**LỜI CẢM ƠN**

Trước tiên chúng em xin chân thành cảm ơn các thầy cô giáo trong Học Viện Công Nghệ Bưu Chính Viễn Thông nói chung và trong khoa Công Nghệ Thông Tin nói riêng đã luôn hết lòng, tận tụy giảng dạy truyền đạt cho chúng em những kiến thức cũng như kinh nghiệm quý báu.

Em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Duy Phương– người đã trực tiếp hướng dẫn tận tình tạo điều kiện cho em hoàn thành đề tài.Cảm ơn thầy đã dành thời gian và những kinh nghiệm quý báu cho em. Em đã học được rất nhiều trong quá trình làm đồ án dưới sự hướng dẫn của thầy. Một lần nữa em xin chân thành cảm ơn thầy.

Trong báo cáo của em, em đã sử dụng tài liệu thu thập từ nhiều nguồn trên mạng internet, các ebooks về Công nghệ thông tin của nhiều thầy cô trong và ngoài trường.

Em xin chân thành cảm ơn.

Hà Nội, ngày tháng 12 năm 2019

Sinh viên

Đặng Quốc Việt

**NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM**

**(Của Người hướng dẫn)**

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

**Điểm:** …………………….………(bằng chữ: …..…………… ….)

**Đồng ý/Không đồng ý** cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp**?**

*…………, ngày tháng năm 2019*

**CÁN BỘ - GIẢNG VIÊN HƯỚNG DẪN**

**NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM**

**(Của Giáo viên phản biện)**

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

……………………………………………………………………………………..........

**Điểm:** …………………….………(bằng chữ: …..…………… ….)

*…………, ngày tháng năm 2019*

**CÁN BỘ - GIẢNG VIÊN PHẢN BIỆN**

# **DANH MỤC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **JDK** | Java Development Kit |
| **JRE** | Java Runtime Environment |
| **JVM** | Java Virtual Machine |
| **SDK** | Software Development Kit |
| **API** | Application Programming Interface |
| **HTML** | HyperText Markup Language |
| **JSON** | JavaScript Object Notation |
| **XML** | eXtensible Markup Language |
| **RPC** | Remote Procedure Call |

**LỜI MỞ ĐẦU**

Sự phát triẻn của khoa học kỹ thuật ngày càng nhanh góp phần nâng cao năng suất làm việc. Đặc biệt là sự ra đời và phát triển của các công nghệ mới nhằm tạo ra sự tự đông hoá , sự tiện lợi trong xã hội cũng như trong công nghiệp.

Đối với nước ngoài thì việc điều khiển bằng giọng nói đã đựơc nghiên cứu và chế tạo để ứng dụng vào đời sống và sản xuất . Riêng ở nước ta lĩnh vực này còn khá mới mẻ và chưa đươc áp dụng nhiều vào thực tế.

Trong những năm gần đây, điện thoại thông minh hay còn được gọi là smartphone luôn nhận được sự quan tâm hàng đầu trong lĩnh vực công nghệ thông tin. Nó được xem là một trong những tiêu chí lớn nhất đề đánh giá thương hiệu của các công ty công nghệ. Với tình hình đó các công ty công nghệ lớn trên thế giới dù trước đây chỉ sản xuất phầm mềm, hay một lĩnh vực linh kiện đơn thuần nào đó đều đầu tư chạy đua trong lĩnh vực sản xuất sản phẩm này. Kết quả của cuộc chạy đua đó là những chiếc điện thoại ngày càng “thông minh” hơn, mạnh mẽ hơn, chất lượng màn hình tốt hơn và giá ngày càng rẻ hơn. Nếu như trước đây sở hữu một chiếc smartphone là điều xa xỉ thì hiện nay smartphone là thiết bị thân thuộc đối với mọi người, mọi độ tuổi, đều dễ dàng sử dụng.

Xã hội càng hiện đại, khoa học kỹ phát triển thì nhu cầu sống của con người càng có nhu cầu sử dụng đầy đủ các thiết bị thong minh để phục vụ cho sinh hoạt và công việc của mình. Một thực tế là con người rất mong muốn được điều khiển smartphone mà không cần phải thao tác trực tiếp, ví dụ như những người đang tham giao phương tiện giao thông.

Hiện nay, tình trạng người tham gia giao thông vừa điều khiển phương tiện vừa sử dụng điện thoại di động đang diễn ra khá phổ biến, nhất là đối với giới trẻ. Các hành vi như: nhắn tin, nghe điện, thậm chí là chụp ảnh trên điện thoại di động khi đang lưu thông trên đường là thói quen hết sức nguy hiểm, được các cơ quan chức năng xác định là một trong những nguyên nhân chính gây ra các vụ tai nạn giao thông nghiêm trọng trong thời gian vừa qua. Việc nhắn tin, nghe điện thoại lúc lái xe gây ra tình trạnh mất tập trung khi xử lý những tình huống đòi hỏi tốc độ nhanh chóng, vì thế đặt người lái xe và những người cùng tham gia giao thông đối với nguy hiểm. Để giải quyết vấn đề đó, cùng với việc tận dụng xu hướng phát triển và sức mạnh của công nghệ, tôi đã bắt tay vào nghiên cứu và thực hiện đề tài : “***Ứng dụng điều khiển smartphone bằng công nghệ nhận diện giọng nói***”.

**Mục lục.**

**I.   Giới thiệu về hệ thống**

1.Giới thiệu

1. Mục đích

- Đơn giản, tự động hoá việc sử dụng smart phone , không cần thao tác bằng tay việc sử dụng smart phone giờ đây chỉ đơn giản là sử dụng giọng nói.

- App sẽ nhận diện giọng nói của người dùng rồi thực hiện các thao tác tác mà người dùng mong muốn.

- App sử dụng giọng nói để điều khiển smartphone hướng đến 1 số lớp người dùng đặc biệt:

+ sử dụng cho người khiếm khuyết , sử dụng trong các nhà tình thương, lớp học tình thương.

+Hỗ trợ người dùng khi đang tham gia giao thông, tránh xảy ra những sự cố ngoài mong muốn.

**III.Công nghệ xử lý**

**1.Công nghệ hỗ trợ thiết bị di động ( Android)**

* 1. **Khái niêm android**

Android là hệ điều hành điện thoại di động mã nguồn mở miễn phí do Google phát triển dựa trên nền tảng của Linux. Bất kỳ một hãng sản xuất phần cứng nào cũng đều có thể tự do sử dụng hệ điều hành Android cho thiết bị của mình, miễn là các thiết bị ấy đáp ứng được các tiêu chuẩn cơ bản do Google đặt ra (có cảm ứng chạm, GPS, 3G,...).

Android là nền tảng cho thiết bị di động bao gồm một hệ điều hành, midware và một số ứng dụng chủ đạo. Bộ công cụ Android SDK cung cấp các công cụ và bộ thư viên các hàm API cần thiết để phát triển ứng dụng cho nền tảng Android sử dụng ngôn ngữ lập trình java.

Những tính năng mà nền tảng Android hổ trợ:

* Application framework**:** Cho phép tái sử dụng và thay thế các thành phần sẳn có của Android.
* Dalvik virtual macine**:** Máy ảo java được tối ưu hóa cho thiết bị di động.
* Intergrated browser**:** Trình duyệt web tích hợp được xây dựng dựa trên WebKit engine.
* Optimized graphics**:** Hổ trợ bộ thư viện 2D và 3D dự vào đặc tả OpenGL ES 1.0.
* SQLite**:** DBMS dùng để lưu trữ dữ liệu có cấu trúc.
* Hổ trở các định dạng media phổ biến như: MPEG4, H.264, MP3, AAC, ARM, JPG, PNG, GIF.
* Hổ trợ thoại trên nền tảng GSM (Phụ thuộc vài phần cứng thiết bị).
* Bluetooth, EDGE, 3G và WiFi (Phụ thuộc vài phần cứng thiết bị).
* Camera, GPS, la bàn và cảm biến (Phụ thuộc vài phần cứng thiết bị).
* Bộ công cụ phát triển ứng dụng mạnh mẽ.
  1. **Kiến trúc android**

Có bốn tầng cơ bản trong hệ điều hành Android: Application Framework, Android Runtime, Native Libraries, Linux Kernel. Mỗi tầng làm việc đều nhờ sự giúp đỡ của tầng bên dưới.



Hình 1.1 Mô hình kiến trúc nền tảng Android

**1.2.1 Application Framwork.**

Đây là tầng mà Google xây dựng cho các developer để phát triển các ứng dụng của họ trên Android, chỉ bằng cách gọi các API có sẵn mà Google đã viết để sử dụng các tính năng của phần cứng mà không cần hiểu cấu trúc bên dưới.

Bằng cách cung cấp một nền tảng phát triển mở, Android cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo thức, thêm các thông báo.

Tất cả các ứng dụng thường gồm một bộ các dịch vụ và hệ thống cơ bản sau:

View UI: dùng để xây dựng layout của ứng dụng bao gồm list view, text field, button, dialog, form…

Content Providers cho phép các ứng dụng có thể truy cập dữ liệu từ các ứng dụng khác hoặc để chia sẻ dữ liệu của riêng ứng dụng.

Resource Manager: cung cấp cách thức truy cập đến non-code resources như các asset, graphic, image, music, video…

Notification Manager: cho phép tất cả các ứng dụng hiển thị thông báo của mình trên hệ đièu hành.

Activity Manager: quản lý vòng đời của các ứng dụng.

Ở góc nhìn của người dùng ta có thêm tầng Application (là ứng dụng do người dùng viết).

**1.2.2 Native Libraries.**

Android bao gồm một tập hợp các thư viện C/C++ được sử dụng bởi nhiều thành phần khác nhau trong hệ thống Android. Điều này được thể hiện thông qua nền tảng ứng dụng Android. Một số các thư viện cơ bản được liệt kê dưới đây:

System C library**:** một thể hiện được xây dựng từ BSD của bộ thư viện hệ thống C chuẩn (libc), được điều chỉnh để tối ưu hóa cho các thiết bị chạy trên nền Linux.

Media libraries**:** Bộ thư viện hổ trợ trình diễn và ghi các định dạng âm thanh và hình ảnh phổ biến.

Surface manager**:** Quản lý hiển thị và kết hợp đồ họa 2D và 3D.

LibWebCore**:**Android dùng lại webkit engine cho việc render trình duyệt mặc định của hệ điều hành Android browser và cho dạng web nhúng (như HTML nhúng).

SGL**:** Engine hổ trợ đồ họa 2D.

3D libraries**:** Một thể hiện được xây dựng dựa trên các APIs của OpenGL ES 1.0. Những thư viện này sử dụng các tăng tốc 3D bằng phần cứng lẫn phần mềm để tối ưu hóa hiển thị 3D.

FreeType**:** Bitmap and vector font rendering.

SQLite**:** Một DBMS nhỏ gọn và mạnh mẽ, quản lý database của ứng dụng.

**1.2.3 Android Runtime.**

Mỗi ứng dụng Android chạy trên một process riêng của Dalvik VM (máy ảo).Dalvik được viết để chạy nhiều máy ảo cùng một lúc một cách hiệu quả trên cùng một thiết bị.

Máy ảo Dalvik thực thi các file mang định dạng .dex (Dalvik Excutable), định dạng này là định dạng đã được tối ưu hóa để chỉ chiếm một vùng nhớ vừa đủ xài và nhỏ nhất có thể. VM chạy các class (đã được compile trước đó bởi một trình biên dịch ngôn ngữ Java), sở dĩ Vm chạy được các class này là nhờ chương trình DX tool đã covert các class sang định dạng .dex

**1.2.4 Linux Kernel.**

Đây là nhân của hệ điều hành Android, mọi xử lý của hệ thống đều phải thông qua tầng này. Linux Kernel cung cấp các trình điều khiển thiết bị phần cứng như: camera, USB, wifi, bluetooth, display, power management…Android dựa trên Linux phiên bản 2.6 lựa chọn các tính năng cốt lõi nhưu bảo mật, quản lý bộ nhớ, quản lý tiến trình, mạng stack và các trình điều khiển phần cứng. Kernel hoạt động như một lớp trừu tượng giữa phần cứng và phần mềm còn lại của hệ thống.

**1.3.Retrofit library**

Trong quá trình phát triển ứng dụng android, có thể sẽ có lúc cần xử lý những công việc liên quan đến mạng như là gửi Request đến Webservice, nhận Response và xử lý những dữ liệu trả về. Những công việc này sẽ được thực hiện 1 cách đơn giản như Retrofit.

Retrofit là một thư viện được phát triển bởi Square ,là 1 type-safe HTTP client cho Java và Android được phát triển bởi Square. Retrofit giúp dễ dàng kết nối đến một dịch vụ REST trên web bằng cách chyển đổi API thành Java Interface.

Thư viện mạnh mẽ này giúp ta dễ dàng xử lý dữ liệu JSON hoặc sau đó phân tích cú pháp thành Plain Old Java Objects (POJOs). Tất cả các yêu cầu **GET**, **POST**, **PUT**, **PATCH**, và **DELETE** đều có thể được thực thi.

Giống như hầu hết các phần mềm nguồn mở khác, Retrofit được xây dựng trên nền của một số thư viện mạnh mẽ và công cụ khác. Đằng sau nó, Retrofit sử dụng OkHttp (từ cùng một nhà phát triển) để xử lý các yêu cầu mạng. Ngoài ra, Retrofit không tích hợp sẵn một bộ chuyển đổi JSON để phân tích từ JSON thành các đối tượng Java. Thay vào đó, nó hỗ trợ cho các thư viện chuyển đổi JSON sau đây để xử lý điều đó:

* Gson: com.squareup.retrofit:converter-gson
* Jackson: com.squareup.retrofit:converter-jackson
* Moshi: com.squareup.retrofit:converter-moshi

**1.4.Service**

Một Service là một thành phần (component) có thể thực hiện các hoạt động lâu dài trong background và nó không cung cấp một giao diện người dùng. Một thành phần khác của ứng dụng có thể start nó, và nó tiếp tục chạy trong background ngay cả khi người dùng chuyển sang ứng dụng khác. Ngoài ra một thành phần có thể liên kết (bind) với một Service để tương tác với Service đó, thậm chí là thực hiện truyền thông liên tiến trình IPC (interprocess communication - IPC bạn có thể hiểu là một hoạt động chia sẽ dữ liệu qua nhiều tiến trình, thông thường sử dụng giao thức truyền thông và nó phải có Client và Server). Ví dụ: một Service có thể thực hiện các giao dịch mạng, chơi nhạc, ra vào file I/O hoặc tương tác với một content provider, tất cả đều từ background.

**1.4.1 Foreground Service.**

Một Foreground Service thực hiện một số thao tác mà người dùng chú ý, có thể thấy rõ ràng. Ví dụ một ứng dụng nghe nhạc có thể chơi một bản nhạc và control nó bằng Foreground Service. Một điều bắt buộc là Foreground Service phải hiện thị một Notification. Foreground Service sẽ tiếp tục chạy ngay cả khi người dùng không tương tác với ứng dụng.

**1.4.2 Background Service.**

Một Background Service sẽ thực hiện các hoạt động mà không được người dùng chú ý trực tiếp. Ví dụ một ứng dụng sử dụng một service để thu gom bộ nhớ chẳng hạn thì service là một Background Service, hoạt động mà người dùng không cần thiết phải để ý.

**1.4.3 Bound Service.**

Một service được gọi là Bound khi một thành phần của ứng dụng ràng buộc với nó bởi lời gọi bindService(). Một Bound Service cung cấp một giao diện Client - Server cho phép các thành phần tương tác với nó: gửi yêu cầu, nhận kết quả và thậm chí là IPC. Một Bound Service chỉ chạy miễn là có một thành phần ràng buộc với nó. Có thể có nhiều thành phần ràng buộc với Bound Service cùng lúc, nhưng khi tất cả tháo bỏ ràng buộc (unbound) thì nó sẽ Destroy. Trước đây Service thường được chia là Started Service và Bound Service.

Một Started Service hay là Unbound Service là service được khởi động bằng phương thức startService() từ thành phần khác. Và nó sẽ tiếp tục chạy trong background kể cả khi thành phần khởi tạo nó bị phá hủy. Đây cũng là xem là một Background Service theo cách chia trên.

**1.5. Audio Record Api**

AudioRecord là 1 lớp quản lý tài nguyên âm thanh cho các ứng dụng android để ghi lại âm thanh từ phần cứng đầu vào của điện thoại android bằng cách “Pulling” (đọc) dữ liệu từ AudioRecord Object. Ta có thể đọc được dữ liệu từ AudioRecord từ 1 trong 3 phương thức sau: read (byte [], int, int), read (short [], int, int) hoặc read (java.nio.ByteBuffer, int). Việc lựa chọn sử dụng phương thức nào sẽ dựa trên định dạng lưu trữ dữ liệu âm thanh mà thuận tiện nhất cho người dung AudioRecord.

Khi tạo, một đối tượng AudioRecord sẽ khởi tạo bộ đệm âm thanh được lien kết với nó để lấp đầy dữ liệu âm thanh mới. Kích thước của bộ đệm này, được khởi tạo trong quá trình xây dựng, xác định thời gian mà AudioRecord có thể ghi lại trước khi “over-running” data chưa được đọc. Dữ liệu phải được đọc từ phần cứng âm thanh theo các kích cỡ nhỏ hơn tổng kích thước bộ đêm.

**2.Công nghệ nhận diện giọng nói Google Cloud Platform Speech To Text.**

**2.1 Giới thiệu**

- Sử dụng Google Cloud platform Speech to text API.

- Google Cloud platform Speech to text là các api được google phát triển sử dụng công nghệ Google’s machine learning . GCP hỗ trợ chuyển từ dạng audio sang dạng văn bản thông qua việc sử dụng các API đơn giản.

- Google Cloud platform Speech to text hỗ trợ hơn 120 tiếng khác nhau , trong đó bao gồm có Tiếng Việt.

- GCP Speech to text nhận diện giọng nói theo thời gian thực, giọng nói lập tức chuyển thành dạng văn bản ngay khi nhận diện đc thông qua streaming audio.

**2.2. Machine Learning và Deep Learning**

Machine learning gây nên cơn sốt công nghệ trên toàn thế giới trong vài năm nay. Trong giới học thuật, mỗi năm có hàng ngàn bài báo khoa học về đề tài này. Trong giới công nghiệp, từ các công ty lớn như Google, Facebook, Microsoft đến các công ty khởi nghiệp đều đầu tư vào machine

learning. Hàng loạt các ứng dụng sử dụng machine learning ra đời trên mọi linh vực của cuộc sống, từ khoa học máy tính đến những ngành ít liên quan hơn như vật lý, hóa học, y học, chính trị. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix. Hay AlphaGo, cỗ máy đánh cờ vây với khả năng tính toán trong một không gian có số lượng phần tử còn nhiều hơn số lượng hạt trong vũ trụ, tối ưu hơn bất kì đại kì thủ nào, là một trong rất nhiều ví dụ hùng hồn cho sự vượt trội của machine learning so với các phương pháp cổ điển.

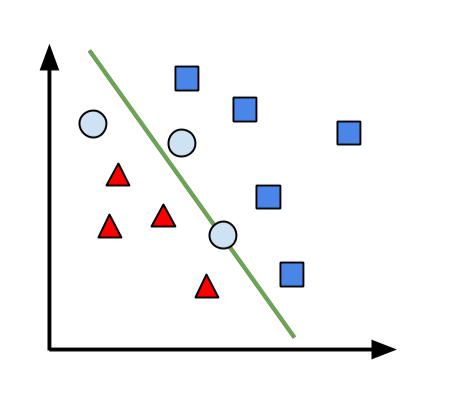
**2.2.1 Tìm hiểu Machine Learning**

Trong giới hàn lâm, theo hiểu biết chung, AI (Artificial Intelligence) là một ngành khoa học được sinh ra với mục đích làm cho máy tính có dược trí thông minh [**4**]. Mục tiêu này vẫn khá mơ hồ vì không phải ai cũng đồng ý với một định nghĩa thống nhất về trí thông minh. Các nhà khoa học phải định nghĩa một số mục tiêu cụ thể hơn, một trong số đó là việc làm cho máy tính lừa dược Turing Test. Turing Test dược tạo ra bởi Alan Turing (1912-1954), người dược xem là cha đẻ của ngành khoa học máy tính hiện đại, nhằm phân biệt xem người đối diện có phải là người hay không. Nếu trong quá trình trao đổi mà người đặt câu hỏi không thể phân biệt dược người hay máy trả lời các câu hỏi này thì máy dược xem là thông minh. Đến nay thì hướng tiếp cận này đã đạt được một số thành quả nhất định như:

* *Natural language processing:* máy có khả năng đọc hiểu và giao tiếp bằng ngônngữ tự nhiên với người.
* *Knowledge representation*: máy có khả năng lưu trữtri thức thông qua thịgiác,thính giác, hay văn bản.
* *Automated reasoning*: máy có khả năng sửdụng tri thức đã lưu trữ đểtrảlờicâu hỏi hay đưa ra kết luận hữu ích.
* ***Machine learning***: máy có khảnăng thích nghi với các điều kiện môi trườngxung quanh để rút trích ra các nguyên lý từ tri thức thu nhận được phục vụ cho việc ra quyết định.
* *Computer vision*: máy có khả năng quan sát và xác định được các đối tượngxung quanh.
* *Robotics:* máy có khả năng tương tác với đối tượng và di chuyển trong môitrường xung quanh.

Hiểu theo nghĩa đó AI thể hiện một mục tiêu của con người. Machine learning là một phương tiện được kỳ vọng sẽ giúp con người đạt được mục tiêu đó. Và thực tế thì machine learning đã mang nhân loại đi rất xa trên quãng đường chinh phục AI. Nhưng vẫn còn một quãng đường xa hơn rất nhiều cần phải đi. Machine learning và AI có mối quan hệ chặt chẽ với nhau nhưng không hẳn là trùng khớp vì một bên là mục tiêu (AI), một bên là phương tiện (machine learning). Chinh phục AI mặc dù vẫn là mục đích tối thượng của machine learning, nhưng hiện tại machine learning tập trung vào những mục tiêu ngắn hạn hơn như: làm cho máy tính có những khả năng nhận thức cơ bản của con người như nghe, nhìn, hiểu được ngôn ngữ, giải toán, lập trình, … hay hỗ trợ con người trong việc xử lý một khối lượng thông tin khổng lồ mà chúng ta phải đối mặt hàng ngày, hay còn gọi là Big Data.

Ngược dòng lịch sử, machine learning đã xuất hiện từ rất lâu trước khi mạng Internet ra đời. Một trong những thuật toán machine learning đầu tiên là thuật toán perceptron được phát minh ra bởi Frank Rosenblatt vào năm 1957. Đây là một thuật toán kinh điển dùng để phân loại hai khái niệm. Một ví dụ đơn gỉan là phân loại thư rác (tam gíac) và thư bình thường (vuông). Chắc các bạn sẽ khó hình dung ra được làm thế nào để làm được điều đó. Đối với perceptron, điều này không khác gì với việc vẽ một đường thẳng trên mặt phẳng để phân chia hai tập điểm:

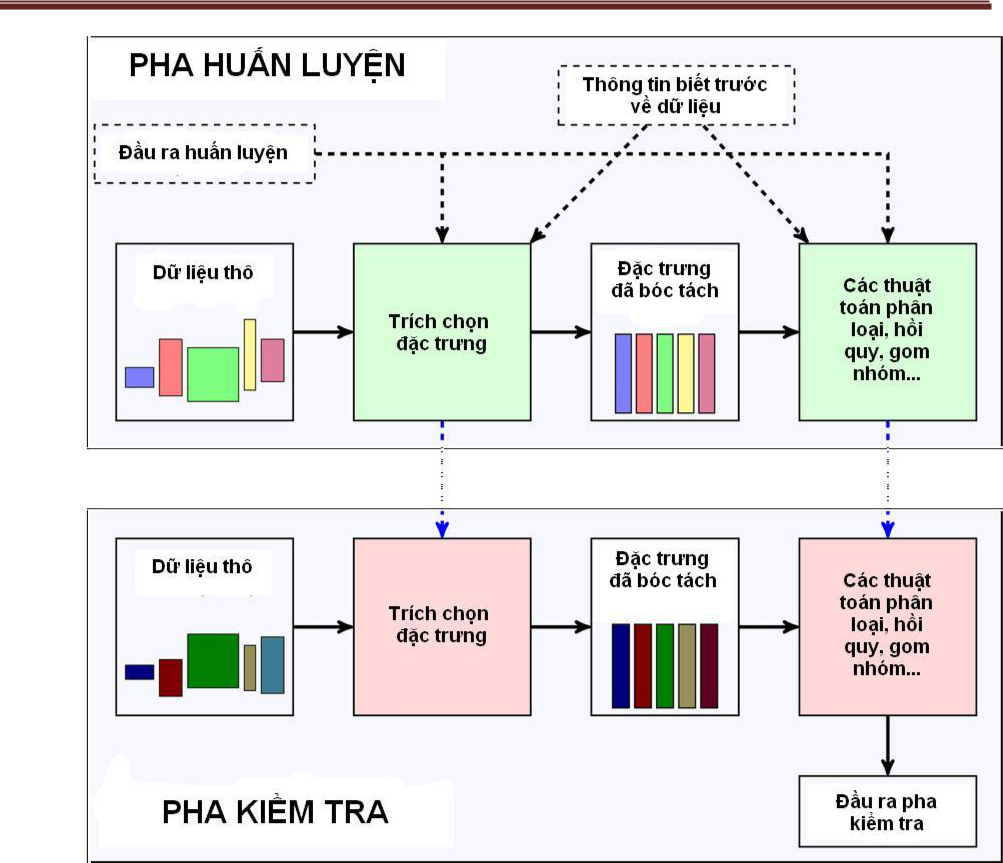


Những điểm tam giác và vuông đại diện cho những email chúng ta đã biết nhãn trước. Chúng được dùng để "huấn luyện" (train) perceptron. Sau khi vẽ đường thẳng chia hai tập điểm, ta nhận thêm các điểm chưa được dán nhãn, đại diện cho các email cần được phân loại (điểm tròn). Ta dán nhãn của một điểm theo nhãn của các điểm cùng nửa mặt phẳng với điểm đó.

Nhìn chung, bài toán ML luôn phải trải qua hai giai đoạn chính, đó là huấn luyện (training) và thử nghiệm (testing) trong mỗi giai đoạn đều sử dụng hai thành phần quan trọng nhất đó là trích chọn đặc trưng - Feature Engineering (hay còn gọi là Feature Extraction) và Thuật toán phân loại, nhận dạng… - Algorithms. Hai thành phần này có ảnh hưởng trực tiếp đến kết quả bài toán, vì thế được thiết kế rất cẩn thận, tốn nhiều thời gian, đòi hỏi người thiết kế phải có kiến thức chuyên môn và nắm rõ đặc điểm của bài toán cần xử lý.

Ta có thể quan sát mô hình hoạt động của phương pháp ML trên hình 2-21.

Công nghệ phát triển rất nhanh và ngày có càng nhiều thuật toán Machine Learning ra đời tuy nhiên đều thuộc một trong số các dạng sau đây.



Công nghệ phát triển rất nhanh và ngày có càng nhiều thuật toán Machine Learning ra đời tuy nhiên đều thuộc một trong số các dạng sau đây.

**2.2.2 TÌm hiểu Deep Learning**

Ta có thể thấy, ML chỉ là một nhánh nghiên cứu, một hướng nhỏ trong các hướng nghiên cứu AI. Và trong hướng nghiên cứu này, các nhà khoa học lại đề ra nhiều phương hướng giải thuật khác nhau như đã trình bày ở trên. Chúng ta có thể liệt kê cụ thể hơn một chút :

* *Supervised-learning*: decision tree, k-NN, naive bayes, SVM, neuralnetwork, deep learning, …
* *Unsupervised-learning*: k-means, hierachical clustering
* *Reinforcement learning*: passive/acive/generalization.

Xem lại hình 2-12 và một lần nữa ta có thể thấy, Deep Learning chỉ là một phương pháp trong nằm trong hướng giải quyết học có giám sát của ML.

Deep Learning ra đời phải kể đến sự ra đời của mạng nơ-ron nhân tạo ANN (Artificial Neural Networks). Neural Networks được lấy cảm hứng từ sự hiểu biết về sinh học của bộ não loài người – sự liên kết giữa các nơ-ron. Tuy nhiên, không giống như một bộ não sinh học nơi mà bất kỳ nơ-ron nào cũng có thể liên kết với các nơ-ron khác trong một khoảng cách vật lý nhất định, các mạng thần kinh nhân tạo này có các lớp rời rạc, các kết nối, và các hướng truyền dữ liệu phức tạp. DL sử dụng rất nhiều lớp nơ-ron nhân tạo để phân tích dữ liệu về nhiều chi tiết khác nhau.

Chẳng hạn nếu bạn dạy máy tính nhận diện hình ảnh một con mèo thì chúng ta sẽ lập trình ra nhiều lớp trong mạng thần kinh nhân tạo, mỗi lớp có khả năng xác định một đặc điểm cụ thể của con mèo như râu, vuốt, chân,… rồi cho máy xem hàng ngàn bức ảnh mèo (chỉ ra rằng “Đây là con mèo”) cùng hàng ngàn bức ảnh không phải mèo (chỉ ra rằng "đây không phải mèo"). Khi mạng thần kinh nhân tạo này xem hết các bức ảnh, các lớp node của nó sẽ dần nhận ra râu, vuốt, chân,..., biết lớp nào là quan trọng, lớp nào không. Nó cũng sẽ nhận ra rằng mèo luôn có chân nhưng những con vật không phải mèo cũng có chân nên khi cần xác định mèo, chúng sẽ tìm chân đi kèm những đặc điểm khác như vuốt hay râu.

Như vậy là, thay vì tìm các đặc trưng trước để huấn luyện và nhận dạng. DL mang đến khả năng tự rút ra đặc trưng của đối tượng được gán nhãn.

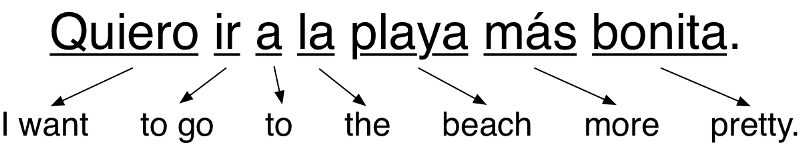
Cụ thể, với một mạng Học sâu cho nhận dạng ảnh, các lớp đầu tiên trong mạng chỉ làm nhiệm vụ rất đơn giản là tìm kiếm các đường thẳng, đường cong, hoặc đốm màu trong ảnh đầu vào. Các thông tin này sẽ được sử dụng làm đầu vào cho các lớp tiếp theo, với nhiệm vụ khó hơn là từ các đường, các cạnh đó tìm ra các thành phần của vật thể trong ảnh. Cuối cùng, các lớp cao nhất trong mạng huấn luyện sẽ nhận nhiệm vụ phát hiện ra vật thể trong ảnh.

Với cách thức học thông tin từ ảnh lần lượt qua rất nhiều lớp, nhiều tầng khác nhau như vậy, các phương pháp này có thể giúp cho máy tính hiểu được những dữ liệu phức tạp bằng nhiều lớp thông tin đơn giản qua từng bước phân tích. Đó cũng là lý do chúng được gọi là các phương pháp DL – nghĩa tiếng việt là “Học sâu”.

Tuy có nhiều điểm ưu việt trong khả năng huấn luyện máy tính cho các bài toán phức tạp, Học sâu vẫn còn rất nhiều giới hạn khiến nó chưa thể được áp dụng vào giải quyết mọi vấn đề. Điểm hạn chế lớn nhất của phương pháp này là yêu cầu về kích thước dữ liệu huấn luyện, mô hình huấn luyện Học sâu đòi hỏi phải có một lượng khổng lồ dữ liệu đầu vào để có thể thực hiện việc học qua nhiều lớp với một số lượng lớn nơ-ron và tham số. Đồng thời, việc tính toán trên quy mô dữ liệu và tham số lớn như vậy cũng yêu cầu đến sức mạnh xử lý của các máy tính server cỡ lớn. Quy trình chọn lọc dữ liệu cũng như huấn luyện mô hình đều tốn nhiều thời gian và công sức, dẫn đến việc thử nghiệm các tham số mới cho mô hình là công việc xa xỉ, khó thực hiện.

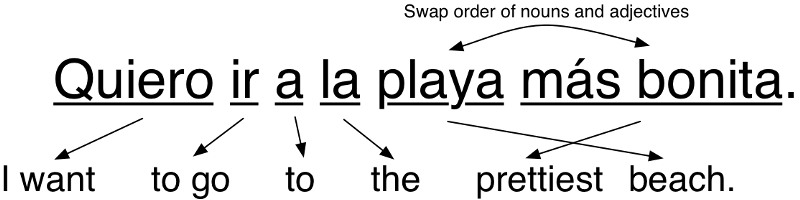
**2.3. Làm thế nào để chương trình máy tính dịch được ngôn ngữ con người.**

Cách đơn giản nhất là thay thế từng từ trong một câu với từ được dịch ở ngôn ngữ hướng tới. Đây là giải thích đơn giản dịch từ tiếng Tây Ba Nha (TBN) sang tiếng Anh từng từ một:



Điều này rất dễ để triển khai, bởi vì tất cả những gì bạn cần là một cuốn từ điển để tìm nghĩa từng từ. Nhưng kết quả sẽ tồi tệ, bởi nó bỏ qua ngữ pháp và văn cảnh.

Điều tiếp theo chúng ta có thể thử là bắt đầu thêm những luật ngôn ngữ để cải thiện kết quả. Ví dụ, bạn có thể dịch cụm 2 từ liên quan vào một nhóm. Và bạn hoán đổi vị trí danh từ với tính từ vì trong tiếng Anh, chúng luôn xuất hiện ở vị trí ngược lại so với tiếng Tây Ban Nha:



Câu đã chính xác! Nếu chúng ta cứ tiếp tục thêm thật nhiều luật lệ cho tới khi bao hàm được tất cả phần của ngữ pháp, chương trình của chúng ta có thể dịch bất cứ câu nào.

Đây chính là hệ thống dịch thuật đầu tiên được phát triển. Những nhà ngôn ngữ học đưa ra những luật lệ phức tạp và lập trình chúng từng dòng một. Một số nhà ngôn ngữ học thông thái nhất đã làm việc ròng rã nhiều năm trong suốt chiến tranh lạnh để tạo ra hệ thống dịch, như là một cách để hiểu tiếng Nga dễ dàng hơn.

Thật không may, hệ thống chỉ hoạt động tốt cho các văn bản có cấu trúc đơn giản như dự báo thời tiết. Nó không đáng tin cậy trong văn bản đời thực.

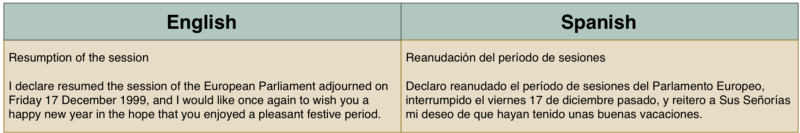
Vấn đề là ngôn ngữ con người không tuân theo những luật lệ cố định. Ngôn ngữ chứa nhiều trường hợp đặc biệt, biến thể vùng miền hay các luật bị phá vỡ.

## Sử dụng thống kê giúp máy tính dịch tốt hơn

Sau hàng loạt thất bại của hệ thống luật lệ, phương pháp dịch mới được phát triển dựa trên mô hình xác suất thống kê hơn là những luật lệ ngữ pháp.

Xây dựng hệ thống dịch dựa trên thống kê yêu cầu rất nhiều dữ liệu đã được dịch sang cả 2 ngôn ngữ. Tài liệu 2 ngôn ngữ này được gọi là **parallel corpora** (tài liệu song ngữ).

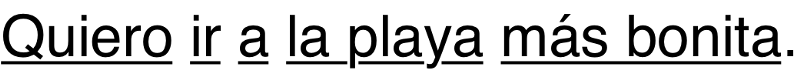
Thật may mắn, có rất nhiều tài liệu song ngữ tồn tại ở nhiều nơi. Ví dụ, Nghị Viện Châu Âu đã dịch tài liệu của họ sang 21 ngôn ngữ. Vì thế, những nhà nghiên cứu sử dụng tư liệu đó để xây dựng hệ thống dịch.



## Nghĩ về xác suất

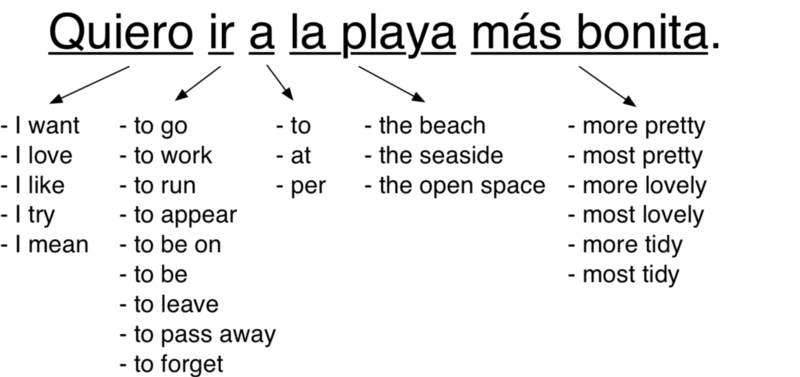
Điều đặc biệt của những hệ thống dịch xác suất là chúng không cố gắng tạo ra đoạn dịch chính xác. Thay vào đó, chúng tạo ra hàng ngàn khả năng dịch và sắp xếp chúng theo khả năng chính xác. Độ chính xác được ước lượng bằng cách so sánh với dữ liệu đào tạo. Dưới đây là cách thức hoạt động:

### Bước 1: Tách câu ra thành từng phần



### Bước 2: Tìm tất cả các cách dịch cho mỗi phần

Chúng ta dịch từng phần bằng cách tìm tất cả các cách đã được dịch trong tập đào tạo. Chú ý là chúng ta không chỉ tìm kiếm cách dịch trong từ điển, mà xem xem những người khác đã dịch phần đó như thế nào. Điều này giúp ta thu thập được các cách khác nhau mà từ ngữ được sử dụng ở văn cảnh khác nhau:



Một vài cách dịch được sử dụng thường xuyên hơn cách khác. Dựa vào tần suất mỗi cách dịch trong tập dữ liệu, chúng ta gán cho chúng một điểm số.

Ví dụ, chúng ta thường hiểu "Quiero" là "I want" - tôi muốn hơn là "I try" - tôi cố gắng. Vì thế, chúng ta dựa vào tần suất số lần dịch "Quiero" là "I want" trong tập dữ liệu để gán điểm số lớn hơn so với các cách dịch khác.

### Bước 3: Tạo ra tất cả câu có thể và tìm câu có khả năng nhất

Tiếp đến, ta tìm tất cả các cách kết nối từng phần và tạo ra một loạt các câu có thể.

Chỉ từ việc dịch từng phần, chúng ta có thể tạo ra hơn 2500 biến thể bằng cách kết nối chúng lại. Dưới đây là một vài ví dụ:

I love | to leave | at | the seaside | more tidy. I mean | to be on | to | the open space | most lovely. I like | to be |on | per the seaside | more lovely. I mean | to go | to | the open space | most tidy.

Nhưng trong hệ thống thực tế, thậm chí còn có nhiều cách kết nối hơn, bởi chúng ta thử các cách sắp xếp khác nhau của từ và các cách tách câu thành từng phần khác nhau.

I try | to run | at | the prettiest | open space. I want | to run | per | the more tidy | open space. I mean | to forget | at | the tidiest | beach. I try | to go | per | the more tidy | seaside.

Bây giờ, ta cần phải duyệt qua tất cả các câu được tạo để tìm ra câu phù hợp nhất. Để làm điều đó, chúng ta so sánh những câu được tạo với hàng triệu câu thực tế trong các văn bản được viết bằng tiếng Anh. Càng nhiều tài liệu, kết quả càng tốt.

Lấy một khả năng dịch:

I try | to leave | per | the most lovely | open space.

Có khả năng là chưa ai từng viết câu như thế cả. Do đó, câu trên sẽ không giống với bất kỳ câu nào trong tập dữ liệu, ta sẽ gán giá trị rất nhỏ cho câu trên.

Nhưng hãy nhìn một khả năng khác:

I want | to go | to | the prettiest | beach.

Câu này giống với nhiều câu trong tập đào tạo, vì thế nó sẽ có điểm số cao.

Sau khi thử tất cả các khả năng, chúng ta chọn câu có điểm số cao nhất. Kết quả dịch cuối cùng của chúng ta là “I want to go to the prettiest beach" - Rất chuẩn!!!

### Dịch thuật dựa trên xác suất là một bước tiến lớn

Nếu bạn cung cấp đủ dữ liệu, hệ thống xác suất thường dịch tốt hơn so với hệ thống dựa trên luật lệ. Franz Josef Och cải tiến ý tưởng này và sử dụng để xây dựng Google dịch đầu những năm 2000s. Dịch Máy cuối cùng cũng ra đời.

Ở thời kì đầu, điều ngạc nhiên tất cả mọi người là phương pháp "còi" dựa trên xác suất lại có thể đánh bại hệ thống luật lệ tạo ra bởi những nhà ngôn ngữ học.

### Giới hạn của hệ thống Dịch Máy dựa trên Xác Suất

Hệ thống Dịch Máy Xác Suất hoạt động tốt, nhưng thường phức tạp để xây dựng hay bảo trì. Mỗi cặp ngôn ngữ bạn muốn dịch cần rất nhiều chuyên gia để điều chỉnh trọng số trong chuỗi dịch thuật gồm rất nhiều bước.

Bởi vì quá trình này rất tốn tài nguyên, nhiều thay thế đã được thực hiện. Nếu bạn yêu cầu Google dịch tiếng Việt sang tiếng Thái, nó thường dịch sang tiếng Anh như cầu nối, bởi vì không đủ tài liệu song ngữ, hoặc nó có thể dịch với hệ thống không tân tiến và chưa có đầy đủ các trường hợp như nếu GG được yêu cầu dịch từ tiếng Pháp sang tiếng Anh.

Chẳng phải sẽ tuyệt với hơn, nếu máy tính làm toàn bộ quá trình phát triển cho chúng ta?

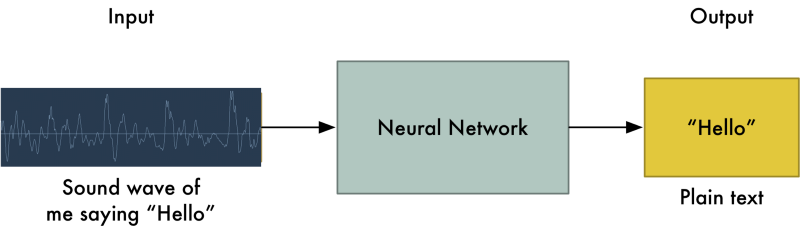
## Dịch tốt hơn, và tiết kiệm nguồn lực hơn

Cốt lõi của dịch máy là một hệ thống hộp đen có thể tự động dịch, chỉ bằng việc nhìn vào dữ liệu. Với Dịch Máy Xác Suất, con người vẫn cần xây dựng và tìm kiếm trọng số trong mô hình xác suất nhiều bước.

Vào năm 2014, nhóm của KyungHuyn Cho đã tạo ra một bước đột phá. Họ tìm ra cách áp dụng **Deep Learning** để tìm ra hệ thống hộp đen. Mô hình này lấy tài liệu dịch 2 chiều và sử dụng chúng để học cách dịch giữa 2 ngôn ngữ mà không cần sự can thiệp của con người.

**2.4 Áp dụng Machine Learning vào nhận diện giọng nói**

Theo như trên, bạn có thể đoán rằng: chúng ta chỉ cần truyền đoạn ghi âm vào mạng nơron và đào tạo nó để tạo ra "bản dịch"



Đó cũng là điều mà nhận diện giọng nói với deep learning hướng tới, nhưng chúng ta chưa đạt đến trình độ đó (có thể cần phải thêm vài năm nữa).

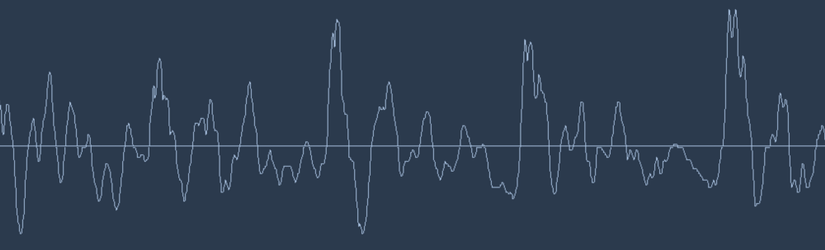
Vấn đề lớn nhất chính là **tốc độ nói biến thiên**. Một người có thể nói "Hello" rất nhanh và người khác nói "heeeellllllloooooo!" cực chậm, tạo ra âm thanh dài hơn với nhiều dữ liệu hơn. Cả 2 âm đều nên được nhận dạng chính xác là từ một - "hello!". Tự động chỉnh file âm thanh với nhiều biến thể độ dài khác nhau của từng từ để tạo ra văn bản đồng nhất lại khá khó.

Để xử lý vấn đề này, chúng ta sẽ sử dụng một số kỹ thuật đặc biệt và thêm một vài bước vào mạng deep learning.

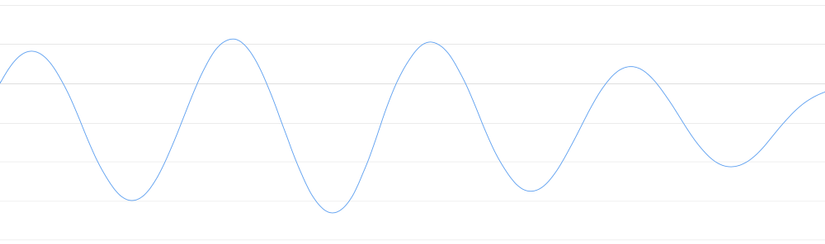
## Chuyển âm thanh thành số

Bước đầu tiên trong nhận diện giọng nói khá rõ ràng - chung ta cần truyền sóng âm vào máy tính.

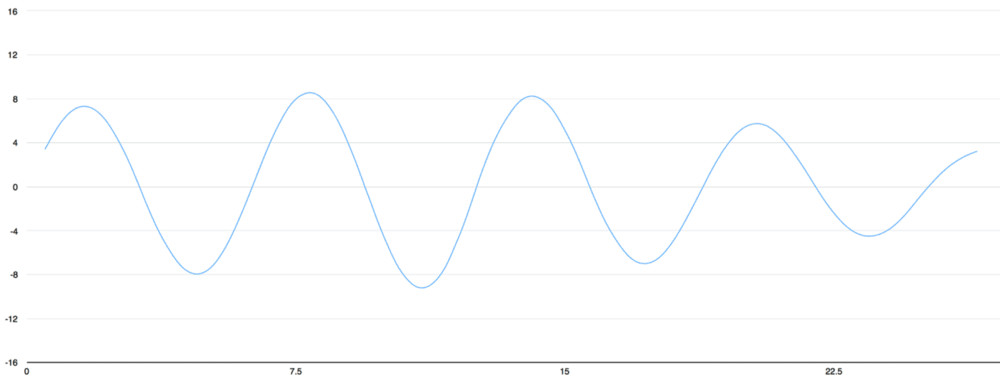
Âm thanh được truyền qua sóng âm. Làm thế nào chúng ta chuyển sóng âm thành số? Hãy sử dụng đoạn âm nói "Hello" dưới đây:



 Sóng âm có **một chiều** dữ liệu. Ở mỗi thời điểm, chúng có một giá trị cao độ. Hãy phóng to một đoạn nhỏ sóng âm để nhìn rõ hơn:



 Để chuyển sóng âm thành số, chúng ta chỉ cần ghi lại độ cao của sóng ở từng khoảng:



Phương pháp này gọi là **sampling** - lấy mẫu. Chúng ta đọc mẫu mỗi 1/1000s và ghi lại con số đại diện chiều cao cùa sóng âm. Đây chính là file .wav khi không bị nén.

Những âm thanh chất lượng tốt được ghi ở tần số 44.1khz (44,100 lần đọc mỗi giây). Nhưng với nhận diện giọng nói, tốc độ lấy mẫu ở 16khz (16,000 mẫu mỗi giây) là quá đủ.

Thử lấy mẫu giọng nói "Hello" trong ví dụ trên với tần suất 16,000 lần mỗi giây. Và đây là 100 mẫu đầu tiên.

MAC OS:Users:macos:Desktop:276438ac-9567-40f2-883b-25da7b2334e0.png

### Lấy mẫu liệu có được chính xác?

Bạn có thể nghĩ rằng, lấy mẫu chỉ tạo ra đồ thị xấp xỉ so với sóng âm, bởi vì nó chỉ đọc dữ liệu theo từng khoảng. Liệu chúng ta có bị mất dữ liệu giữa mỗi lần đọc?

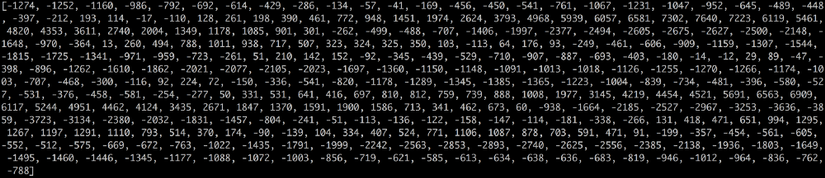
 Nhờ có lý thuyết Nyquist, chúng ta có thể sử dụng toán học để tái tạo chính xác sóng âm gốc từ những mẫu tách biệt - miễn là chúng ta lấy mẫu với tần số **gấp đôi** tần số âm chúng ta muốn ghi lại. Không phải cứ lấy mẫu với tần số càng cao thì chất lượng âm thanh càng tốt !

## Tiền xử lý dữ liệu mẫu âm thanh

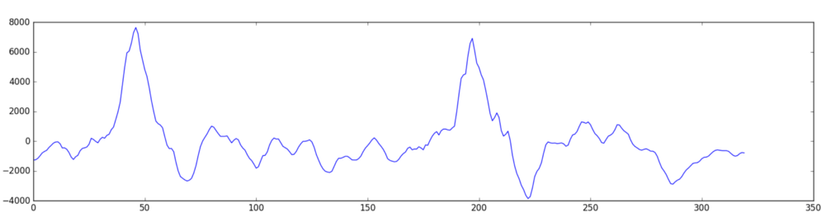
Chúng ta đã có dãy số với mỗi số đại diện cho cao độ âm tại 1/16,000s.

Ta có thể truyền những số này vào mạng nơron, nhưng cố gắng nhận diện cấu trúc âm thanh trực tiếp bằng những mẫu này rất khó. Thay vào đó, chúng ta giải quyết vấn đề dễ hơn bằng cách **tiền xử lý** dữ liệu.

Hãy bắt đầu nhóm mẫu âm thanh trong khoảng 20ms. Và đây là 320 mẫu âm thanh trong 20ms đó:



Ghi lại những con số này trong đồ thị giúp chúng ta có ước lượng xấp xỉ về âm thanh gốc trong chu kỳ 20ms:



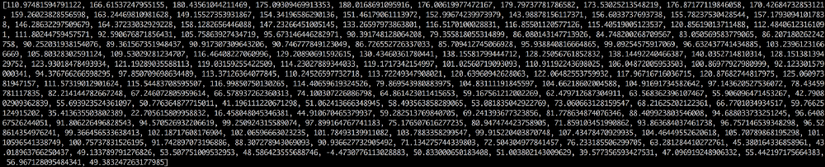
Bản ghi âm này chỉ khoảng 1/50s. Nhưng thậm chí một đoạn ghi âm rất ngắn là một mớ hỗn độn cao độ âm khác nhau. Có những âm thấp, âm trung và thậm chí cả âm cao. Nhưng cùng với nhau, những âm này tạo lên giọng nói.

Để giúp mạng nơron xử lý dữ liệu dễ hơn, ta tách sóng âm phức tạp này thành từng phần: phần chứa âm thấp, âm cao hơn, cao hơn nữa... Sau đó, ta tính tổng năng lượng ở những những dải tần số (từ thấp đến cao) và kết nối lại tạo ra **fingerprint** - nhận dạng duy nhất cho từng đoạn trích âm thanh.

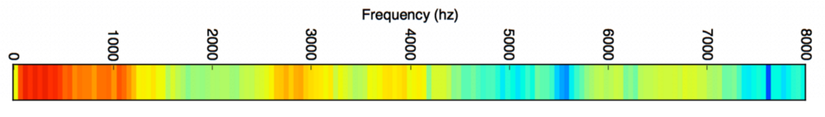
Chúng ta làm điều đó nhờ vào việc sử dụng Fourier transform trong toán học. Nó chia nhỏ những sóng âm phức tạp thành sóng âm đơn tạo ra nó, và ta có thể tính tổng năng lượng ở mỗi đơn âm.

Sau khi sử dụng lý thuyết Nyquist ở trên, sóng âm đã trở thành một dải liên tục. Và sử dụng Fourier transform, chúng ta lại tách dải liên tục đó ra thành các notes riêng biệt (được tính toán bởi thuật toán Fourier) để tìm ra tổng năng lượng ở từng note.

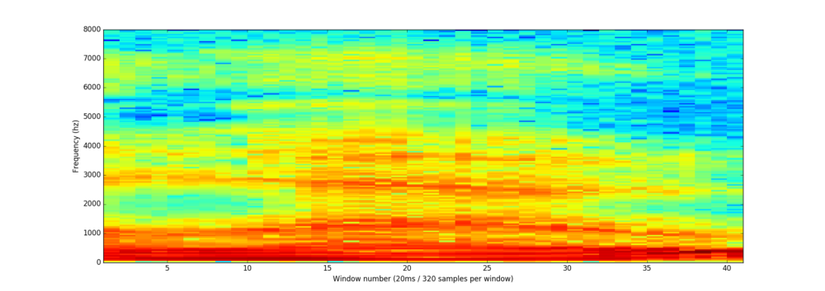
Kết quả cuối cùng là một bảng số thể hiện độ năng lượng của mỗi khoảng tần số, từ âm thấp tới âm cao. Mỗi số dưới đây đại diện cho năng lượng dải 50hz trong clip 20ms:



Nhưng sẽ dễ dàng hơn nhiều khi bạn biểu diễn dãy số trên trên đồ thị:



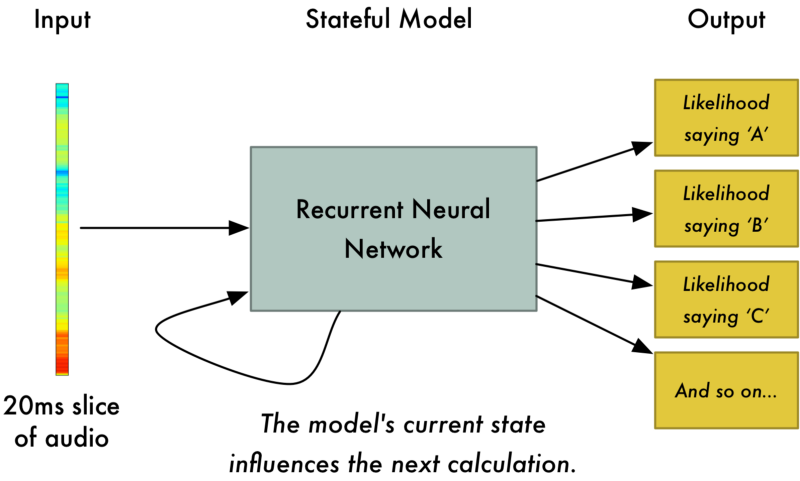
Nếu chúng ta lặp lại quá trình trên cho mỗi khoảng 20ms, chúng ta sẽ có quang phổ (mỗi cột từ trái qua phải là một khoảng 20ms):



Tạo ra quang phổ giúp chúng ta thực sự nhìn thấy âm thanh và cấu trúc độ cao của nó. Mạng nơron có thể tìm những cấu trúc trong dữ liệu này dễ dàng hơn so với sóng âm thô. Do đó, đây chính là đặc trưng mà ta truyền vào mạng nơron..

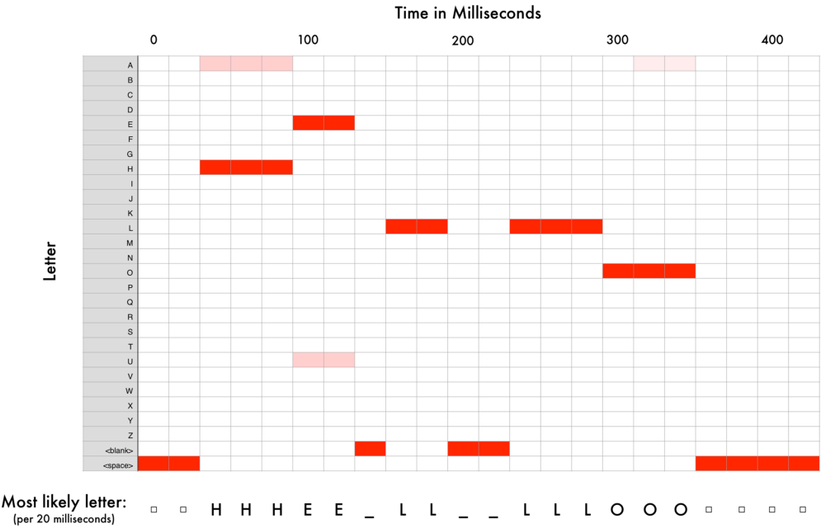
## Nhận diện ký tự từ đoạn âm ngắn

Bây giờ chúng ta truyền từng dải âm 20ms vào mạng nơron đa lớp. Với mỗi mảng cắt âm thanh, chúng ta cố gắng tìm ra ký tự đại diện cho âm thanh phát ra.



 Chúng ta sử dụng Recurrent Neural Network – RNN - mạng nơron hồi quy: kết quả tiên đoán quá khứ có ảnh hưởng tới kết quả tiên đoán trong tương lai. Đó là bởi vì các ký tự có sự liên quan đến nhau. Ví dụ chúng ta đã tìm ra "HEL", thì rất có khả năng chúng ta sẽ nói tiếp "LO". Vì thế, những dự đoán trong quá khứ sẽ giúp dự đoán tương lai được tốt hơn.

Sau khi chạy toàn bộ âm thanh thông qua mạng nơron, chúng ta kết nối mỗi dải âm với một ký tự có khả năng được nói cao nhất. Và đây là bản đồ kết nối của từ "HELLO":



Mạng nơron trên dự đoán từ được nói là “HHHEE\_LL\_LLLOOO”, nhưng nó cũng nghĩ có khả năng từ đó là “HHHUU\_LL\_LLLOOO”, hoặc thậm chí là “AAAUU\_LL\_LLLOOO”

Chúng ta có thêm một vài bước để làm sạch kết quả. Đầu tiên, chúng ta bỏ đi những ký tự bị lặp, rồi bỏ đi khoảng trống:

* HHHEE\_LL\_LLLOOO => HE\_L\_LO => HELLO
* HHHUU\_LL\_LLLOOO => HU\_L\_LO => HULLO
* AAAUU\_LL\_LLLOOO => AU\_L\_LO => AULLO

Như vậy, ta có 3 khả năng phân âm là "Hello", "Hullo" và "Aullo". Nếu bạn nói chúng thật to, cả 3 đều nghe giống với "Hello". Bởi vì dự đoán từng ký tự một, mạng nơron tìm ra cách đọc các âm chứ không phải cách viết. Ví dụ: nếu bạn nói "He would not go", máy có thể dịch là "He wud net go".

Thủ thuật ở đây là kết hợp những dự đoán phiên âm này với khả năng xuất hiện trong các văn bản (sách, bài bảo...). Bạn sẽ loại bỏ đi những phiên âm ít có khả năng ngoài thực tế và giữ phiên âm thực tế nhất. Và trong 3 từ "Hello", Hullo" và "Aullo". Rõ ràng, "Hello" có tần xuất cao hơn rất rất nhiều, và đây chính là bản phiên âm chúng ta lựa chọn.

**2.3 Sử dụng Google Cloud Speech To Text API**

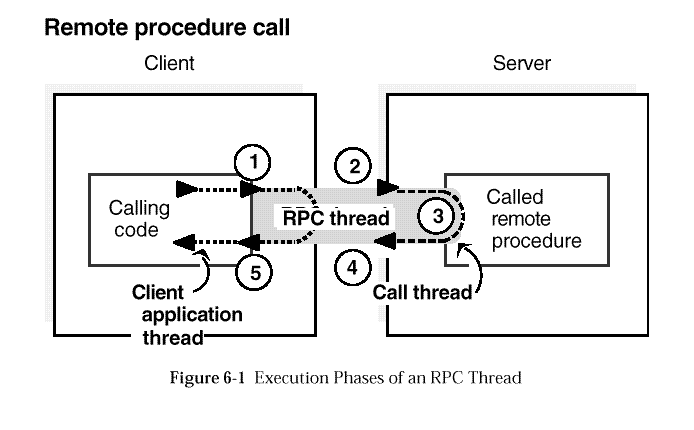
**2.3.1 GRPC ( Google Remote Produre Call)**

**gRPC**là một RPC platform được phát triển bởi Google nhằm tối ưu hoá và tăng tốc việc giao tiếp giữa các service với nhau trong kiến trúc microservice.gRPC là sự kết hợp của Remote Produre Call và http2, Protocol Buffers được phát triển bởi google . **gRPC**dùng Protocal Buffer giảm kích thước request và response data, RPC để đơn giản hoá trong việc tạo ra các giao tiếp giữa các service với nhau, HTTP/2 để tăng tốc gửi/nhận HTTP request. Nó nhẹ hơn, nhanh hơn và cung cấp hiệu năng tốt hơn so với sử dụng XML hoặc Json gRPC cũng cho phép định nghĩa cấu trúc của data dưới dạng file protoc và nó tự động generate ra file sử dụng để giao tiếp với ngôn ngữ mà bạn sử dụng. gRPC hiện tại cũng đã hỗ trợ khá đầy đủ các ngôn ngữ như C++, Java, Python, Go, ...

Vậy, RPC, Protocol Buffers và http2 là gì. là gì ?

**2.3.1.1 RPC (Remote Produre Call)**

**Remote Procedure Call (RPC) tạm dịch là các cuộc gọi thủ tục từ xa là một phương pháp dùng để trao đổi dữ liệu. RPC khiến cho việc thực hiện IPC dễ dàng, giống như một lời gọi hàm bình thường.** RPC có thể được thực hiện giữa hai tiến triền trên cùng một máy tính, hoặc giữa các máy tính khác nhau trong mạng.

**Remote Procedure Call (RPC) là một khái niệm nhằm cố gắng khái quát một lời gọi thủ tục thông thường trong trường hợp mà caller và receiver không cùng nằm trong một process – và được phân tán trên các máy riêng biệt.**

Mục tiêu chính của phương pháp này là làm cho lời gọi từ xa RPC tương tự như thể lời gọi thủ tục thông thường cục bộ và ẩn đi việc truyền dữ liệu đi về qua mạng.

Mục tiêu này rất hấp dẫn ở chỗ nó có khả năng cho phép chuyển sự phân tán của hệ thống cuối cùng vào một quyết định ở thời điểm triển khai – hay nói cách khác, từ quan điểm của lập trình viên thì không quan trọng việc cuộc gọi đó là cục bộ hay từ xa miễn là nó có cú pháp giống nhau, và quyết định cuối cùng về sự phân tán của các thành phần hệ thống riêng lẻ có thể được thực hiện sau này. Việc loại bỏ khía cạnh phân tán từ code có thể mang lại rất nhiều lợi ích cho các dự án vì ở giai đoạn đầu thì các chi tiết cuối cùng của việc triển khai thường chưa được biết đầy đủ. Lập trình viên có thể tùy biến chuyển từ lời gọi cục bộ sang lời gọi từ xa RPC mà không thay đổi quá lớn cấu trúc ban đầu của chương trình.

**Tuy nhiên, những lợi ích tiềm năng của RPC cũng có cái giá của nó:**

* Trong cơ chế gọi hàm trong nội bộ một process, lập trình viên thường không quan tâm đến thời gian chuyển thực thi từ đối tượng gọi (caller) vào đối tượng bị gọi (receiver). Thời gian này rất ngắn, chương trình nạp biến cục bộ của caller vào stack. Ngược lại trong mô hình RPC thực tế, khoảng thời gian truyền tham số đối tượng gọi (caller) đến đối tượng bị gọi (receiver) ở xa, rồi kết quả trả về sẽ đi từ receiver về tới caller là không nhỏ. Lập trình viên đành phải chấp nhận hoặc lờ đi hoặc đặt cơ chế hết hạn (time out). Cách lập trình viên tối ưu chương trình cục bộ là chia nhỏ chương trình thành nhiều đối tượng gọi nhau, hoặc các hàm tái sử dụng được gọi lại. Tuy nhiên với RPC, cách chia nhiều hàm để gọi này chưa chắc hợp lý khi thời gian trễ mỗi lần gọi RPC là khó có thể bỏ quả, càng nhiều lần gọi, tổng thời gian trễ sẽ tăng, khả năng nghẽn cổ chai do kiểu hỏi đáp liên tục sẽ tăng. Nhiều nơi gọi là chatty ~ gọi vặt liên tục.
* Đối với lời gọi cục bộ, đối tượng gọi (caller) và đối tượng bị gọi (receiver) nằm trong cùng một process. Kiểu tham số truyền được kiểm tra nghiêm ngặt khi biên dịch. Việc viết unit test để kiểm tra cũng đơn giản. Tuy nhiên với lời gọi RPC, tham số gửi đi, dữ liệu kết quả trả về sẽ phải quá trình chuyển đổi: serialize và de-serialize, hay còn gọi là marshalling. Đối với lời gọi RPC, việc kiểm tra kiểu có nhiều rủi ro hơn nhiều, chưa kể rủi ro dữ liệu bị nghe lén hoặc bị thay đổi trên đường truyền. Việc bảo mật lời gọi RPC dẫn đến cần phải mã hóa, gắn kèm chữ ký kiểm tra… khiến thư viện bên dưới của caller và receiver sẽ phải làm việc nhiều hơn, độ trễ lại cao hơn. Chưa kể đồng hồ thời gian ở máy tính chứa caller và receiver có thể sai khác nhau, hệ điều hành cũng như phần mềm, ngôn ngữ lập trình cũng khác nhau, kiểu dữ liệu có sự sai khác…

Vấn đề của RPC là bởi nó ẩn đi thực tế phân tán ở mức cú pháp, điều này gây khó khăn hơn cho các lập trình viên để giải quyết đúng đắn các thách thức cố hữu đi kèm với các khía cạnh vật lý của phân tán.

2.3.1.2 Protocol Buffer

Protocol buffer còn được biết như protobuf là language-neutral, platform-neutral của google phiên bản nội bộ được công bố vào năm 2001 và phiên bản công khai đầu tiên được giới thiệu vào năm 2008 ,về cơ bản nó được sủ dụng để Serialized object, có vẻ nó khá giống XML hoặc JSON. Nó lưu trữ dữ liệu có cấu trúc có thể được Serialize hoặc De-Serialized tự động bưởi nhiều ngôn ngữ khác nhau. Nó được thiết kế để trở thành language/platform neutral và có thể mở rộng. Hiện tại, protobuf có hỗ trợ cho C ++, C, Go, Java và Python.

Protobuf là một open source dùng để encode dữ liệu có cấu trúc được phát triển tại google. Nó rất hữu ích trong việc phát triển các chương trình để giao tiếp với nhau qua một wire hoặc để lưu trữ dữ liệu. Tất cả những gì bạn phải làm là chỉ định một thông báo cho từng cấu trúc dữ liệu mà bạn muốn Serialize (theo định dạng giống như lớp Java) bằng cách sử dụng file đặc tả .proto.

Từ file .proto compiler của protobuf ( protoc ) tạo ra code thực hiện encode tự động và phân tích cú pháp dữ liệu protobuf với định dạng Binary hiệu quả, tùy thuộc vào từng ngôn ngữ nó sẽ tạo ra mã tương ứng .

# Protobuf so sánh một số định dạng kiểu dữ liệu khác.

Dưới đây chúng ta sẽ đề cập đến ưu và nhược điểm của Protocol buffer và một số kiểu khác.

### Protobuf

1. Dữ liệu rất dày đặc, đầu ra nhỏ.
2. Khó decode mà không biết schema, định dạng dữ liệu không rõ ràng và cần schema để biết rõ.
3. Xử lý rất nhanh, nhỏ hơn 3 - 10 lần so với XML hoặc JSON
4. không dành cho con người vì là Binary.
5. Tạo các code truy cập dữ liệu dễ sử dụng hơn theo chương trình.

### JSON

1. Con người có thể có thể đọc và chỉnh sửa dễ dàng.
2. Có thể phân tích cú pháp mà không cần biết schema.
3. Các browser đều hỗ trợ rất tốt.
4. Ít dài dòng hơn XML.

### XML

1. Con người có thể có thể đọc và chỉnh sửa dễ dàng.
2. Có thể phân tích cú pháp mà không cần biết schema.
3. Tiêu chuẩn cho SOAP...
4. Hỗ trợ tốt các công cụ như xsd, xslt, sax, dom ...

Protobuf rất nhanh nhưng nhưng nhưng có những những tình huống chúng ta không nên sử dụng nó. Ví dụ như các tình huống dưới đây

1. Khi bạn cần hoặc muốn dữ liệu con người có thể đọc dễ dàng.
2. Dữ liệu từ Service được sử dụng trực tiếp bởi Browser.
3. Server của bạn viết bằng ngôn ngư khác như Javascript
4. Gánh nặng hoạt động của việc vận hành một loại dịch vụ mạng khác là quá lớn

**2.3.1.3 Http/2**

HTTP/1.1 được phát triển cách đây quá lâu nên nó bắt đầu bộc lộ những nhược điểm của sự "già cỗi" đó. Các trang web ngày nay không chỉ có những đoạn mã HTML đơn giản, ngoài ra nó còn được kết hợp với các công cụ dùng cho việc trang trí - thiết kế (CSS), các đoạn mã thực thi (JavaScript), hình ảnh, video, thậm chí là cả nội dung Flash nữa. Để truyền tải lượng dữ liệu đó, trình duyệt phải tạo ra nhiều kết nối đến máy chủ, mỗi kết nối chứa rất nhiều thông tin về nguồn lấy là ai, lấy ở đâu, lấy nội dung, lấy ra sao gì và rất nhiều thứ khác nữa. Tất cả những thứ này đã tạo ra một khối lượng công việc rất lớn lên trình duyệt trong máy tính (hoặc thiết bị di động) của chúng ta cũng như máy chủ nơi đặt trang web.

HTTP/2 mang lại một số lợi ích như giúp trang web tải nhanh hơn, các kết nối có thời gian “sống” dài hơn, nội dung xuất hiện nhanh hơn, hỗ trợ nhiều kết nối song song. Ngoài ra, các yêu cầu HTTP do trang web gửi đến máy chủ cũng sẽ nhẹ hơn nên rất nhiều yêu cầu có thể được thực hiện cùng lúc, hạn chế tình trạng nghẽn hoặc từ chối truy cập.

Nói chung, việc sử dụng htttp2 đơn giản để phục vụ mục đích tăng tốc độ truy cập, tải web. Trong một số thử nghiệm của HTTPWatch, http/2 cải thiện tốc độ so với http lên đến 30 %. HTTP/2 cải thiện tốc độ chủ yếu bằng cách tạo ra một kết nối liên tục giữa máy chủ và trình duyệt, thay vì thiết lập một kết nối mới mỗi lần trình duyệt cần thông tin gì đó. Cách thức này giảmđi đáng kể lượng thông tin được truyền tải. Ngoài ra, HTTP/2 còn truyền dữ liệu ở dạng nhị phân thay vì dạng văn bản như trước. Điều này có nghĩa là trình duyệt của bạn không phải tốn thời gian dịch văn bản sang một định dạng nào đó mà nó có thể hiểu được.

Một vài kĩ thuật khác mà HTTP/2 sở hữu để tăng tốc bao gồm multiplexing (gửi nhận nhiều thông tin cùng lúc), prioritization (đánh thứ tự ưu tiên, dữ liệu quan trọng sẽ được gửi đi trước), compression (nén để thu nhỏ kích cỡ gói dữ liệu) và server push (máy chủ sẽ đoán yêu cầu dữ liệu kế tiếp mà bạn cần là gì để mà gửi trước thông tin cho đỡ tốn thời gian).

Tóm lại, ở cấp độ cao, HTTP/2 có các đặc điểm:

• Nhị phân

• Đa kết nối

• Sử dụng một kết nối cho trạng thái song song

• Sử dụng nén header để giảm tải cho máy chủ

Hiện nay, thời gian bắt đầu sử dụng HTTP/2 vẫn chưa thống nhât và cụ thể, cũng có thể chúng ta đang sử dụng mà không biết vì nó đã được tích hợp trên một số máy chủ và trình duyệt như Chrome và Firefox (dù chưa chính thức). Với máy chủ thì IIS của Microsoft sẽ hỗ trợ trên Window 10, ngoài ra Apache và Nginx cũng sẽ hỗ trợ trong thời gian tới.

**2.3.2 StreamObserver ,StreamingRecognizeRequest, StreamingRecognizeResponse**

StreamObserver, StreamingRecognizeRequest và StreamingRecognizeResponse là những lớp của thư viện GRPC, hỗ trợ ta trong việc sử dụng RPC kết nối giữa client và server của Google Cloud.

**2.3.2.1 StreamObserVer**

StreamObserver là 1 interface của thư viện GRPC, dùng để nhận thông báo từ 1 mesage stream. Ta sử dụng StreamObserver để kết nối giữa client và server của Google Cloud thay thế cho Json Over Http.

StreamObserver được sử dụng cho cả client lẫn server để gửi và nhận message. Để nhận message, client sử dụng StreamObserver và truyền vào cho thư viện GRPC để nhận tin nhắn và cũng tương tự như vậy đối với việc gửi tin nhắn.

**2.3.2.2 StreamingRecognizeRequest và StreamingRecognizeResponse**

StreamingRecognizeRequest và StreamingRecognizeResponse là các lớp dùng để xác định kiểu cho interface StreamObserver. Với StremObserver kiểu StreamingRecognizeRequest sẽ liên tục gửi request lên server , và StremObserver kiểu StreamingRecognizeResponse sẽ nhận response mỗi khi server trả về.

**2.4 Hạn chế**

Hiện nay Google Cloud Platform Speech To Text API đang tính phí theo độ dài audio của từng request gửi lên. Với mỗi audio có độ dài nhỏ hơn 15 giây sẽ tốn 0.06$ . Nếu audio có độ dài lớn hơn 15 giây thì sẽ chiều thành nhiều đoạn nhỏ hơn hoặc bằng 15 giây rồi tính phí như trên. Một chi phí khá là đắt đỏ nếu đưa ứng dụng áp dụng vào thực tế với số lượng người dùng lớn.

Mặc dù Google Cloud Platform Speech To Text API đã hạn chế tối đa việc bị nhiễu bởi tiếng ồn xung quanh, nhưng đối với tiếng ồn lớn và kéo dài thì việc lọc được giọng nói của người sử dụng vẫn còn gặp nhiều khó khăn , từ đó việc chuyển giọng nói dưới dạng audio thành văn bản không được chính xác.

**2.5 Thực nghiệm trên 3 thứ tiếng Bắc, Trung, Nam**

**3.FPT AI Text To Speech(FPT Speech Synthesis)**

**3.1. Giới thiệu**

Tổng hợp tiếng nói (Speech synthesis) là bài toán cơ bản trong bài toán lớn về giao tiếp giữa người và máy (Human-Machine Interface).

FPT Speech Synthesis là sản phẩm của ban công nghệ FPT (FTI) đã nỗ lực nghiên cứu suốt gần 5 năm qua. Sản phẩm là sự kết hợp của:

* Ngôn ngữ học (Linguistics): âm vị học (Phonology), hình thái học (Morphology), ngữ dụng học (Pragmatics)
* Vật lý học: âm học
* Công nghệ: Xử lý ngôn ngữ tự nhiên (Natural Language Processing) , học máy (Machine Learning), xử lý tín hiệu số (Digital Signal Processing)

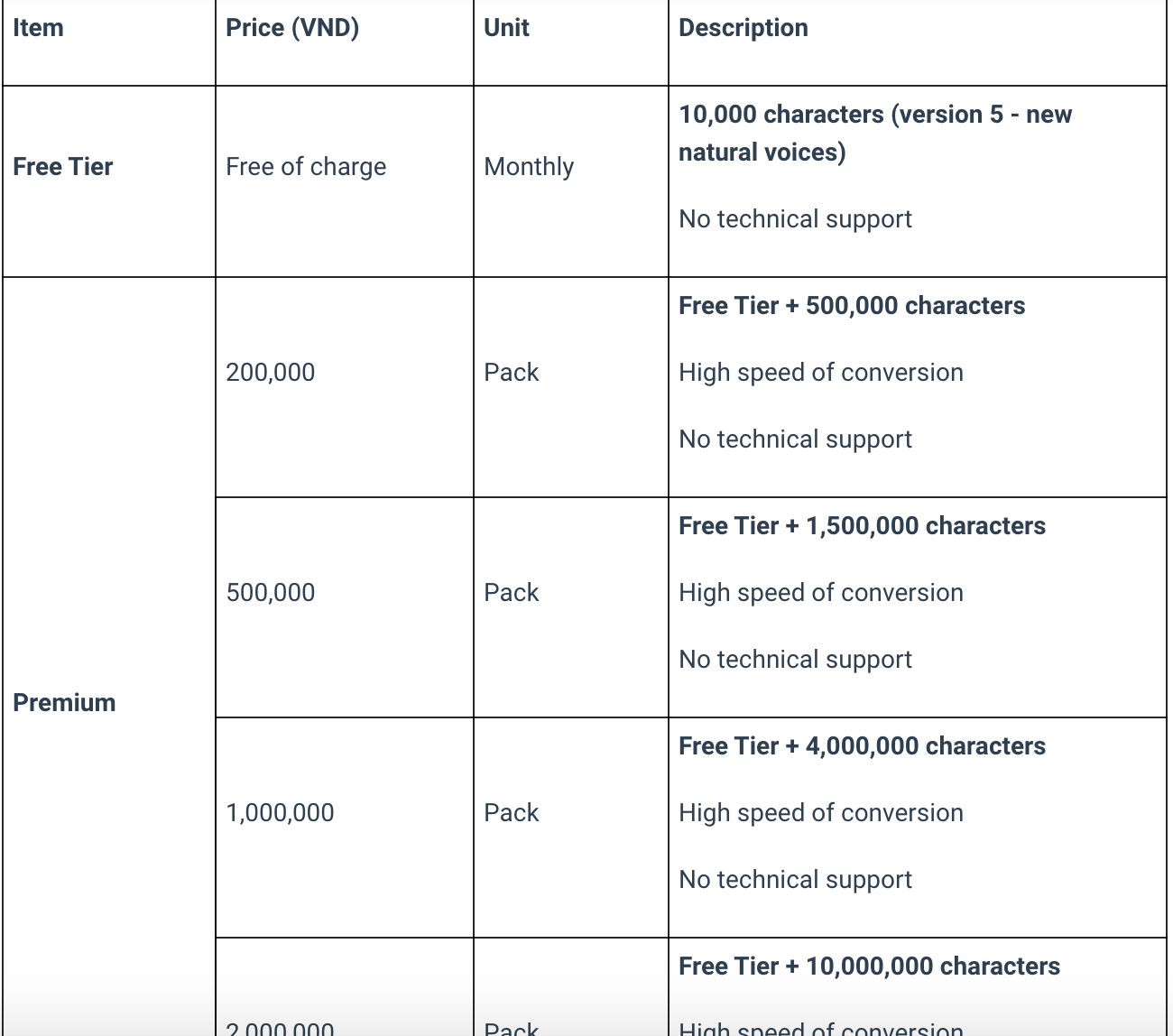
Được đánh giá là hệ thống tổng hợp tiếng nói tiếng Việt có chất lượng tốt nhất thị trường hiện nay, hệ thống tổng hợp tiếng Việt mới của FPT đang được mở trên Open FPT (http://openfpt.vn/). Các nhà phát triển có thể khai thác nguồn tài nguyên này để xây dựng ứng dụng của riêng mình trên các nền tảng khác nhau. Cho đến thời điểm này, đã có nhiều sản phẩm như “Giao thông thông minh” do FPT IS phát triển, hệ thống thông báo thông tin tài chính của VHT, hệ thống sinh ra video tự động từ bái báo điện tử, … đang sử dụng Speech Synthesis API của Open FPT và nhận được rất nhiều phản hồi tích cực từ người dùng.

Một số ứng dụng tiêu biểu của FPT Speech Synthesis:

* Xây dựng hệ thống giao tiếp tự động: tổng đài tự động, máy phục vụ tự động, robot giúp việc,…
* Tăng cường trải nghiệm: đọc nội dung cho người dung khi không thể theo dõi màn hình, các ứng dụng đọc sách báo tự động,..
* Phục vụ tương tác bằng tiếng dành cho người khiếm thị

**3.2.Hạn chế**

Cũng như Google Cloud Platform Speech To Text API,Hiện nay Google Cloud Platform Speech To Text API đang tính như bảng sau :



Và vẫn sẽ khá là đắt đỏ nếu đưa ứng dụng áp dụng vào thực tế với số lượng người dùng lớn.

**IV.Xây dựng ứng dụng**

**1.Khảo sát**

**1.1. Khảo sát 1 số ứng dụng tương tự hoạt động trên thị trường**

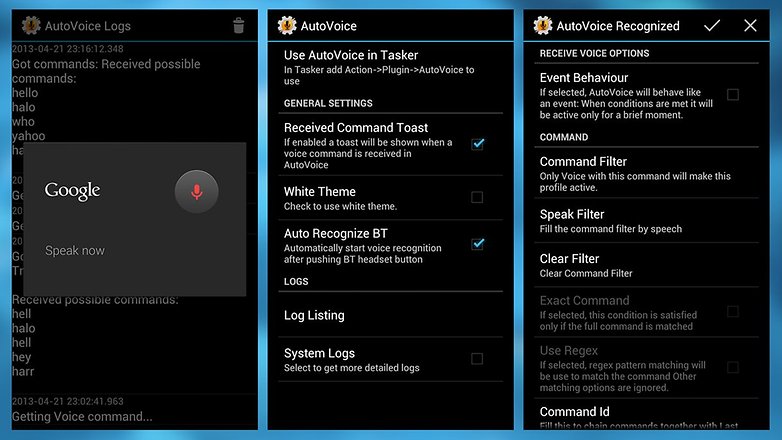
**- Google Now**:



Chức năng nhận diện giọng nói của **Google Now** ngày càng trở nên phổ biến, đây cũng là khởi đầu của vô số ứng dụng ‘ăn theo’ trên chợ điện tử hiện nay. Thay vì phải gõ từ khóa dài loằng ngoằng, giờ bạn chỉ cần nhấc điện thoại lên và đọc câu lệnh để tìm kiếm, nhưng nhớ là phải kích hoạt Google Now trước đó. Các câu lệnh này sẽ được xử lý trên công nghệ đám mây nên đảm bảo tốc độ và sự linh hoạt. Đây là tính năng cực kỳ hữu ích khi bạn phải tìm kiếm những câu hỏi dài, khi đang lái xe hay bận tay…

Ngoài các gói ngôn ngữ phổ biến, hiện nay người dùng ở Việt Nam đã hoàn toàn có thể yên tâm khi sử dụng tính năng này nhờ gói hỗ trợ tiếng Việt.

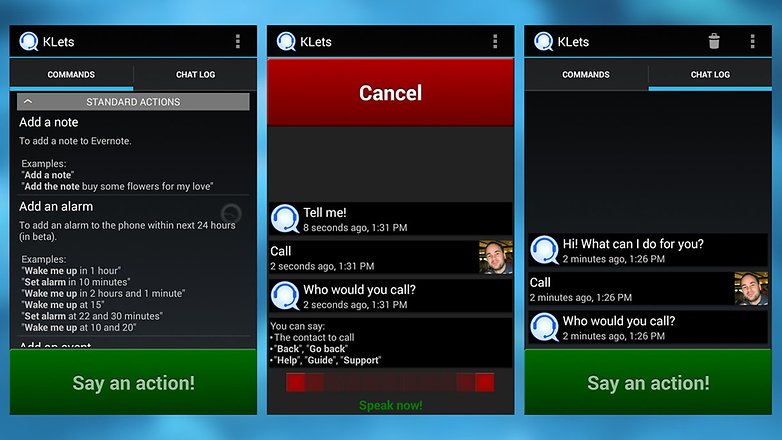
**- AutoVoice**:



AutoVoice là một ứng dụng cực kỳ đáng để cân nhắc nếu bạn muốn thêm lệnh tùy chỉnh đến **Google Now** — ứng dụng này không yêu cầu phải Root máy. Bạn có thể sử dụng phần mềm miễn phí trong vòng bảy ngày, nhưng sau đó nếu muốn tiếp tục dùng, bạn sẽ phải trả phí.

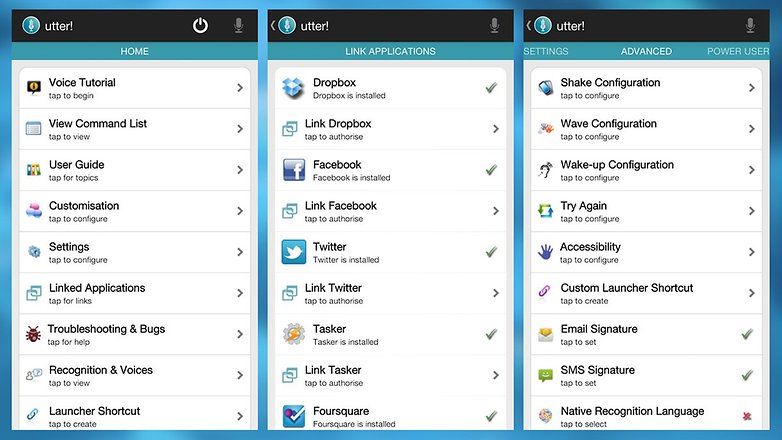
Bạn có thể tạo lệnh thoại tùy chỉnh để kích hoạt ứng dụng cụ thể, hoặc điều chỉnh âm lượng trên điện thoại. Hãy nhớ bạn cần mua và tải Tasker để có thể kích hoạt tích hợp với Google Now, bạn sẽ nhận được rất nhiều tính năng tương ứng với số tiền bỏ ra để có thể **mở khóa màn hình bằng giọng nói.**

**-Klets:**



Klets là công cụ tập trung để làm những điều bạn muốn ngay trên chiếc điện thoại Android của mình bằng giọng nói. Đây là một ưng dụng miễn phí khá hữu ích mà bạn nên sử dụng. Ứng dụng này sử dụng tính năng nhận dạng giọng nói đã được tích hợp trên Android để cung cấp thêm quyền kiểm soát và lựa chọn bổ sung. Chẳng hạn như gọi điện, gửi tin nhắn, kích hoạt ứng dụng, tương tác với các ứng dụng khác, nhận thông tin tình trạng điện thoại…

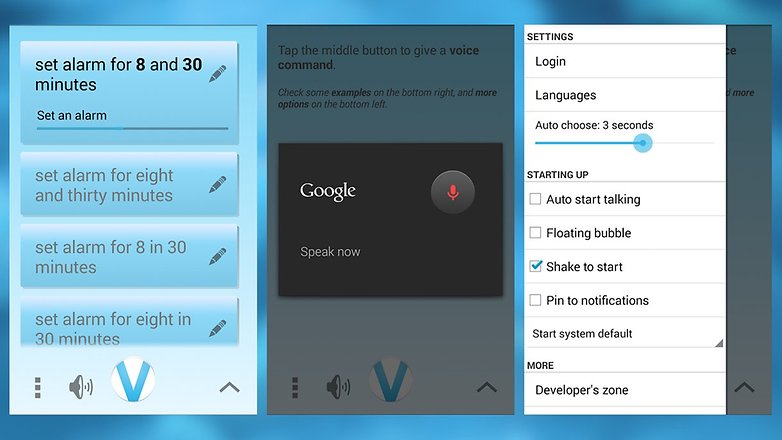
## ****Utter Voice Commands:****



**Utter Voice Commands**được thiết kế giống như một trợ lý thông minh, nhưng kiểm soát bằng giọng nói là tính năng chính của ứng dụng này. Nó được thiết kế tập trung vào độ khả dụng và tốc độ, nhiều tính năng của ứng dụng có thể sử dụng khi bạn đang offline trong trường hợp không có kết nối mạng dữ liệu và Wifi.

Thông qua **Utter**, người dùng có thể sử dụng điều khiển giọng nói để kích hoạt ứng dụng, lựa chọn menu. Ứng dụng này tích hợp với hầu hết các ứng dụng phổ biến khác, cũng như các tính năng hỗ trợ trên Android, chẳng hạn GPS và Wifi.

**- MyVoice:**



Ứng dụng **MyVoice** là một thiết kế thông minh trong việc điều khiển bằng giọng nói trên **Android**, nó mang đến sự thoải mái cho người dùng. Đặc biệt, bạn có thể nói tự nhiên hơn với chiếc điện thoai của mình. Ứng dụng này giúp tạo ra những lệnh thoại tùy chỉnh dễ dàng.

**Ứng dụng MyVoice**cũng hỗ trợ rất nhiều các tính năng, chẳng hạn như mở ứng dụng, kiểm tra thông báo hay truy cập lựa chọn bên trong từng ứng dụng. Thông qua website, người dùng có thể chỉnh sửa hay ngắt lệnh thoại của mình và chắc chắn nó sẽ đáp ứng được đầy đủ các nhu cầu của người dùng.

**2.Sơ đồ usecase**

**3.Kịch bản**

**Usecase gửi tin nhắn:**

|  |  |
| --- | --- |
| Tên usecase | Gửi tin nhắn |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Người dùng nói lệnh : “Gửi tin nhắn” |
| Đảm bảo tối thiểu | Điện thoại đã gửi được tin nhắn |
| Đảm bảo thành công | App hiển thị thong báo và phát ra đoạn âm thanh : “Tin nhắn gửi thành công” |
| Kích hoạt | Người dung nói lệnh :”Gửi tin nhắn” |
| **Chuỗi sự kiện chính: Người dung gửi tin nhắn:**  1. Người dùng nói “Gửi tin nhắn”.  2.App sẽ phát ra đoạn âm thanh “Đọc số điện thoại muốn gửi” và sau đó phát ra tiếng bíp.  3. Người dùng đọc số điện thoại mà mình muốn gửi tin nhắn .  4. App sẽ phát ra đoạn âm thanh “Đọc Nội dung tin nhắn” và sau đó phát ra tiếng bíp.  5. Người dùng đọc nội dung tin nhắn.  6. Người dung nói “Gửi”, app bắt đầu gửi tin nhắn.  7. App hiển thị thông báo và phát ra đoạn âm thanh “Tin nhắn gửi thành công”. | |
| **Ngoại lệ:**  3.1 Người dùng đọc sai số điện thoại.  3.2 Hệ thống hiển thị thông báo và phát ra đoạn âm thanh “Tin nhắn gửi thất bại”. | |

**Usecase Gọi điện :**

|  |  |
| --- | --- |
| Tên usecase | Gọi điện |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Người dùng nói lệnh : “Gọi điện” |
| Đảm bảo tối thiểu | Hệ thống đã gọi điện |
| Đảm bảo thành công | App hiển thị giao diện đang gọi điện |
| Kích hoạt | Người dung nói lệnh :”Gọi điện” |
| **Chuỗi sự kiện chính: Người dung gọi điện:**  1. Người dùng nói “Gọi điện”.  2.App sẽ phát ra đoạn âm thanh “Đọc số điện thoại muốn gọi” và sau đó phát ra tiếng bíp.  3. Người dùng đọc số điện thoại mà mình muốn gọi.  4. App hiển thị giao diện giao diện đang gọi điện thoại. | |
| **Ngoại lệ:**  3.1 Người dùng đọc sai số điện thoại hoặc sai định dạng số điện thoại.  3.2 Hệ thống hiển thị thông báo và phát ra đoạn âm thanh “Số điện thoại sai định dạng, vui long đọc lại số điện thoại”.  3.3 Người dùng đọc lại số điện thoại và hệ thống hiển thị giao diện cuộc gọi. | |

**Usecase Gửi Mail:**

|  |  |
| --- | --- |
| Tên usecase | Gửi mail |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Người dùng nói lệnh : “Gửi mail” |
| Đảm bảo tối thiểu | Hệ thống đã gửi được mail |
| Đảm bảo thành công | App hiển thị thong báo và phát ra đoạn âm thanh : “Mail gửi thành công” |
| Kích hoạt | Người dung nói lệnh :”Gửi mail” |
| **Chuỗi sự kiện chính: Người dung gửi tin nhắn:**  1. Người dùng nói “Gửi mail”.  2.App sẽ phát ra đoạn âm thanh “Đọc địa chỉ gmail muốn gửi” và sau đó phát ra tiếng bíp.  3. Người dùng đọc địa chỉ gmail mà mình muốn gửi.  4. App sẽ phát ra đoạn âm thanh “Đọc tiêu đề” và sau đó phát ra tiếng bíp.  5. Người dùng đọc tiêu đề.  6. App sẽ phát ra đoạn âm thanh “Đọc nội dung” và sau đó phát ra tiếng bíp.  7. Người dùng đọc nội dung mail.  8. Người dung nói “Gửi”, hệ thống bắt đầu gửi mail.  9. App hiển thị thông báo và phát ra đoạn âm thanh “Mail gửi thành công”. | |
| **Ngoại lệ:**  3.1 Người dùng đọc sai địa chỉ gmail.  3.2 Hệ thống hiển thị thông báo và phát ra đoạn âm thanh “Mail gửi thất bại”. | |

**Usecase Đọc tin nhắn:**

|  |  |
| --- | --- |
| Tên usecase | Đọc tin nhắn |
| Tác nhân chính | Người dùng đã khởi động app |
| Tiền điều kiện | Người dùng nói lệnh : “Đọc tin nhắn” |
| Đảm bảo tối thiểu | Hệ thống phát ra đoạn âm thanh nội dung của tin nhắn muốn đọc. |
| Đảm bảo thành công | Hệ thống phát ra đoạn âm thanh với nội dung đầy đủ của tin nhắn muốn đọc. |
| Kích hoạt | Người dung nói lệnh :”Đọc tin nhắn” |
| **Chuỗi sự kiện chính: Người dung gửi tin nhắn:**  1. Người dùng nói “Đọc tin nhắn”.  2.App sẽ phát ra đoạn âm thanh “Đọc thứ tự tin nhắn muốn đọc” và sau đó phát ra tiếng bíp.  3. Người dùng đọc số thứ tự của tin nhắn muốn đọc .  4. Hệ thống phát ra đoạn âm thanh nội dung của tin nhắn muốn đọc. | |
| **Ngoại lệ:**  3.1 Người dùng đọc số thứ tự tin nhắn vượt quá số lượng tin nhắn có trong máy.  3.2 Hệ thống hiển thị thông báo và phát ra đoạn âm thanh “Tin nhắn không tồn tại”. | |

**Usecase Nhận diện cuộc hội thoại:**

|  |  |
| --- | --- |
| Tên usecase | Nhận diện cuộc hội thoại |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Người dùng đã gọi điện thoại và đang bật loa ngoài |
| Đảm bảo tối thiểu | Đoạn hội thoại của 2 người được hiển thị đầy đủ trên màn hình hệ thống |
| Đảm bảo thành công | Đoạn hội thoại của 2 người được hiển thị đầy đủ trên màn hình hệ thống cho đến khi cuộc điện thoại kết thúc |
| Kích hoạt | Người dung gọi điện và bật loa ngoài |
| **Chuỗi sự kiện chính: Người dung gửi tin nhắn:**  1. Người dung gọi điện thoại , trong màn hình giao diện cuộc gọi bật chế độ loa ngoài.  2. Hệ thống sẽ hiển thị đầy đủ cuộc hội thoại 2 người dưới dạng văn bản trên màn hình. | |
| **Ngoại lệ:**  3.1 Kết nối mạng bị ngắt khi đang trong cuộc hội thoại.  3.2 Hệ thống sẽ ngừng nhận diện cuộc hội thoại thành dạng văn bản của 2 người | |

**Usecase Đọc mail:**

|  |  |
| --- | --- |
| Tên usecase | Gửi tin nhắn |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Người dùng nói lệnh : “Gửi tin nhắn” |
| Đảm bảo tối thiểu | Điện thoại đã gửi được tin nhắn |
| Đảm bảo thành công | App hiển thị thong báo và phát ra đoạn âm thanh : “Tin nhắn gửi thành công” |
| Kích hoạt | Người dung nói lệnh :”Gửi tin nhắn” |
| **Chuỗi sự kiện chính: Người dung gửi tin nhắn:**  1. Người dùng nói “Gửi tin nhắn”.  2.App sẽ phát ra đoạn âm thanh “Đọc số điện thoại muốn gửi” và sau đó phát ra tiếng bíp.  3. Người dùng đọc số điện thoại mà mình muốn gửi tin nhắn .  4. App sẽ phát ra đoạn âm thanh “Đọc Nội dung tin nhắn” và sau đó phát ra tiếng bíp.  5. Người dùng đọc nội dung tin nhắn.  6. Người dung nói “Gửi”, app bắt đầu gửi tin nhắn.  7. App hiển thị thông báo và phát ra đoạn âm thanh “Tin nhắn gửi thành công”. | |
| **Ngoại lệ:**  3.1 Người dùng đọc sai số điện thoại.  3.2 Hệ thống hiển thị thông báo và phát ra đoạn âm thanh “Tin nhắn gửi thất bại”. | |

**Usecase Nhận cuộc gọi đến:**

|  |  |
| --- | --- |
| Tên usecase | Nhận cuộc gọi đến |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Có cuộc gọi đến điện thoại |
| Đảm bảo tối thiểu | Đồng ý nhận cuộc gọi đến |
| Đảm bảo thành công | Đồng ý nhận cuộc gọi đến |
| Kích hoạt | Người dung nói lệnh :”Nhận cuộc gọi” |
| **Chuỗi sự kiện chính: Nhận cuộc gọi đến:**  1. Có cuộc gọi đến điện thoại.  2. Người dung nói “Nhận cuộc gọi”  3. Cuộc gọi được nhận mà không cần phải them 1 thao tác gì với điện thoại. | |
| **Ngoại lệ:**  3.1 Do có lẫn âm thanh của tiếng chuông điện thoại nên hệ thống nhận diện câu lệnh không chính xác.  3.2 Hệ thống không nhận được cuộc gọi. | |

**Usecase Từ chối cuộc gọi đến:**

|  |  |
| --- | --- |
| Tên usecase | Từ chối cuộc gọi đến |
| Tác nhân chính | Người dung đã khởi động app |
| Tiền điều kiện | Có cuộc gọi đến điện thoại |
| Đảm bảo tối thiểu | Từ chối nhận cuộc gọi đến |
| Đảm bảo thành công | Từ chối nhận cuộc gọi đến |
| Kích hoạt | Người dung nói lệnh :”Từ chối” |
| **Chuỗi sự kiện chính: Nhận cuộc gọi đến:**  1. Có cuộc gọi đến điện thoại.  2. Người dung nói “Từ chối”  3. Cuộc gọi bị từ chối mà không cần phải thêm 1 thao tác gì với điện thoại. | |
| **Ngoại lệ:**  3.1 Do có lẫn âm thanh của tiếng chuông điện thoại nên hệ thống nhận diện câu lệnh không chính xác.  3.2 Hệ thống không từ chối được cuộc gọi. | |

**V.Cài đặt ứng dụng.**