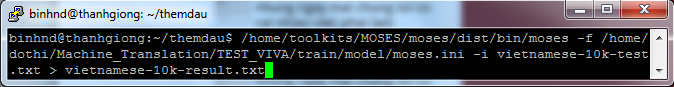
* **Kết quả:** Sau khi ghép nội module thêm dấu vào VIVA và sử dụng trong chức năng đọc tin nhắn, nội dung không dấu đã được thêm dấu hoàn chỉnh và bộ tổng hợp đã có thể tổng hợp được nội dung này thành tiếng cho người dùng.

Hình 4.10 Nội dung tin nhắn không dấu được thêm dấu trong chức năng đọc SMS

## 4.6 Đánh giá hệ thống.

Để dánh giá tỉ lệ chính xác của hệ thống dịch Moses đã xây dựng cho việc thêm dấu cho văn bản tiếng Việt, em dùng 2 file trích từ corpus (trình bày trong phần 4.4.2): file không dấu gồm 10.000 câu tiếng Việt không dấu, file có dấu gồm 10.000 câu tiếng Việt có dấu tương ứng. File không dấu vietnamese-10k-test.txt được dịch bởi hệ Moses cho ra file kết quả vietnamese-10k-result.txt. File kết quả này được so sánh với file nguồn vietnamese-10k-source.txt, để đánh giá độ chính xác của hệ thống.



Hình 4.11 Chạy Moses để thêm dấu cho file 10.000 câu tiếng Việt không dấu

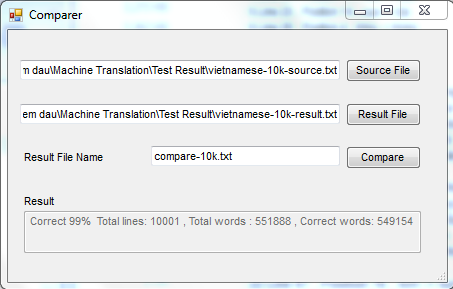
Để so sách 2 file: file kết quả và file nguồn, em đã sử dụng một công cụ so sánh word by word tự tạo và sử dụng thêm công cụ tính điểm BLEU cho hệ dịch máy.

### 4.6.1 Đánh giá bằng so sánh Word-By-Word.

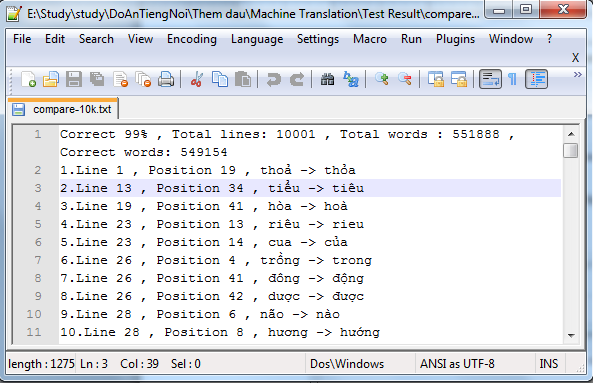
Công cụ so sánh này được em viết trên nền Windows với chức năng sau:

* Nhận hai file đầu vào: file nguồn (vietnamese-10k-source.txt), file kết quả thêm dấu (vietnamese-10k-result.txt).
* So sánh từng dòng tương ứng, trong từng dòng, so sánh từng từ tương ứng, đếm tổng số từ được thêm dấu đúng, tổng số từ được so sánh, tổng số dòng.
* Tính tỉ lệ dịch đúng: tỉ lệ dịch đúng = (số từ đúng)/(tổng số từ) \* 100 (%).
* Ghi ra file các kết quả so sánh, đông thời ghi ra các từ thêm dấu sai cùng vị trí số dòng, vị trí từ trên dòng đó.

Phương pháp này tuy đơn giản nhưng hoàn toàn có thể tin tưởng được, bởi đặc thù bài toán là chỉ thêm dấu cho các từ trong câu, vị trí các từ, cú pháp câu không hề thay đổi.



Hình 4.12 Chương trình so sách file kết quả thêm dấu và file nguồn



Hình 4.13 File thông tin so sánh

* **Kết quả**: tỉ lệ đúng ~ 99,5%

### 4.6.2. Đánh giá bằng điểm BLEU.

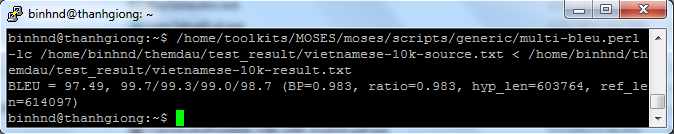
* **Điểm BLEU trong đánh giá hệ dịch máy:**

BLEU là một phương pháp dùng để đánh giá chất lượng bản dịch được đề xuất bởi IBM tại hội nghị ACL ở Philadelphie vào tháng 7-2001. Ý tưởng chính của phương pháp là so sánh kết quả bản dịch tự động bằng máy với một bản dịch chuẩn dùng làm bản đối chiếu. Việc so sánh được thực hiện thông qua việc thống kê sự trùng khớp của các từ trong hai bản dịch có tính đến thứ tự của chúng trong câu (phương pháp n-grams theo từ). Phương pháp này dựa trên hệ số tương quan giữa bản dịch máy và bản dịch chính xác được thực hiện bởi con người để đánh giá chất lượng của một hệ thống dịch.

Việc đánh giá được thực hiện trên kết quả thống kê mức độ trùng khớp các n-grams (dãy ký tự gồm n từ hoặc ký tự) từ kho dữ liệu của kết quả dịch và kho các bản dịch tham khảo có chất lượng cao. Giải thuật của IBM đánh giá chất lượng của hệ thống dịch qua việc trùng khớp của các n-grams đồng thời nó cũng dựa trên cả việc so sánh độ dài của các bản dịch.

Giá trị *score bleu* đánh giá mức độ tương ứng giữa hai bản dịch và nó được thực hiện trên từng phân đoạn, ở đây phân đoạn được hiểu là đơn vị tối thiểu trong các bản dịch, thông thường mỗi phân đoạn là một câu hoặc một đoạn. Việc thống kê đồ trùng khớp của các n-grams dựa trên tập hợp các n-grams trên các phân đoạn, trước hết là nó được tính trên từng phân đoạn, sau đó tính lại giá trị này trên tất cả các phân đoạn.

* **Kết quả:** sau khi chạy dòng lệnh để gọi công cụ tính điểm bleu cho 2 file, file nguồn và file kết quả dịch, công cụ cho thấy điểm bleu đạt 97.49. Điều này thêm một lần nữa khẳng định chất lượng dịch là rất tốt.



Hình 4.14 Tính điểm BLEU

* **Kết luận:** qua hai phương pháp đánh giá khách quan, kết quả đánh giá cho thấy tỉ lệ chính xác trong việc thêm dấu bằng phương pháp dịch máy dựa trên hệ dịch Moses là rất cao (trên 99%), đây là tỉ lệ chính xác cao nhất đã từng công bố của bài toán này. Với lưu ý rằng, corpus để huấn luyện hệ dịch không quá lớn (70.000 câu ~ 20Mb text) ta có thể hi vọng, với corpus lớn hơn, nhiều lĩnh vực, chủ đề trong nội dung hơn (hiện tại mới chỉ có nội dung từ các bài báo mạng, thiếu các nội dung hội thoại như chat, sms, comment, email… ) thì chất lượng của hệ thống sẽ còn cao hơn.

# CHƯƠNG TRÌNH VIVA PHIÊN BẢN DEMO

## 5.1 Ghép nối các module

Sau khi các thành viên trong nhóm xây dựng xong các module của VIVA (trình bày trong mục 2.1), các module được ghép nối thành một chương trình hoàn chỉnh:

* Chương trình VIVA được đóng gói thành file VIVA.apk có thể cài đặt và chạy trên các máy cài hệ điều hành Android version 10 (2.3.3) trở lên.
* Các module nhận dạng, module hiểu được đặt trên server Unix địa chỉ IP 172.16.31.78 tại viện MICA. Ứng dụng VIVA trên máy client kết nối tới dịch vụ nhận dạng và hiểu câu lệnh trên server này qua địa chỉ host MICA.edu.vn cổng 1569.
* Hệ dịch máy Moses cung cấp dịch vụ chuyển văn bản tiếng Việt không dấu thành có dấu cũng trên server này tại cổng 1571.

## 5.2 Demo chương trình VIVA

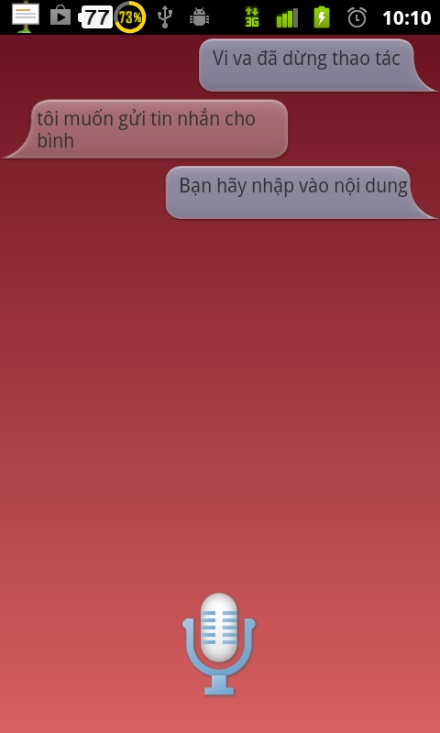
Nhóm đã thực hiện cài đặt và chạy thử chương trình VIVA trên hai các thiết bị Android bao gồm: điện thoại Google Nexus One (Android 2.3.6), máy tính bảng Galaxy Tab 7 (Android 2.3.3) và điện thoại Galaxy Note 2 (Android 4.1), các chức năng đều hoạt động tốt. Dưới đây là kết quả chạy thử chương trình trên điện thoại Google Nexus One:

Hình 5.1Màn hình khởi động & chào hỏi của VIVA

* Chức năng nhắn tin:

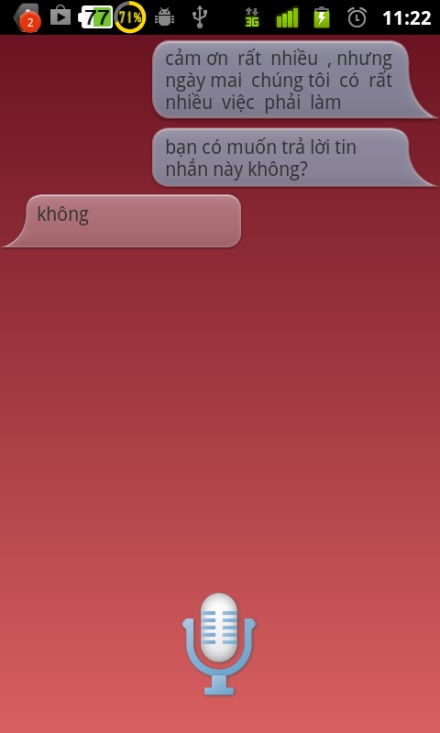
|  |  |  |
| --- | --- | --- |
| **Kịch bản** | **Yêu cầu của người dùng** | **Đáp ứng** |
| Gửi tin nhắn | Tôi muốn gửi tin nhắn | OK |
| Gửi tin nhắn với số điện thoại | Tôi muốn gửi tin nhắn cho số 01689928600 | OK |
| Gửi tin nhắn với tên người nhận | Gửi tin nhắn cho Bình | OK |
| Gửi tin nhắn với nội dung | Soạn tin nhắn, hôm nay đi chơi không? | OK |
| Gửi tin nhắn với tên người nhận + nội dung | Gửi tin nhắn cho bình, nội dung xin chào bạn | OK |

Hình 5.2 Màn hình thực hiện chức năng gửi tin nhắn

* Chức năng đọc tin nhắn đến

|  |  |  |
| --- | --- | --- |
| **Kịch bản** | **Yêu cầu của người dùng** | **Đáp ứng** |
| Đọc tin nhắn đến | Tin nhắn đến “xin chao” | OK |
| Đọc tin nhắn đến từ một người trong dạnh bạ gửi. | Tôi muốn gửi tin nhắn cho số 01689928600 | OK |
| Đọc tin nhắn tiếng Việt không dấu | Gửi tin nhắn cho Bình | OK |
| Đọc tin nhắn đến và trả lời | Soạn tin nhắn, hôm nay đi chơi không? | OK |

Hình 5.3 Màn hình đọc tin nhắn đến

* Chức năng trả lời thông tin thời tiết

|  |  |  |
| --- | --- | --- |
| **Kịch bản** | **Yêu cầu của người dùng** | **Đáp ứng** |
| Hỏi thời tiết | Tôi muốn xem thông tin thời tiết | OK |
| Hỏi thời tiết một tỉnh, thành phố | Hà nội thời tiết thế nào? | OK |
| Hỏi thời tiết một ngày | Thời tiết ngày mai thế nào? | OK |
| Hỏi thời tiết một theo thứ | Thời tiết thứ bảy thế nào? | OK |
| Hỏi thời tiết với thành phố + ngày | Thời tiết Sài Gòn thứ ba? | OK |

Hình 5.4 Màn hình chức năng trả lời thời tiết

* Chức năng trả lời thông tin giá vàng

|  |  |  |
| --- | --- | --- |
| **Kịch bản** | **Yêu cầu của người dùng** | **Đáp ứng** |
| Hỏi giá vàng | Tôi muốn xem thông tin giá vàng | OK |

Hình 5.5Màn hình chức năng trả lời giá vàng

* Chức năng trả lời thông tin giá tỉ giá

|  |  |  |
| --- | --- | --- |
| **Kịch bản** | **Yêu cầu của người dùng** | **Đáp ứng** |
| Hỏi tỉ giá | Tỉ giá đô la thế nào? | OK |

Hình 5.6 Màn hình chức năng trả lời tỉ giá

# TỔNG KẾT, ĐÁNH GIÁ

## 6.1 Kết luận chung

Trong quá trình thực hiện dự án VIVA, được sự chỉ dẫn, hỗ trợ rất nhiều từ phía các cán bộ viện MICA, đặc biệt là các thầy: Ts.Trần Đỗ Đạt, Ts.Nguyễn Quốc Cường và Ts.Mạc Đăng Khoa đã trực tiếp hướng dẫn và truyền đạ kiến thức. Nhờ vậy cả nhóm đã hoàn thành mục tiêu xây dựng chương trình VIVA, một thư ký ảo trên điện thoại Android với những chức năng cơ bản trước tiên.

Nhìn tổng thể, cả nhóm đã thành công trong việc xây dựng phiên bản VIVA demo, chương trình đã có thể nghe được người dùng, hiểu được người dùng muốn gì, thực thi các chức năng và trả lời người dùng bằng tiếng nói. Về mặt cá nhân, trong đồ án em đã trình bày được tổng quan hệ thống VIVA, các cơ sở lý thuyết dựa trên và kiến trúc các khối, cũng như trình bày chi tiết các công việc em đã thực hiện: thiết kế & xây dựng kiến trúc cơ sở cho VIVA, xây dựng module chức năng tin nhắn, nghiên cứu, giải quyết bài toán thêm dấu cho văn bản tiếng Việt không dấu với kết quả chính xác cao, đồng thời áp dụng kết quả đó vào chức năng đọc tin nhắn đến của VIVA.

Bên cạnh đó, việc phát triển VIVA vẫn cần phải tiếp tục để khắc phục những hạn chế sau đây:

* Khối nhận dạng tiếng nói:
  + Tỉ lệ nhận dạng chưa cao, hệ thống mới chỉ nhận dạng được số lượng câu trong phạm vi hẹp, liên quan tới các chức năng tin nhắn, truy vấn thông tin thời tiết, tỉ giá, giá vàng…
  + Tốc độ nhận dạng chưa cao khi có nhiều người dùng cùng sử dụng VIVA cùng lúc.
* Khối hiểu câu lệnh:
  + Mô hình khối hiểu sử dụng automat hữu hạn và bắt các từ khóa khá đơn giản, không thể hiểu được các câu lệnh phức tạp như “tôi muốn nhắn tin cho bảo bảo rằng …”
  + Khối hiểu đang gặp khó khăn khi muốn mở rộng phạm vi hiểu vì các trạng thái, từ khóa đang được đặt trong code (hard code).
* Khối tổng hợp tiếng:
  + Tiếng nói được tổng hợp đôi khi còn vấp và chưa được tự nhiên.
  + Chỉ có một giọng duy nhất.
  + Thiếu chức năng chuẩn hóa văn bản để đọc số, tiền tệ, ngày tháng hay các từ viết tắt, ký tự đặc biệt.
* Khổi thêm dấu tiếng Việt:
  + Thêm dấu cho các câu nói nội dung hội thoại, nội sms thông thường tỉ lệ chính xác chưa đạt cao nhất.

## 6.2 Hướng phát triển

Từ mục tiêu phát triển VIVA với những chức năng đề xuất (mục 1.3.3) và những mặt hạn chế còn tồn tại trong bản VIVA demo, nhóm đưa ra hướng phát triển tiếp tục cho hệ thống VIVA như sau:

* Xây dựng thêm các chức năng mới (mục 1.3.3), đặc biệt là chức năng hội thoại với người dùng, để chương trình thông minh và thân thiện hơn.
* Nâng cấp, khắc phục các mặt còn hạn chế:
* Khối nhận dạng tiếng nói:
  + Mở rộng cơ sở dữ liệu huấn luyện: thu âm trên nhiều câu trong nhiều chủ đề hơn, tăng số lượng người thu âm, thiết bị, ngữ cảnh thu âm.
  + Tăng tốc độ nhận dạng:
    - Chuyển phần xử lý trích chọn đặc trưng lên ứng dụng trên điện thoại, để giảm tải xử lý cho server đồng thời giảm kích thước gói tin truyền qua socket.
    - Tạo nhiều server hoặc luồng chạy khối nhận dạng song song, module Service Provider sẽ phân bố các client kết nối đến cho các server hay luồng hợp lý để tăng tốc độ đáp ứng của hệ thống.
* Khối hiểu câu lệnh:
  + Tăng khả năng mở rộng linh hoạt cho khối hiểu bằng việc phát triển một automat động với các trạng thái, từ khóa được lưu trong cơ sở dữ liệu hoặc trong file (.txt, .xml) tách biệt với thuật toán trong phần code.
  + Phát triển thêm chức năng hiểu nội dung tham số, hiểu lựa chọn của người dùng: thí dụ hệ thống khi yêu cầu người dùng nhập vào tên người nhận tin nhắn, VIVA có thể hiểu nội dung “cho bạn tôi tên là bình” tức là người dùng muốn gửi tin nhắn cho “bình”, hay “có chứ”, “tất nhiên rồi”, “tôi có muốn”, “đồng ý” là lựa chọn “có” khi confirm.
* Khối tổng hợp tiếng nói:
  + Tăng khả năng tùy biến tần số F0 để giọng đọc biểu cảm hơn
  + Tăng cơ sở dữ liệu để tổng hợp được chuẩn hơn và nhiều giọng hơn.
  + Phát triển module chuẩn hóa văn bản đầu vào trước khi đưa vào tổng hợp: cách đọc số, tiền tệ, ngày tháng, tên riêng, ký tự viết tắt…
* Khối thêm dấu cho văn bản tiếng Việ không dấu:
  + Mở rộng corpus huấn luyện: thêm các nội dung với chủ đề hội thoại, các nội dung SMS, Email, chat, comment.

Trong khuôn khổ đồ án tốt nghiệp đại học em xin kết thúc báo cáo của mình tại đây, trong bài báo cáo này không tránh khỏi những sai sót mặc dù đã được kiểm tra cẩn thận nên em rất mong được nghe những lời góp ý của thầy giáo phản biện cũng như hội đồng bảo vệ đồ án. Một lần nữa em xin gửi lời chân thành cảm ơn tới thầy giáo hướng dẫn Ts.Trần Đỗ Đạt và các cán bộ viện MICA đã giúp đỡ em hoàn thành đồ án này.

Danh sách tài liệu tham khảo

[1] <http://www.businessinsider.com/chart-of-the-day-androids-growth-finally-slows-down-2013-3> truy cập lần cuối ngày 01/06/2013.

[2] Flurry Analysis, January 2012 – January 2013, countries with at least 500.000 active devices as of January 2012.

[3] <http://en.wikipedia.org/wiki/Siri_(software)> truy cập lần cuối ngày 01/06/2013.

[4] <http://www.i-programmer.info/news/85-humour-/3222-how-siri-qreallyq-works.html> truy cập lần cuối ngày 01/06/2013.

[5] <http://en.wikipedia.org/wiki/Google_Now> truy cập lần cuối ngày 01/06/2013.

[6] <http://vlsp.vietlp.org:8080/demo/?page=resources> truy cập lần cuối ngày 01/06/2013.

[7] <http://www.statmt.org/moses> truy cập lần cuối ngày 01/06/2013.

[8] Philipp Koehn, Franz Josef, Daniel Marcu. “Statistical Phrase-Based Translation”, Conference of the North American, Chapter of the Associat ion for Computational Linguist ics on Human Language technology - Volume 1, 2003

[9] Tufiş, D., Chiţu, A. “Automatic Insertion of Diacritics in Romanian Texts”. Proceedings of the 5th Int ernat ional Workshop on Comput at ional Lexicography COMP LEX, P ecs, Ungaria, 1999,185-194

[10] Rada Mihalcea, Vivi Nastase, “Letter Level Learning for Language Independent Diacrit ics Restoration ”, In: P roceedings of CoNLL 2002.

[11] Simard, M. “Automatic Insertion of Accents in French Texts”. In Ide & Vuotilainen (eds) P roceedings of the Third Conference on Empirical Methods in Natural Language Processing, Granada, Spain, 27-35, 1998

[12] Luu Tuan Anh, Kazuhide Yamamoto , “A P oint wise Approach for

Viet namese Diacritics Rest oration”, in proceedings of IALP, 2012

[13] Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison- Burch, Richard Zens, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen and Christ ine Moran. “Moses: Open Source Toolkit for St at ist ical Machine Translat ion”. Proceedings of the ACL 2007.

PHỤ LỤC

**Phụ lục 1: Mô tả các lớp trong chương trình VIVA**

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller | |
| Class | Đặc tả | |
| BaseModuleController | Lớp cha | android.app.Activity |
| 1. Là lớp cơ sở của các lớp thực thi chức năng (gửi tin, đọc tin nhắn, trả lời thời tiết, giá vàng…) 2. Có thuộc tính **protected** Thread process là luồng thực thi chức năng. Các lớp con sẽ khai báo kịch bản, tiến trình thực hiện trong hàm **public** **void** run() của process | |
| UIController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển giao diện chính của ứng dụng. * Cung cấp các phương thức public để thao tác với giao diện chính: * **void** displayRequestText(String requestText) hiển thị câu nói của người dùng nhập vào. * **void** displayResponseText(String responseText) hiển thị câu trả lời của VIVA * **void** displayHtmlResult(String htmlContent) hiển thị thông tin dạng html | |
| InputController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc nhập câu nói. * Các phương thức public để thực hiện quá trình nhập câu nói: * **void** startInputSentence(Activity caller) bắt đầu nhập vào câu lệnh * **void** startInputParameter(Activity caller) bắt đầu nhập vào tham số (thí dụ: tham số tên ngươi nhận trong chức năng gửi tin nhắn) * **void** startInputChoise(Activity caller) bắt đầu nhập vào lựa chọn có hay không? | |
| ResponseController | Lớp cha | java.lang.Object |
| * Là lớp điều khiển việc trả lời bằng tiếng nói của VIVA. * Cung cấp các phương thức public thực hiện việc đọc và hiển thị câu trả lời: * **void** responseVoiceMessage(String message) đọc câu trả lời. * **void** responseMessage(String message) đọc câu trả lời đồng thời hiện câu trả lời ra màn hình. * **void** responseRandomMessage(String type) tìm câu ngẫu nhiên theo kiểu định trước trong cơ sở dữ liệu và trả lời người dùng. | |
| FunctionController | Lớp cha | java.lang.Object |
| * Là lớp tiếp nhận dữ liệu trả về từ khối hiểu và kích hoạt chức năng tương ứng. * Các phương thức public quan trọng: * Function getFunctionFromData(String st) nhận vào đoạn text có định dạng trả về từ khối hiểu và đưa ra chức năng tương ứng cùng các tham số nếu có. * **void** startFunction(Context context, Function function) Kích hoạt một chức năng. * **void** startFunctionForResult(Activity activity,Function function) kích hoạt một chức năng và trả về kết quả sau khi thực hiện xong. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.entity | |
| Class | Đặc tả | |
| Module | Lớp cha | java.lang.Object |
| * Lưu thông tin module chức năng | |
| Function | Lớp cha | java.lang.Object |
| * Lưu thông tin từng chức năng (gửi tin nhắn, đọc tin nhắn, trả lời thời tiết…) | |
| Parameter | Lớp cha | java.lang.Object |
| * Lưu thông tin tham số của chức năng (tham số nội dung trong chức năng gửi tin nhắn) | |
| Sentence | Lớp cha | java.lang.Object |
| * Lưu thông tin câu trả lời cho VIVA | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.resource | |
| Class | Đặc tả | |
| FunctionsManager | Lớp cha | java.lang.Object |
| 1. Đọc dữ liệu các chức năng từ file data và lưu vào danh sách các đối tượng. 2. ArrayList<Function> getFunctions() Lấy danh sách các chức năng 3. Function getFunction(String key) Lấy đối tượng chức năng - Function theo mã. | |
| SentencesManager | Lớp cha | java.lang.Object |
| * Đọc dữ liệu các câu nói từ file data, lưu vào danh sách câu nói. * ArrayList<Sentence> getSentences() Lấy danh sách các câu nói * Sentence getRandom(String type) Lấy ra một câu ngẫu nhiên theo kiểu truyền vào | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller.sms | |
| Class | Đặc tả | |
| SendingSms | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng gửi tin nhắn | |
| ReadingSms | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng đọc tin nhắn | |
| ReceivingSms | Lớp cha | android.content.BroadcastReceiver |
| * Thực hiện bắt bản tin Broadcast khi có tin nhắn đến, đọc các thông tin và kích hoạt chức năng đọc tin nhắn. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.controller.queryinformation | |
| Class | Đặc tả | |
| ForeCast | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin thời tiết. * Thông tin thời tiết được lấy từ trang *wunderground.com* qua API Webservice. | |
| Gold | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin giá vàng. * Thông tin giá vàng được lấy từ trang *24h.com.vn* | |
| Exchange | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Thực thi chức năng trả lời thông tin tỉ giá. * Thông tin tỉ giá được lấy từ trang *24h.com.vn* | |

|  |  |  |
| --- | --- | --- |
| Package | com.MICA.VIVA | |
| Class | Đặc tả | |
| ApplicationConfigs | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Lưu và hỗ trợ truy cập đến các tham số setting của chương trình. | |
| ApplicationContext | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Lưu ngữ cảnh hoạt động của chương trình. | |
| Constant | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Chứa các trường giá trị constant của trương trình | |
| MainActivity | Lớp cha | com.mica.viva.controller. BaseModuleController |
| * Activity chính, mang giao diện tương tác của ứng dụng với người dùng | |
| SettingActivity | Lớp cha | android.preference. PreferenceActivity |
| * Activity cho phép người dùng tùy chọn các trường setting của chương trình. | |

|  |  |  |
| --- | --- | --- |
| Package | com.mica.viva.inputting | |
| Class | Đặc tả | |
| VADRecord | Lớp cha | android.app.Activity |
| * Là lớp điều khiển việc thu âm, tự ngắt thu âm, gửi mảng mẫu tiếng nói lên server nhận dạng, nhận kết quả text từ server và hiển thị lên màn hình * Có thuộc tính: Thread thread1, Thread thread2, Thread thread3 * Thread 1: thu âm, đưa vào buffer và vào hàng đợi * Thread 2: lấy ra từ buffer lần lượt 8000 mẫu để áp dụng thuật toán VAD và đánh dấu có tự ngắt hay không việc thu âm ở Thread 1 * Thread 3: được tạo ra sau khi ngắt thu âm, lấy từ hàng đợi và gửi lên server nhận dạng và nhận kết quả text | |
| MsHandler | Lớp cha | android.os.Handler |
| * Phục vụ việc trao đổi giữa thread chính: UI và thread 2, thread 3 và UI. Khi ngắt thu âm, thread 2 gửi tín hiệu cho UI, UI sinh ra thread 3 để gửi lên server nhận dạng. Khi nhận được text từ server thread 3 báo tín hiệu cho UI để hiển thị text trên lên màn hình. | |
| Record\_Task | Thực thi | java.lang.Runnable |
| * Phục vụ thu âm, thực hiện bởi thread 1 | |
| Read\_DelteleFromBuffer\_Task2 | Thực thi | java.lang.Runnable |
| * Phục vụ việc tự ngắt thu âm, thực hiện bởi thread 2 | |
| SendSamplesToAndReceive FromServer | Thực thi | java.lang.Runnable |
| * Phục vụ việc gửi lên server và nhận text từ server, thực hiện bởi thread 3 | |
| VADForVectorInputOnline | Lớp cha | java.lang.Object |
| * Là lớp cung cấp các methode để áp dụng thuật toán phát hiện tiếng nói LTSD với đầu vào là một vector các mẫu tiếng nói kiểu int * Phục vụ việc: phân khung, phát hiện tiếng nói hay không theo từng khung, thiết lập thời gian ngắt thu âm, đánh dấu việc ngắt thu âm * Có các methode: * int getNumSamplesPerFrame() : trả về số mẫu trên 1 khung sau khi phân khung * int getNumSamplesShift() : trả về khoảng cách 2 khung liên tiếp ( tính theo số mẫu ) * int[] getVectorSamplesInput(): trả về vector các mẫu tiếng nói ban đầu * int[][] computeFrames\_Matrix(int numSamplesPerFrame, int numSamplesShift, int[] vectorSamplesInput) : trả về mảng 2 chiều sau khi phân | |

|  |  |
| --- | --- |
| Package | com.mica.viva.utility |
| * Package chứa các lớp utility với các hàm static để hỗ trợ thao tác như xử lý text (TextUtils), đọc danh bạ (ContactUtils), đọc ghi file (FileUtils)… | |