)

Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito final para obtener el Título de Ingeniero de Sistemas

Sistema de órdenes de voz en software libre

Por

Br. Ortega A., Jorge E.

Tutor: Prof. Dávila, Jacinto

Cotutor: Prof. Márquez, Melva

Enero 2016

Sistema de órdenes de voz en software libre

Br. Ortega A., Jorge E.

Proyecto de Grado — Sistemas computacionales, 135 páginas
Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2016

Resumen: En este proyecto se desarrolló, bajo tecnologías libres, un software o sistema que permite a una persona interactuar con el sistema operativo y su interfaz gráfica mediante órdenes de voz (como: abrir, cerrar, maximizar o minimizar ventana, entre otros) con las aplicaciones instaladas. El sistema, al que se ha denominado "gpyvozControl", también permite manejar los programas como el Evince (visualizador de PDF) y el reproductor de música (Rhythmbox). Estas fueron las características que se alcanzaron en el proyecto, pero teniendo presente que la finalidad superior es continuar con el desarrollo del Sistema de órdenes de voz que permita a cualquier persona en un futuro realizar todas las tareas del sistema solo con la voz. El proyecto se decantó por el uso de tecnologías libres, después de una revisión de la oferta disponible y una estimación del esfuerzo y recursos requeridos en otras plataformas, además de las ventajas que ofrece al utilizar las mismas. Se revisó las bibliotecas de desarrollo de uso libre, creadas por un par de proyectos de procesamiento de voz como lo son Julius y CMUSphinx y que se adaptaron en la tarea especifíca de la transmición al computador de un subconjunto, seleccionado cuidadosamente, de los comandos que se transmiten con el teclado de un computador personal estándar. En el ejercicio, se identificó debilidades, ventajas y posibilidades de desarrollo, al tiempo que se terminó con un prototipo funcional personal para cada unade las bibliotecas. La metodología usada, es el método ágil Scrum, que permite el desarrollo progresivo de ese prototipo, planificando los requisitos por prioridad de desarrollo, la duración de cada sprints y las actividades definidas por las historias de usuario que fueron diseñadas. Este proyecto, además, fue un ejercicio interdisciplinario que combinó la experiencia de vida del propio desarrollador con generosas "donaciones de voz", con herramientas para el procesamiento de voz y con el conocimiento de una experta en lingüística computacional.

Palabras clave: órdenes de voz, Tecnologías libres, Lingüística computacional, Motores de reconocimiento de voz, Inteligencia artificial

Índice general

Dedicatoria

Todo este trabajo, esfuerzo y tiempo empleado a la culminación de la carrera de Ingeniería de Sistemas, están dedicados a Dios y la Virgen María, Rosa Mística, por sus bendiciones, a mi mamá quien con sus consejos, cariño, trabajo incondicional y trasnochos compartió conmigo este gran sueño que juntos estamos logrando, a mi hermano Julio Cesar Ortega que desde allá donde solo los ángeles pueden hacer milagros, estuvo a mi lado en los momentos alegres, tristes y difíciles de mi carrera, se que siempre me acompañas a todos lados "Te Quiero Mucho", y a todas las personas con alguna discapacidad ya sea visual, auditiva, motora o congnitiva, que como dice el Filósofo latino Lucio Anneo Séneca: "No nos atrevemos a muchas cosas porque son difíciles, pero son difíciles porque no nos atrevemos a hacerlas", solo necesitamos hacer el esfuerzo para lograr y conseguir lo que deseamos, no existen límites cuando nos proponemos algo.

Agradecimientos

Agradezco a Dios por haber permitido cumplir con este anhelado sueño, a mi mamá que con sus enseñanzas, amor, dedicación me inspira a seguir adelante en todo momento, a mi hermano Jose Constantino quien me apóyo siempre en mis decisiones, a los profesores que compartieron sus conocimientos en cada clase y que ayudaron en mi aprendizaje, a aquellos profesores que dedicaron tiempo fuera de las aulas a explicar lo que me costaba entender, a los que hicieron el esfuerzo en cambiar su metodología de enseñanza para que entendiera los contenidos y en general por hacerme sentir como un estudiante mas a pesar de mi discapacidad visual, a los compañeros de clases que compartieron conmigo esta larga travesía y en especial aquellos que me ofrecieron su amistad. A mi tutor Jacinto Dávila quien con su guía y dedicación se logró concluir este proyecto. Por ultimo quiero agradecer a todo el personal de vigilancia, cafetín y secretarias de la escuela de sistemas por su cariñosa atención. Que Dios los bendiga a todos.

Capítulo 1

Introducción

1.1. Introducción

Las tecnologías de información se han convertido en una herramienta clave para la integración y comunicación de las personas. El computador es ahora una herramienta habitual de trabajo. Se usa para llevar la agenda, como elemento de ocio, acceso a internet, pagar servicios básicos, entre otros y esto no excluye a las personas con alguna discapacidad. Por este motivo, se visualiza a las tecnologías como herramientas que habilitan el acceso a la información y a mecanismos de aprendizaje para estas personas, de tal forma que con ciertas ayudas tecnológicas puedan incorporarse a estos procesos de aprendizaje en condiciones de igualdad. En este punto nos encontramos con dos aspectos de esa realidad. Por un lado, muy pocos desarrolladores o arquitectos de software incorporan en sus sistemas y aplicaciones alguna plataforma de accesibilidad que las personas con alguna discapacidad puedan usar libremente. Por el otro lado se tiene el alto costo de las licencias privativas de algunas aplicaciones con este tipo de ayudas tecnológicas.

Es importante resaltar que cuando se hace referencia a "libre", no quiere decir que es "gratis", sino que los fuentes puedan estar disponibles para su revisión, distribución, uso y modificación, así puedan ser adaptados de acuerdo a cada circunstancia. Esto las hace particularmente valiosas para atender discapacidades específicas y en lugares de mundo con menos recursos tecnológicos.

Entre las mejores alternativas para una persona que no puede utilizar el teclado o el ratón, se cuenta el reconocimiento de voz. El hardware y los programas han venido evolucionando en los últimos tiempos, sobre todo en los teléfonos celulares de nueva generación, permitiendo al usuario realizar tareas como búsquedas en internet, llamar a un contacto, escribir un mensaje y abrir o ejecutar programas. En el software libre, existen algunos motores de reconocimiento de voz, que permiten desarrollar programas de este tipo. Entre los motores de software libre más conocidos está Julius[1] que es un sistema de reconocimiento preciso en tiempo real, de alta velocidad, basado en la estrategia de doble paso (consiste en un árbol de busqueda hacia adelante en tiempo sincrónico y una busqueda hacia atrás en tiempo asíncrono). Este algoritmo incorpora las principales técnicas de decodificación que incluyen léxico organizado como árbol, aproximación del contexto por primer mejor par de palabras, podado por rankeo, factorización de N-grama, manejo de dependencias de contexto por cruce de palabras, búsqueda enfocada, podado gausiano, selección gausiana, entre otros [2]. Otro motor que promete es CMUSphinx[3] de la Universidad Carnegie Mellon, un conjunto de herramientas con algoritmos de última generación para el reconocimiento eficaz del habla, las herramientas están diseñadas específicamente para plataformas de bajos recursos. Tiene un diseño flexible para centrarse en el desarrollo de la aplicación práctica y no en la investigación, soporte para varios idiomas como el Inglés en sus variantes británica y estadounidense, francés, mandarín, alemán, holandés, ruso y capacidad de construir un modelo para los demás idiomas, posee una amplia gama de herramientas para múltiples propósitos relacionados con el reconocimiento de voz.

Estas aplicaciones de reconocimiento de voz permiten al usuario cierta independencia de la interacción con el teclado y el ratón. Si resaltamos el hecho de que con bastante frecuencia se desconocen los problemas de accesibilidad, entendiendo esta como el grado en que una persona con sus capacidades puede utilizar un objeto, visitar un sitio o acceder a un servicio, estamos ante un gran problema que se le presenta a las personas con discapacidad para utilizar un computador. El reconocimiento de voz puede ofrecer una alternativa para aliviar ese gran problema. Una alternativa que, además no beneficia únicamente a un perfil de discapacidad particular, sino que puede ayudar a muchas personas que encuentren difícil, en cualquier medida, usar el ratón o

el teclado para usar un computador.

Por este motivo, en este proyecto se desarrolló, bajo tecnologías libres, un software o sistema que permite a una persona interactuar con el sistema operativo y su interfaz gráfica mediante órdenes de voz (como: abrir programas, cerrar, maximizar o minimizar ventanas, entre otros) con las aplicaciones instaladas. El sistema, al que se ha denominado "gpyvozcontrol", también permite manejar los programas como el Evince (visualizador de PDF) y el reproductor de música (Rhythmbox). Estas fueron las características que se alcanzaron en el proyecto, pero teniendo presente que la finalidad superior es continuar con el desarrollo del sistema de órdenes de voz que permita a cualquier persona en un futuro realizar todas las tareas del sistema solo con la voz y a través del oído.

El nombre "gpyvozcontrol" surge de la combinación de las letras "g" por el escritorio Gnome, "py" de Python, ya que esta implementado en este lenguaje y "vozcontrol" porque es con la voz del usuario que se controla el equipo.

En este trabajo se hace una revisión de los desarrollos de reconocimiento de voz existentes en software libre, su funcionamiento, su implementación y su evolución. Luego se explica la metodología "Scrum"[4] aplicada en el desarrollo del proyecto. Seguidamente se hace el entrenamiento de los modelos acústicos con las bibliotecas Julius[1] y CMUSphinx[3], los cuales están basado en los tutoriales"Create Acoustic Model - Manually"[5]y el conjunto de herramientas de HTK[6] (The Hidden Markov Model Toolkit), y el tutorial "Training Acoustic Model For CMUSphinx"[7], respectivamente. Ya con los modelos acústicos creados se diseña el sistema de órdenes de voz "gpyVozControl" el cual fue implementado con el lenguaje python. Se describen las pruebas realizadas y los resultados obtenidos para los modelos acústicos, haciendo un análisis de los errores de reconocimiento y la diferencia entre ellos. Por último se concluye con los logros obtenidos y el futuro del proyecto.

1.2. Antecedentes

Esta investigación se ha limitado a los desarrollos en software libre debido a que ofrecen la posibilidad de estudiarlos a profundidad y adaptarlos a voluntad a condiciones muy particulares de uso como las que se plantean en este proyecto. Haciendo una revisión de los programas de reconocimiento de voz que existen en el software libre, se pueden encontrar entre los más destacados los siguientes:

CVoiceControl[8] desarrollado por Daniel Kiecza, es un programa de reconocimiento de voz, que permite a un usuario ejecutar comandos en sistemas Unix, mediante la entrada de voz por un micrófono. Si el comando es reconocido el programa ejecuta la acción asociada. El enfoque que emplea este sistema es el reconocimiento de palabras aisladas dependiente de la comparación con plantillas. Se debe tener en cuenta que cuando se refiere al reconocimiento de palabras aisladas, no significa que necesariamente se refiere a una sola palabra, sino que describe una clase de sistemas entre reconocimiento continuo, dictado, habla espontánea, entre otros. Es de destacar que no soporta la concatenación de frases para ejecutar varios comandos, como por ejemplo "Por favor abrir editor de texto, el navegador web y el terminal" en una misma oración. Se recomienda oraciones largas para evitar que la precisión disminuya, debido a su enfoque de comparación de plantillas como se mencionó anteriormente. La última versión es la 0.9alpha, del 12 de febrero del 2000.

También existe PerlBox[9], un software de voz para GNU-Linux, que permite tener control del escritorio mediante la entrada de voz por un micrófono. Entre sus características más importantes está la conversión de texto a voz con el sintetizador Festival, control de voz para abrir aplicaciones, plugins para controlar el escritorio y comandos personalizados por el usuario. Se adapta a la voz de cualquier usuario por lo que no se necesita entrenamiento particular. Es robusto al ruido externo y de ambiente y ha sido desarrollado con perl y perl-tk, lo cual implica que posee un API (Perlbox::VoiceServer[10]) de interfaz para la creación de aplicaciones de voz. Debido a que la interfaz gráfica está desarrollada con perl-tk, el software no es accesible a través de lectores de pantalla.

En el 2007, Google Summer of Code[11] desarrolló el proyecto GnomeVoiceControl[12]. El mismo es un programa que permite controlar con

voz el escritorio de Gnome, el sistema supervisa la entrada de audio por medio del micrófono y cuando detecta una señal de audio significativa, el programa captura, procesa, reconoce la señal y luego ejecuta la acción deseada en el escritorio Gnome. Entre las acciones que se pueden realizar están minimizar, maximizar, cerrar la ventana activa; cambiar de un escritorio a otro; abrir un programa; entre otros. GnomeVoiceControl[12] está implementado en C junto con CMUSphinx[3], que es una herramienta de código libre, creada para convertir voz en texto. En [13] se encuentran los fuentes y las revisiones de este programa para su distribución en Ubuntu.

Google2ubuntu[14] es otro programa de reconocimiento de voz comenzado por el usuario benoitfragit en el 2010, que utiliza la API de Google Voice[15]. Se emplean dos tipos de comandos de voz, los externos y los internos. Los externos ejecutan acciones como indicar la hora y el tiempo, leer un texto seleccionado o buscar una palabra. Los internos ejecutan operaciones como minimizar, maximizar y abrir (correr) un programa. Es decir, ejecutan comandos de la interfaz del usuario. También se puede personalizar los comandos con el script de atajos de teclas del programa. Su código está escrito en python. Debido a que usa la API de Google requiere de una conexión a internet para ejecutar los comandos. Sin embargo, la mayor limitación para efectos de este proyecto es que algunos de los códigos de la API no son software libre. Google solamente se compromete a una liberación parcial con lo cual se genera una condición de dependencia relativa. En [16] se pueden revisar en inglés los términos de uso del API.

Simon[17] es un programa de reconocimiento de voz para el escritorio de KD, es el mas reciente actualmente, se lanzó en el 2013 por Peter Grasch[18]. El desarrollo comenzó en Alemania. Actualmente soporta el alemán y el inglés. El español podría estar soportado, pero hace falta el diccionario y las traducciones fonéticas en español para que funcione correctamente. El motor de reconocimiento de voz que utiliza este programa es Julius[1], el mismo que revisaremos en este proyecto. Simon[17] esta desarrollado en C++ junto con QT y KD para la interfaz de usuario, lo que asegura su integración con KD4, aunque es posible usarlo con otros entornos de escritorio. Debido a que el sistema esta implementado en QT y KD, su interfaz no es muy accesible con el lector de pantalla Orca, lo que limita su uso a personas con discapacidad visual, además que su compilación en escritorios diferentes a KD es complejo.

Una prueba realizada para comprobar el reconocimiento de voz en ese software libre, consistió en crear un archivo de texto plano con algunas frases en inglés, se suben en [19], que genera un comprimido el cual contiene los archivos con extensión ".dic" (diccionario), ".vocab" (vocabulario), ".lm" (modelo de lenguaje), ".sent" (sentencias); luego con esos archivos se ejecuto por consola/terminal el siguiente comando:

```
pocketsphinx_continuous -inmic yes \-lm archivo.lm -dict
    archivo.dic
```

ya en ese momento se puede comenzar a pronunciar las frases para su reconocimiento. Es interesante resaltar que la prueba realizada fue satisfactoria, aunque aún no tengamos estadísticas de cuantas frases pueden ser reconocidas. En [20] se puede revisar el procedimiento que se hace para lograr lo anteriormente explicado.

También se hizo una revisión de la tesis "Reconocedor automático de comandos por medio del habla para las funciones de un automóvil"[21], realizado por Alfonso J. Bonnet, José A. Gutierres y Héctor A. Hernández, para optener el titulo de Ingeniero en comunicaciones y electrónica, Instituto Politécnico Nacional, Escuela Superior De Ingeniería Mecánica Y Eléctrica - México. El proyecto de grado fue el diseño y desarrollo de un programa para controlar funciones de un automóvil por medio de comandos de voz. El sistema guarda la instrucción del usuario en un audio, luego lo compara (aplicando diversos métodos) con los audios que están en una base de datos e identifica cual es la órden. El software se diseño en Matlab. Es importante resaltar que para nuestro trabajo no se aplicara este método, ya que son muchas las órdenes a almacenar, lo que ocuparía bastante memoria en disco, además de hacer lenta la respuesta al usuario y poco personalizable.

La revisión de esos desarrollos previos permitió realizar una comparación preliminar de los beneficios que ofrecen las plataformas en las que fueron desarrollados. Gracias a ese ejercicio, este proyecto optó por explorar las plataformas Julius[1] y CMUSphinx[3] en profundidad. En [5] y [7] se pueden revisar los tutoriales completos para la creación de un modelo acústico utilizando Julius[1] o CMUSphinx[3] respectivamente, que han sido implementados por los desarrollos anteriormente mensionados.

1.3. Objetivos

OBJETIVO GENERAL

- Investigar y desarrollar, con y para tecnologías libres, un software o sistema que permita a una persona interactuar con un sistema operativo y su interfaz gráfica mediante órdenes de voz (abrir programas, cerrar, maximizar o minimizar ventanas, entre otros) con las aplicaciones instaladas en un computador personal.

OBJETIVOS ESPECíFICOS

- Identificar y adecuar las bibliotecas que permiten hacer reconocimiento de voz en software libre.
- Identificar las bibliotecas del sistema operativo que se deben implementar para poder manipular las aplicaciones con el sistema de órdenes de voz.
- Realizar pruebas piloto con las bibliotecas encontradas.
- Grabar en audio los phonemas en español mediante una lista de frases, buscando diferentes acentos (Zulia, Mérida, Caracas, entre otros).
- Especificar las funciones principales del sistema operativo a ser controladas solo con la voz del usuaria o usuario, capturada con el micrófono.
- Desarrollar el sistema de órdenes de voz que permita interactuar con el sistema, teniendo en cuenta las historias de usuario predefinidas y el modelo acústico.
- Realizar pruebas con el sistema desarrollado en un ambiente controlado (condiciones ambientales promedio, voces diversas, entre otras).

Capítulo 2

Metodología

2.1. Metodología

La metodología adoptada para este trabajo de grado es una variación de Scrum[4], el cual es un marco de desarrollo ágil e incremental. A continuación se describe el proceso de desarrollo que se llevó a cabo según esta metodología.

Los actores y roles que intervinieron en el desarrollo del sistema de ordenes de voz se identifican en el cuadro 2.1.

Cuadro 2.1: Roles.

Rol	Actor
Responsable del proyecto	Br. Jorge Ortega
Líder del proyecto	Prof. Jacinto Dávila
Grupo desarrollo	Prof. Melva Márquez; Prof Jacinto Dávila; Br. Jorge Ortega

A continuación se describen las historias de usuario según dos perspectivas. En primer lugar, lo que hace una persona con discapacidad visual para abrir y manipular

los programas actualmente. En segundo lugar como lo haría con ordenes de voz.

Historias de usuario

1. Con teclado

   - Como usuario presiono el atajo de teclas Ctrl+Alt+(tecla asociada al programa) y abro el programa. Cuando este atajo no existe, pulso Alt+F1, escribo el nombre del programa, luego presiono la tecla enter y abro el mismo.
   - Como usuario presiono las teclas Alt+Space, me muevo con las teclas de dirección sobre las diferentes opciones y cierro, maximizo o minimizo la ventana activa.
   - Como usuario pulso las teclas Alt+Tab, cambio de ventana y suelto las teclas en el programa que quiero colocar activo.
   - Como usuario presiono las teclas Ctrl+Alt+D o Ctrl+Alt+Tab y me voy al Escritorio.

2. Con ordenes de voz

   - Como usuario digo en voz la orden"Abrir programa.." y abro el programa que quiero.
   - Como usuario digo en voz las ordenes "Minimizar; Maximizar o Cerrar ventana" en el programa que tiene el foco (está activo).
   - Como usuario digo en voz la orden "Minimizar/maximizar todo" o "Minimizar/maximizar todas las ventanas" y voy al escritorio.
   - Como usuario digo en voz la orden "Canción siguiente", entre otras ordenes para manipular el reproductor de música.
   - Como usuario digo en voz la orden "Lámina siguiente", entre otras ordenes para manipular el visualizador de archivos pdf.

De las historias de usuario, fueron extraídos los requisitos por prioridad de desarrollo del sistema. Luego se planificaron los sprint que se cumplieron para cada uno de los

requisitos según su importancia y por último las actividades o tareas que se realizaron para conseguir el objetivo de dicho sprint.

- Requisitos.

    1. Identificación de las bibliotecas necesarias del sistema operativo a utilizar, que permite comunicarse con las aplicaciones del sistema, realizando algunos script de pruebas para revisar su funcionamiento.
    2. Definición del motor y modelo acústico de reconocimiento de voz que permite identificar una gramática definida y que se utilizó en el sistema o software desarrollado, incluyendo el listado de fonemas básicos requeridos.
    3. Grabación de los fonemas en español mediante una lista de frases, con diferentes acentos (zulia, mérida, caracas, entre otros).
    4. Módulo de captura de audio que identifica la entrada de sonido por medio de un micrófono y que reconoce las frases pronunciadas por la usuaria o usuario.
    5. Integración del software o sistema de reconocimiento de voz utilizando el modelo acústico y que permite las funciones de abrir programas; minimizar, maximizar, cerrar ventana; manipular el visualizador de pdf y el reproductor de música.
    6. Reporte de pruebas funcionales del sistema de voz que permite tener un modelo acústico robusto y preciso.

En el cuadro o tabla 2.2 siguiente se muestran los Sprint y el tiempo estimado de duración que se dedicaron en cada uno de los requisitos.

Las actividades realizadas por cada actor y los sprint que fueron asignados a cada uno, se muestra a continuación:

- Sprint 1
    - bibliotecas

Cuadro 2.2: Sprint

Requisito	Duración
3	6 semanas
1	2 semanas
4	4 semanas
2	4 semanas
5	4 semanas
6	2 semanas

1. Identificación de las bibliotecas que permite comunicarse con las aplicaciones del sistema, que estén en software libre. Responsable: Br. Jorge. Revisado: Prof. Jacinto.
2. Implementación de las pruebas de funcionamiento de las bibliotecas encontradas utilizando python o java. Responsables: Br. Jorge y Prof. Jacinto. Revisado: Prof. Jacinto y Br. Jorge.
3. Identificación de la lista de fonemas básicos que requiere el modelo acústico. Responsables: Br. Jorge y Prof. Jacinto. Revisado: Prof. Melva.

- Grabaciones. Responsables: Profs. Jacinto y Melva. Revisado: Br. Jorge.
  1. Grabación de cada frase a 16000Hz y 16-bit.

- Sprint 2: Modelo acústico. Responsable: Br. Jorge. Revisado: Prof. Jacinto.
  1. Creación de la gramática implementada en el sistema de ordenes de voz.
  2. Identificación de los motores de voz utilizados en el reconocimiento de voz.
  3. Creación de los modelos acústicos con los motores de reconocimiento de voz encontrados.
  4. Pruebas de reconocimiento de voz utilizando la gramática anteriormente definida.

- Sprint 3: Modulo de captura de audio. Responsable: Br. Jorge. Revisado: Prof. Jacinto.

1. Creación del modulo que revisa constantemente la entrada de audio y que identifica las frases de la gramática definida.

- Sprint 4: Desarrollo del software o sistema. Responsable: Br. Jorge. Revisado: Profs. Jacinto y Melva.

  1. Implementación de las funciones que permite abrir, minimizar, maximizar, cerrar ventanas, entre otros por medio de la voz de la usuaria o usuario.

- Sprint 5: Pruebas. Responsable: Prof. Jacinto. Revisado: Br. Jorge y Prof. Melva.

  1. Pruebas funcionales del software o sistema
  2. Pruebas de reconocimiento de voz, revisando que las ordenes se ejecutaran correctamente.

En los siguientes capítulos se puede verificar el uso de esta metodología para el desarrollo del proyecto. Se menciona el hecho que las grabaciones se realizaron en dos grupos, 15 personas grabaron 200 frases y 19 personas grabaron 125 frases, esto se hizo para reforzar algunos fonemas que son utilizados en la gramática creada para este sistema. También se resalta que 176 de las frases son tomadas de [22] y 106 entre refranes y de creación propia.

Capítulo 3

Modelo acústico

Para la creación del sistema de órdenes de voz en software libre, se han creado un par de modelos acústicos para el lenguaje español con las bibliotecas Julius[1] y CMUSphinx[3], cada uno de los cuáles consiste en una representación de los sonidos o fonemas del idioma y su representación textual y simbólica en el computador.

Todo lo que se reporta en la sección 3.1 y 3.2, esta basado en los tutoriales"Create Acoustic Model - Manually"[5], utilizando el decodificador Julius[1] y el conjunto de herramientas de HTK[6] (The Hidden Markov Model Toolkit), y el tutorial "Training Acoustic Model For CMUSphinx"[7], con la biblioteca de CMUSphinx[3] respectivamente. Los tutoriales son inspirados por [5], [7] y algunos de los textos son nuestras traducciones libres de ese original.

3.1. Creación del modelo acústico con Julius y HTK

3.1.1. Bibliotecas (libraries) de Software necesarias

Los programas de soporte necesarios para la creación del modelo acústico son:

1. El conjunto de herramientas HTK Es un conjunto de herramientas de software para la construcción y manipulación de modelos acústicos representados como modelos ocultos de Markov[23]. Se utiliza principalmente en el reconocimiento de voz. La licencia de HTK[6] requiere estar registrado antes de descargar el conjunto de herramientas. Además, aunque las fuentes están disponibles, se establecen limitaciones para su libre distribución. Sin embargo el modelo acústico creado no tiene ninguna restricción para su distribución y operación autónoma. Para este desarrollo se utilizó la versión 3.4 de HTK[6], que se puede descargar desde `http://htk.eng.cam.ac.uk/ftp/software/HTK-3.4.1.tar.gz` una vez que tiene usuario y clave. La página web del registro es: `http://htk.eng.cam.ac.uk`

2. Julius Es un motor de reconocimiento continuo del habla. Se puede utilizar para el desarrollo de aplicaciones de control y dictado. Julius[1] es software libre y por tanto no tiene limitaciones para su uso. Utiliza modelos acústicos en formato HTK[6] y archivos de gramática en su propio formato. Se usó para este proyecto la version 4.2.2. Se puede descargar desde: `http://julius.osdn.jp/en_index.php?q=index-en.html#download_julius`. La versión usada en el proyecto está en: `http://osdn.jp/projects/julius/downloads/56549/julius-4.2.2.tar.gz/`

3. Julia Es un lenguaje para programar a Julius. Se dice que es un lenguaje dinámico, flexible, de alto rendimiento similar a los lenguajes estáticos, combina programación imperativa, funcional y orientada a objetos. La sintaxis es similar a la de Matlab y GNU Octave. En [24] hay un manual explicativo de julia. Se puede descargar desde: `http://julialang.org/downloads/`. Algunos de los script de

ejecución están realizados en este lenguaje, como se explicará en cada caso.

3.1.2. Preparación de los datos de entrenamiento

Paso 1 - Creación de la gramática

La gramática de reconocimiento en Julius[1] se divide en dos archivos, el archivo con extensión .grammar en el cual se definen un conjunto de reglas con las que se norman las expresiones lingüísticas (en este caso órdenes) que se espera reconocer y el archivo con extensión .voca que contiene la pronunciación o fonema de las palabras que se van a reconocer, las cuales se encuentran definidas por las categorías en el archivo .grammar.

Formato de los archivos .grammar y .voca

- .grammar

En este archivo se definen el conjunto de reglas que rigen las palabras permitidas y las expresiones que se esperan reconocer, utilizando un formato BNF modificado escrito de la siguiente forma:

```
Sentencia: [Expresión con sentencias]
```

donde "Sentencia" es un no terminal y "[Expresión con sentencias]" es un conjunto de secuencias de expresiones que pueden ser terminales y/o no terminales.

A continuación se muestra un fragmento del archivo .grammar creado para el sistema de órdenes de voz:

```
S: SN_B SENT SN_E
SENT: NOMBRE
SENT: ACCION EDITOR DE EDICION
SENT: ACCION PROGRAMA_O
SENT: ACCION  CARPETA ARCHIVO
SENT: ACCION  CARPETA DE ARCHIVO
SENT: ACCION_V LA VENTANA
```

donde “S”és el inicio del símbolo de frase, “SN_B”, “SN_E” son necesarios en toda gramática de Julius[1]. “SN_B”, “SN_E”, “NOMBRE”, “ACCION”, “EDITOR”, “DE”, “EDICION”, “PROGRAMA_O”, “CARPETA”, “ARCHIVO”, entre otros son terminales que representan categorías de palabras en el archivo .voca, “SENT” son no terminales y no tienen una definición en el archivo .voca.

- .voca

En este archivo se definen las palabras con su pronunciación para cada categoría. El símbolo “%” (porcentaje) define cada categoría, las definiciones del conjunto de palabras de cada categoría son definidas en la línea siguiente, una por línea, donde la primera columna representa la cadena que se mostrará cuando es reconocida y la segunda su pronunciación. El formato sería el siguiente:

```
% CATEGORIA
palabra_1       pronunciación_1
palabra_2       pronunciación_2
palabra_3       pronunciación_3
...     ...
...     ...
```

a continuación un fragmento del archivo .voca creado para el sistema (gpyvozcontrol):

```
% SN_B
<s>         sil

% SN_E
</s>         sil

% ACCION
abrir a b r i r
abre a b r e
```

```
% ACCION_V
cerrar th e rr a r
cierra th i e rr a
maximizar m a k s i m i th a r
maximiza m a k s i th a
minimizar m i n i m i th a r
minimiza m i n i m i th a
ampliar a m p l i a r
amplia a m p i a
reducir r e d u th i r
reduce r e d u th e

% EDICION
presentacion p r e s e n t a th i o n
texto t e k s t o
documento d o k u m e n t o s
hoja o x a

% PROGRAMA\_O
calculadora k a l k u l a d o r a
navegador n a b e g a d o r
mensajeria m e n s a x e r i a
terminal t e r m i n a l
consola k o n s o l a
correo k o rr e o

% NOMBRE
vozcontrol b o th k o n t r o l
```

Las categorías “SN_B”, “SN_E” son un modelo de silencio definidos en el modelo acústico y representan el inicio y el final de la entrada de voz; “NOMBRE”, “ACCION”, “EDITOR”, “EDICION”, “PROGRAMA_O”, entre otros tiene definida las palabras

con sus respectivas pronunciaciones, que son los fonemas que serán reconocidos y que deben estar en el modelo acústico.

Julius[1] necesita de un archivo, donde cada transición de palabra a palabra esta en una lista de forma explícita, esto se logra compilando los archivos .grammar y .voca. mkdfa.pl genera este archivo buscando la sentencia inicial "S" en el archivo .grammar y reemplazando todas las categorias por las palabras que se encuentran definidas en el archivo .voca, formando una lista predefinida de todas las posibles combinaciones de las frases a reconocer.

Cuando Julius[1] escucha los sonidos de una palabra o frase de un usuario, lo que se intenta hacer es coincidir estos sonidos con la representación estadistíca del modelo acústico, si se produce una coincidencia, Julius determina el fonema correspondiente al sonido. Se realiza un seguimiento de los fonemas que coinciden hasta que el usuario haga una pausa. Entonces es cuando se busca la gramática equivalente de los fonemas. Por ejemplo, si ha reconocido los fonemas " a b r i r n a b e g a d o r ", devolverá las palabras "<sil> abrir navegador </sil>" en el manejador de llamada.

Para obtener los archivos ".term", ".dfa" y ".dic" se utiliza el siguiente comando:

```
mkdfa.pl gpyvozcontrol
```

donde ".term" y ".dfa" obtenidos contienen información del automáta finito y ".dic" la pronunciación del diccionario, todos en formato Julius[1].

La representación de la gramática se muestra a continuación:

(SENT-START (vozcontrol | $ACCION (editor de $EDICION | $PROGRAMA_O | carpeta $ARCHIVO | reproductor) | $ACCION_V [todas] $LA ventanas] | $ACCION_R $LAMINA | $ACCION_RP volumen al (reproductor | $SISTEMA) | título de la $ACTIVO | $DECIR la $DATE | $ACCION_L $LAMINA | $ACCION_RP_1 el reproductor | $ACCION_S $SISTEMA | presentar $TESIS | $LAMINA $ACCION_R | cambiar a la lámina ($DIGIT5 | $DIGIT9 | $DIGIT10 | $DIGIT100 [y ($DIGIT5 | $DIGIT9)]) | $ACCION_A $ACCESIBILIDAD) SENT-END)
$ACCION = abrir | abre;
$ACCION_V = cerrar | cierra | maximiza[r] | minimiza[r] | amplía[r] | reducir | reduce;
$EDICION = presentaci|ó|n | texto | documento | hoja;

$PROGRAMA_O = calculadora | navegador | mensajería | terminal | consola | correo;
$SISTEMA = equipo | sistema;
$ACCION_S = apagar | reiniciar;
$ARCHIVO = personal | videos | música | descargas | documentos;
$LA = la | las;
$ACCION_R = siguiente | anterior | detener | reproducir;
$ACCION_L = primera | última | cambiar | próxima;
$ACCION_RP = bajar | subir | silenciar;
$ACCION_RP_1 = cerrar | cierra | pausa[r];
$LAMINA = lámina | canción;
$TESIS = tesis | proyecto;
$ACTIVO = canción | ventana;
$DECIR = decir | dime;
$DIGIT5 = uno | dos | tres | cuatro | cinco;
$DIGIT9 = seis | siete | ocho | nueve;
$DIGIT10 = diez | once | doce | trece | catorce | quince;
$DIGIT100 = veinte | treinta | cuarenta | cincuenta | sesenta | setenta | ochenta | noventa;
$ACCION_A = activa[r] | desactiva[r];
$ACCESIBILIDAD = orca | teclado | magnificador;
$DATE = hora | fecha;

donde las "|" (barras verticales) son elementos alternativos, los "[ ]'(corchetes)'son elementos opcionales y "< >'son elementos que se repiten'.

Paso 2 - Creación del diccionario de pronunciación

En primer lugar se crea un archivo "prompts.txt" que contiene los fonemas que se utilizan en nuestra gramática y los fonemas necesarios para crear un diccionario foneticamente balanceado. A continuación se muestra algunas líneas del archivo "prompts.txt" de las 282 frases que contiene este archivo, que fue creado para el sistema de órdenes de voz (gpyvozcontrol):

```
*/01 entre los libros infantiles más leídos está el cuento
   de caperucita roja
*/02 el inspector carlos ha tenido mucho peso en la
   resolución del caso
*/03 a lo lejos se divisaba una luz azulada que se movía en
    zigzag
*/04 siempre se ha dicho que por la boca muere el pez
*/05 clavó el gusano en el anzuelo
*/06 las autoridades sanitarias advierten que el tabaco
   perjudica seriamente la salud
*/07 en el techo el yeso estaba agrietado
*/08 mi madre me ha dicho que coma todo lo que tengo en el
   plato
*/09 el mago parecía de goma se metió en la caja en un
   abrir y cerrar de ojos
*/10 el padre dio como dote unos zapatos y una colección de
    sellos muy cara
...
...
```

Es de acotar que cada frase se grabo en un audio diferente, esto se explica mejor en la sección 3.3.
En segundo lugar se genera un archivo "wList" que contiene la lista de palabras que se encuentran en el archivo "prompts.txt", el script "prompts2wlist.jl" crea este archivo. Se ejecuta:

```
julia ../bin/prompts2wlist.jl prompts.txt wList
```

lo que genera el archivo "wlist".

Nota: el script agrega las siguientes palabras en orden a la lista:

–SENT–END

–SENT–START

que son entradas internas necesarias para la creación y proceso del modelo acústicocon Julius.

El siguiente paso es añadir información de la pronunciación, es decir, los fonemas que componen la palabra, a cada una de las palabras en el archivo "wList". HTK[6] utiliza el comando HDMan que pasa por el archivo "wList", y busca la pronunciación de cada palabra en un archivo de léxico independiente, y dando como resultado la creación del diccionario de pronunciación.

Así que se crea el diccionario de pronunciación, ejecutando el comando mencionado anteriormente:

```
HDMan -A -D -T 1 -m -w wList -n -i monophones1 -l dlog dict ../
   lexicon/VoxForgeDict.txt
```

suponiendo que el archivo "VoxForgeDict.txt" es un diccionario de pronunciación y que contiene todas las palabras que están en el archivo "prompts.txt".

Esto genera el archivo "dict" que es el diccionario de pronunciación necesario para el modelo acústico y "monophones1" que es simplemente los fonemas utilizados en "dict".

Por último se crea otro archivo "monophones" para pasos posteriores. Solo se copia el archivo "monophones1" a otro nuevo archivo "monophones0" y se elimina la pausa corta ''sp'del archivo "monophones0", esto se hace de la siguiente forma:

```
cat monophones1 | grep -v sp >monophones0
```

Paso 3 - Grabaciones

Revisar la sección 3.3 de este mismo capítulo.

Paso 4 - Creación de los archivos de transcripción

El kit de herramientas de HTK[6] no puede procesar directamente el archivo "prompts.txt", por esta razón se crea el archivo de transcripción el cual tiene el siguiente

formato:

```
"*/01.lab"
entre
los
libros
infantiles
más
leídos
está
el
cuento
de
caperucita
roja
.
```

y así para cada frase del archivo “prompts.txt”. El script “prompts2mlf.jl” permite hacer esto automaticamente, el proceso se describe a continuación.

Se crea el archivo de transcripción “words.mlf”, ejecutando en consola:

```
julia ../bin/prompts2mlf.jl prompts.txt words.mlf
```

Se amplía el archivo de transcripción “words.mlf”. Es decir, se cambia cada palabra con su fonema y colocando la salida en un nuevo archivo, esto se consigue comparando el archivo mlf con el diccionario creado anteriormente, lo que genera el archivo “phones0.mlf” y que no tendrá pausas cortas ‘sp’después de cada grupo de palabras.

En primer lugar se crea el archivo “mkphones.led” con lo siguiente:

```
EX
ES sil sil
DE sp
```

Nota: este archivo lleva una línea en blanco al final

y luego se ejecuta:

```
HLEd -A -D -T 1 -l '*' -d -i dict phones0.mlf mkphones0.led
    words.mlf
```

esto genera el archivo "phones0.mlf".

A continuación, se crea un segundo archivo "phones1.mlf" (que incluye pausas cortas ('sp') después de cada grupo depalabra). Se Crea el archivo "mkphones1.led" con lo siguiente:

```
EX
ES sil sil
```

Nota: igual que en el archivo anterior, este archivo tiene una línea en blanco al final y se ejecuta así:

```
HLEd -A -D -T 1 -l '*' -d -i dict phones1.mlf mkphones1.led
    words.mlf
```

con lo que se obtiene el archivo "phones1.mlf".

Paso 5 - Codificación de los datos (audios) de entrenamiento

HTK[6] no es eficiente en el procesamiento de archivos ".wav" como lo es con su formato interno, por lo tanto se debe convertir los archivos ".wav" al formato ".mfcc (Mel Frequency Cepstral Coefficients)"[25] que hace referencia generalmente a "Vectores de características" (conjunto de parámetros espectrales, que se obtienen de pequeños segmentos de audio)[26]

El comando HCopy realiza la conversión de formato "wav" a "mfcc"[25]. Para ello, se requiere un archivo de configuración (config) que especifica todos los parámetros de conversión necesarios. Se crea el archivo "wav-config" que contiene:

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC\_0\_D
```

```
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WindowSize = 250000.0
USEHAMMING = T
PREEMCOEF = 0,97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
```

y el script "codetrain.scp" que contiene la ruta de origen de los archivos "wav" y de destino "mfcc"[25] los cuáles, para el sistema de órdenes gpyvozControl, lucen así (esto es solo un fragmento. Habrá tantos como archivos wav se estén procesando) :

```
../train/wav/quinteron/01.wav ../train/mfcc/quinteron/01.mfc
../train/wav/quinteron/02.wav ../train/mfcc/quinteron/02.mfc
../train/wav/quinteron/03.wav ../train/mfcc/quinteron/03.mfc
../train/wav/quinteron/04.wav ../train/mfcc/quinteron/04.mfc
../train/wav/quinteron/05.wav ../train/mfcc/quinteron/05.mfc
...
...
../train/wav/guzmanl/01.wav ../train/mfcc/guzmanl/01.mfc
../train/wav/guzmanl/02.wav ../train/mfcc/guzmanl/02.mfc
../train/wav/guzmanl/03.wav ../train/mfcc/guzmanl/03.mfc
../train/wav/guzmanl/04.wav ../train/mfcc/guzmanl/04.mfc
../train/wav/guzmanl/05.wav ../train/mfcc/guzmanl/05.mfc
...
...
../train/wav/ramirezm/01.wav ../train/mfcc/ramirezm/01.mfc
../train/wav/ramirezm/02.wav ../train/mfcc/ramirezm/02.mfc
../train/wav/ramirezm/03.wav ../train/mfcc/ramirezm/03.mfc
../train/wav/ramirezm/04.wav ../train/mfcc/ramirezm/04.mfc
```

../train/wav/ramirezm/05.wav ../train/mfcc/ramirezm/05.mfc

...

...

luego se ejecuta:

```
HCopy -A -D -T 1 -C wav\_config -S codetrain.scp
```

El resultado es la creación de una serie de archivos "mfc"[25] correspondientes a los archivos que aparecen en el script codetrain.scp en la carpeta "train/mfcc/".

3.1.3. Creación de los monofónos en los HMMS

Paso 6 - Creación de los Flat Start Monophones (Monófonos planos de inicio)

El primer paso en la formación de un hmm[23] (Hidden Markov Model), es la definición de un modelo prototipo, el énfasis es crear una estructura de modelo. Los parámetros no son importantes al comienzo, su propósito es definir la topología del modelo.

Se crea el archivo "proto" con lo siguiente:

õ <VecSize> 25 <MFCC_0_D_N_Z>

h̃ "proto"

<BeginHMM>

<NumStates> 5

<State> 2

<Mean> 25

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 25

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State> 3

<Mean> 25

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 25

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<State> 4

<Mean> 25

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

<Variance> 25

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

<TransP> 5

0.0 1.0 0.0 0.0 0.0

0.0 0.6 0.4 0.0 0.0

0.0 0.0 0.6 0.4 0.0

0.0 0.0 0.0 0.7 0.3

0.0 0.0 0.0 0.0 0.0

<EndHMM>

Para mayor información sobre este archivo, se recomienda leer el libro de HTK[27], la sección 3.2.1.

También es necesario un archivo de configuración, por lo que se crea un archivo llamado “config” que contiene:

```
TARGETKIND = MFCC\_0\_D\_N\_Z
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
```

HTK[6] necesita saber donde están ubicados todos los archivos de vectores de

caracteristicas, que son los archivos “mfc” creados anteriormente, por lo que se crea un script “train.scp” que contiene donde esté el origen de estos archivos. Para el sistema de órdenes de voz luce así:

```
../train/mfcc/quinteron/01.mfc
../train/mfcc/quinteron/02.mfc
../train/mfcc/quinteron/03.mfc
../train/mfcc/quinteron/04.mfc
../train/mfcc/quinteron/05.mfc
...
...
../train/mfcc/guzmanl/01.mfc
../train/mfcc/guzmanl/02.mfc
../train/mfcc/guzmanl/03.mfc
../train/mfcc/guzmanl/04.mfc
../train/mfcc/guzmanl/05.mfc
...
...
../train/mfcc/ramirezm/01.mfc
../train/mfcc/ramirezm/02.mfc
../train/mfcc/ramirezm/03.mfc
../train/mfcc/ramirezm/04.mfc
../train/mfcc/ramirezm/05.mfc
...
...
```

En el siguiente paso se crea una carpeta “hmm0” y a continuación una nueva versión del archivo “proto” dentro de la carpeta recién creada (“hmm0”) utilizando la herramienta “HCompV” de HTK[6] ejecutando:

```
HCompV -A -D -T 1 -C config -f -m 0,01 -S train.scp -M hmm0
   proto
```

esto genera los archivos “proto” y “vFloors” en la carpeta “hmm0”.

Para formar los monófonos, solo se crean los archivos “hmmdefs” y “macros”, los cuáles se construyen de la siguiente forma:

- Archivo “hmmdefs”.

Se copia el archivo “monophones0” a la carpeta “hmm0” y se renombra como “hmmdefs”; para cada fonema se hace:

– Añadir al principio de cada fonema: ‘~h ’(sin los apostrofes y dejando un espacio luego de la h)

– Poner el fonema entre comillas dobles.

– Copiar de la linea 5 en adelante del archivo “proto” que se encuentra en la carpeta “hmm0” y pegarlo después de cada fonema.

– Dejar una linea en blanco al final del archivo.

Con esto se crea el archivo “hmmdefs”, que contiene los monófonos planos de inicio, donde se tiene la probabilidad inicial que representa el estado donde se cree que se encuentra inicialmente el hmm, la probabilidad de transición que representa el cambio en la cadena de Márkov y la probabilidad de emisión que representa con cuanta probabilidad aparece el siguiente fonema. A continuación se muestrael monófono ‘sil’:

```
~h "sil"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 25
 8.909576e-09 -3.484646e-09 -2.725015e-09 1.818136e-09 -1.063415e-08
    2.443322e-08 1.127184e-08 6.560171e-10 -2.626804e-09
    -1.013708e-09 -3.878243e-09 8.729842e-09 -2.313642e-04 4.048051e-04
    -4.219529e-05 -4.874423e-04 -6.414966e-05 -1.043901e-03 -9.296974e-04
    -6.693655e-04 1.530694e-04 -2.798930e-04 1.348085e-03 -2.755471e-04
    -6.417893e-04
<VARIANCE> 25
 4.503074e+01 2.756504e+01 3.618806e+01 3.808241e+01 3.697804e+01
    5.397874e+01 4.106139e+01 3.781839e+01 4.048951e+01 3.191657e+01
```

```
    2.334843e+01 2.268922e+01 1.443263e+00 1.474413e+00 1.613365e+00
    1.815563e+00 1.749648e+00 2.556612e+00 2.102865e+00 1.877820e+00
    1.969723e+00 1.811243e+00 1.592847e+00 1.414474e+00 1.156754e+00
<GCONST> 9.561387e+01

<STATE> 3

<MEAN> 25

 8.909576e-09 -3.484646e-09 -2.725015e-09 1.818136e-09 -1.063415e-08
    2.443322e-08 1.127184e-08 6.560171e-10 -2.626804e-09
    -1.013708e-09 -3.878243e-09 8.729842e-09 -2.313642e-04 4.048051e-04
    -4.219529e-05 -4.874423e-04 -6.414966e-05 -1.043901e-03 -9.296974e-04
    -6.693655e-04  1.530694e-04 -2.798930e-04 1.348085e-03 -2.755471e-04
    -6.417893e-04
<VARIANCE> 25

 4.503074e+01 2.756504e+01 3.618806e+01 3.808241e+01 3.697804e+01
    5.397874e+01 4.106139e+01 3.781839e+01 4.048951e+01 3.191657e+01
    2.334843e+01 2.268922e+01 1.443263e+00 1.474413e+00 1.613365e+00
    1.815563e+00 1.749648e+00 2.556612e+00 2.102865e+00 1.877820e+00
    1.969723e+00 1.811243e+00 1.592847e+00 1.414474e+00 1.156754e+00
<GCONST> 9.561387e+01}<STATE>4<MEAN>25\small 1.909576e-09
   -3.484646e-09 -2.725015e-09 1.818136e-09 -1.063415e-08
   2.443322e-08 1.127184e-08 6.560171e-10 -2.626804e-09
   -1.013708e-09 -3.878243e-09 8.729842e-09 -2.313642e-04
   4.048051e-04 -4.219529e-05 -4.874423e-04 -6.414966e-05
   -1.043901e-03 -9.296974e-04 -6.693655e-04 1.530694e-04
   -2.798930e-04 1.348085e-03 -2.755471e-04 -6.417893e-04}
   <VARIANCE>25{\small }4.503074e+01 2.756504e+01 3.618806e
   +01 3.808241e+01 3.697804e+01 5.397874e+01 4.106139e+01
   3.781839e+01 4.048951e+01 3.191657e+01 2.334843e+01
   2.268922e+01 1.443263e+00 1.474413e+00 1.613365e+00
   1.815563e+00 1.749648e+00 2.556612e+00 2.102865e+00
   1.877820e+00 1.969723e+00 1.811243e+00 1.592847e+00
   1.414474e+00 1.156754e+00
<GCONST> 9.561387e+01}<TRANSP>5{\small 0.000000e+00 1.000000e
   +00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 6.000000e-01 4.000000e-01 0.000000e+00
    0.000000e+00
 0.000000e+00 0.000000e+00 6.000000e-01 4.000000e-01
    0.000000e+00
```

```
0.000000e+00 0.000000e+00 0.000000e+00 7.000000e-01
   3.000000e-01
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
   0.000000e+00}<ENDHMM>
```

- Archivo “macros”.

Se crea un archivo “macros” en la carpeta “hmm0” copiando a “vFloor”; se copia también las tres primeras lineas del archivo “proto” ubicado en la carpeta “hmm0” y se agrega al principio del archivo “macros”.

Para hacer la reestimación de los monófonos, se crean nueve carpetas hmm consecutivas (hmm1:hmm9). Cada carpeta contendrá una versión mejor elaborada de los modelos ocultos de Markov que están siendo entrenados para el reconocedor que se desarrolla.

Los Monofonos al comienzo son re-calculadas utilizando la herramienta HERest. El propósito de esto es cargar todos los modelos en la carpeta hmm0 (éstos están contenidos en el archivo “hmmdefs”), y volver a evaluarlos utilizando los archivos MFCC enumerados en el script “train.scp”, y crear un nuevo modelo que figura en “hmm1”. Ejecutando el comando HERest:

```
HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0
   1000.0 -S train.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1
   monophones0
```

esto genera los archivos “hmmdefs” y “macros” en la carpeta “hmm1”, se repiten estos pasos para la carpeta “hmm2” y “hmm3” respectivamente para la creación de nuevos conjuntos de modelos:

```
HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0
   1000.0 -S train.scp -H hmm1/macros -H hmm1/hmmdefs -M hmm2
   monophones0
```

esto genera los archivos “hmmdefs”’y “macros” en la carpeta “hmm2”.

```
HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0
   1000.0 -S train.scp -H hmm2/macros -H hmm2/hmmdefs -M hmm3
   monophones0
```

y se obtiene los archivos “hmmdefs” y “macros” en la carpeta “hmm3”

Paso 7 - Fijar los silencios en el modelo

Los modelos hmmdefs creados hasta ahora no incluyen una pausa corta ‘sp’, que se producen entre las palabras en el habla normal. Sin embargo, se ha creado un modelo de silencio ‘sil’de mayor duración y hace referencia a las pausas que se producen al final de la oración.

Para incluir la pausa corta “sp”, se agrega un estado “sp” en los modelos “hmmdefs”, el cual utiliza el centro de los estados “sil”, luego ambos deben estar atados. Esto se hace mediante la copia del estado principal del modelo “sil” en el archivo hmmdefs, se agrega el estado “sp”, y luego se ejecuta una herramienta especial llamada HHeD de “lazo” del estado “sp” al estado “sil” para que compartan el mismo estado principal.

Para el sistema de órdenes de voz, se copia el contenido de la carpeta “hmm3” a “hmm4”, se modifica el archivo “hmmdefs” ubicado en “hmm4” añadiendo un nuevo estado “sp” al final, de la siguiente forma:

- Se copia y pega el estado “sil”, se cambia “sil” al nuevo “sp” (no se elimina el viejo estado “sil”, ya que se necesita).
- Se elimina el estado 2 y 4 del nuevo estado “sp” (es decir, se mantiene el estado principal del viejo estado “sil” en el nuevo estado “sp”).
- Se cambia <NUMSTATES> a 3
- Se cambia <STATE> a 2
- Se cambia <TRANSP> a 3
- Se cambia la matriz en <TRANSP> a 3 by 3 array

– Se cambia los números de la matriz a: 0,0 1,0 0,0
0,0 0,9 0,1
0,0 0,0 0,0

A continuación se crea el archivo “sil.hed”, el cual contiene lo siguiente:

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

la última linea es el comando de atadura.

Se ejecuta (utilizando monophones1, ya que contiene el estado ‘sp’):

```
HHEd -A -D -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed
   monophones1
```

para atar los estados “sp” a los estados principales “sil”. Esto genera los archivos “hmmdefs” y “macros” dentro de la carpeta “hmm5”.

Se ejecuta dos veces mas con el comando “HERest” y utilizando el archivo monophones1:

```
HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0
   3000.0 -S train.scp -H hmm5/macros -H  hmm5/hmmdefs -M hmm6
   monophones1
```

el comando genera los archivos “hmmdefs” y “macros” en la carpeta “hmm6”.

```
HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0
   3000.0 -S train.scp -H hmm6/macros -H  hmm6/hmmdefs -M hmm7
   monophones1
```

y se obtienen los archivos “hmmdefs” y “macros” en la carpeta “hmm7”.

Paso 8 - Realineación de los datos de entrenamiento

Esta operación es similar a la operación llevada a cabo en el paso 4. Sin embargo, en este caso el comando "HVite" puede considerar todas las pronunciaciones de cada palabra (en el caso en que una palabra tiene más de una pronunciación), y luego usar la salida de la pronunciación que mejor coincide con los datos acústicos.

Se ejecuta el comando "HVite" de la siguiente forma:

```
HVite -A -D -T 1 -l '*' -o SWT -b SENT-END -C config -H hmm7/
   macros -H hmm7/hmmdefs -i aligned.mlf -m -t 250.0 150.0
   1000.0 -y lab -a -I words.mlf -S train.scp dict monophones1>
   hvite\_log
```

y se obtiene el archivo "aligned.mlf".

Se ejecuta "HERest" para "hmm8" y "hmm9":

```
HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0
   3000.0 -S train.scp -H hmm7/macros -H hmm7/hmmdefs -M hmm8
   monophones1
```

esto genera el archivo "hmdefs" y "macros" en la carpeta "hmm8".

```
HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0
   3000.0 -S train.scp -H hmm8/macros -H hmm8/hmmdefs -M hmm9
   monophones1
```

y se genera el archivo "hmdefs" y "macros" en la carpeta "hmm9".

3.1.4. Creación de los estados de ligadura (atadura) de los trifonemas

Paso 9 - Hacer trifonemas a partir de los fonemas

En el diccionario creado en el paso 2, la pronunciación de una palabra fue dada por una serie de fonemas (también llamados monofonemas, es decir, un solo fonema). Para

generar un trifonema (un grupo de 3 fonemas) se usa una la declaración de fonemas, con un formato de tres letras. el fonema “L” (el fonema de la izquierda) precede del fonema “X” y el fonema “R” (el fonema de la derecha) sigue al fonema “X”. El trifonema se declara en forma de “L - X + R”.

A continuación se muestra un ejemplo del trifonema de la palabra “texto”, la primera linea muestra el fonema y la segunda linea el trifonema:

```
textot e s k t o
texto t+e t-e+s e-s+k s-k+t k-t+o t-o
```

hay que tener en cuenta que también se pueden obtener bifonemas (un grupo de 2 fonemas) al principio y al final de la palabra.

Para convertir las transcripciones fonéticas en el archivo “aligned.mlf” creado en el paso 8 a un conjunto equivalente de transcripciones de trifonemas, se ejecuta el comando “HLEd”, el cual se utiliza para generar una lista de todos los trifonemas para los cuales existe al menos un ejemplo en los datos de entrenamiento.

Pero antes, se crea el archivo “mktri.led”, con lo siguiente:

```
WB sp
WB sil
TC
```

luego se ejecuta el editor de archivos de etiquetas (HLEd):

```
HLEd -A -D -T 1 -n triphones1 -l '*' -i wintri.mlf mktri.led
   aligned.mlf
```

se obtiene los archivos “wintri.mlf” (archivo de trifonos de multi etiquetas) y “triphones1 (lista de trifonos en su conjunto de entrenamiento)”

A continuación se ejecuta el archivo “mktrihed.jl”, de la siguiente forma:

```
../julia/bin/julia ../bin/mktrihed.jl monophones1 triphones1
   mktri.hed
```

Esto genera el archivo “mktri.hed”. Este archivo contiene un clon de comando ‘CL’seguido de una serie de comandos, para que compartan el mismo conjunto de parámetros.

Se crea tres carpetas mas “hmm10”, “hmm11” y “hmm12” y se ejecuta el comando “HHEd” (es el editor de definiciones de hmm en HTK y se utiliza principalmente para la aplicación de “ataduras” a través de parámetros HMM seleccionadas):

```
HHEd -A -D -T 1 -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.
   hed monophones1
```

lo que genera los archivos “hmmdes” y “macros” en la carpeta “hmm10”.

Se ejecuta “HERest” dos veces mas:

```
HERest  -A -D -T 1 -C config -I wintri.mlf -t 250.0 150.0
   3000.0 -s stats -S train.scp -H hmm10/macros -H hmm10/hmmdefs
    -M hmm11 triphones1
```

esto genera los archivos “hmmdes” y “macros” en la carpeta “hmm11”.

```
HERest  -A -D -T 1 -C config -I wintri.mlf -t 250.0 150.0
   3000.0 -s stats -S train.scp -H hmm11/macros -H hmm11/hmmdefs
    -M hmm12 triphones1
```

y se obtiene los archivos “hmmdes” y “macros” en la carpeta “hmm12”.

Paso 10 - Hacer trifonemas Estado-Atado

Según el libro de HTK[27], una limitación de la creación de un modelo acústico basado en trifonemas en el paso 9 es que no trata con trifonemas para los que no existen ejemplos en los datos de entrenamiento. Este problema se puede aliviar mediante el diseño cuidadoso de la base de datos de entrenamiento, tratando de anticipar todos los trifonemas necesarios. Pero en la construcción de sistemas de grandes vocabularios los trifonemas invisibles son inevitables”.

El árbol de decisión utilizado aquí permite reconocer trifonemas invisibles para ser sintetizados mediante el uso de un arbol de decisión fonética. Se organizan los

modelos en un árbol y los parámetros que se pasan al árbol son llamados preguntas. El decodificador hace una pregunta sobre el contexto y decide cuál es el modelo a utilizar. "Un árbol de decisión fonética es un árbol binario en el que una pregunta sí/no fonética se adjunta a cada nodo. La pregunta en cada nodo se elige para maximizar la probabilidad de los datos de entrenamiento, dado el conjunto final de estados atados. Los árboles se definen por la TB de comandos. Todas las posibles cuestiones fonéticas debe cargar en HHED mediante comandos de QS" (es una traducción libre del tutorial [5] del paso 10).

A continuación, se crea un archivo "maketriphone.ded" que contenga:

```
AS sp
MP sil sil sp
TC
```

El comando TC dirige la edición "HDMan" a la salida de trifonemas.

Se ejecuta para todo el diccionario léxico, no solo para el diccionario utilizado hasta ahora, lo siguiente:

```
HDMan -A -D -T 1 -b sp -n fulllist0 -g maketriphones.ded -l
   flog dict-tri ../lexicon/VoxForgeDict.txt
```

y se obtiene los archivos "dict-tri" y "fulllist0"

Para anexar el contenido de "monophones0" al principio del archivo "fulllist0", eliminar las entradas duplicadas y colocar el resultado en "fulllist", se ejecuta el script "fixfulllist.jl" como sigue:

```
julia ../bin/fixfulllist.jl fulllist0 monophones0 fulllist
```

Ahora se crea el archivo "tree1.hed" (que contiene el contexto de preguntas que utilizara HTK para seleccionar los trifonemas pertinentes), puede encontrar mas información sobre como se crea este archivo desde [28] y [29]. Para el sistema de órdenes de voz se utiliza el encontrado en [22].

Nota: el archivo "tree1.hed" contiene una linea en blanco al final.

Se ace una copia del archivo “tree1.hed” con el nombre de “tree.hed”. Consultar el libro HTK[27] para obtener más información sobre estos comandos.

Para agregar los grupos estatales al archivo “tree.hed” creado anteriormente, se ejecuta el archivo “mkclscript.jl”:

```
julia ../bin/mkclscript.jl monophones0 tree.hed
```

Ahora se crea tres carpetas mas “hmm13”, “hmm14” y “hmm15” y se ejecuta “HHEd”:

```
HHEd -A -D -T 1 -H hmm12/macros -H hmm12/hmmdefs -M hmm13 tree.
   hed triphones1
```

esto genera los archivos “hmmdefs”, “macros” y “tiedlist”.

Por último se ejecuta dos veces el comando “HERest”:

```
HERest -A -D -T 1 -T 1 -C config -I wintri.mlf  -t 250.0 150.0
   3000.0 -S train.scp -H hmm13/macros -H hmm13/hmmdefs -M hmm14
    tiedlist
```

esto genera los archivos “hmmdefs” y “macros” en la carpeta “hmm14”.

```
HERest -A -D -T 1 -T 1 -C config -I wintri.mlf  -t 250.0 150.0
   3000.0 -S train.scp -H hmm14/macros -H hmm14/hmmdefs -M hmm15
    tiedlist
```

y se obtiene los archivos “hmmdefs” y “macros” en la carpeta “hmm15”.

El archivo de “hmmdefs” en la carpeta “hmm15” junto con el archivo “tiedlist”, ahora se utiliza con Julius para reconocimiento de voz.

3.1.5. Ejecución de Julius

Antes de ejecutar Julius[1], se debe crear un archivo de configuración para la aplicación, en este caso “gpyvozcontrol.jconf” donde sus principales parámetros son:

– dfa gpyvozcontrol.dfa# archivo de gramática del autómata de estado finitos.

- v gpyvozcontrol.dict# Diccionario de pronunciación
- h hmm15/hmmdefs# modelo acústico
- hlist tiedlist
- smpFreq 16000# Velocidad de muestreo
- spmodel 'sp'# nombre de una pausa corta
- multipath# force enable MULTI-PATH model handling
- gprune safe# Metodo de poda de gauss
- iwcd1 max# Metodo de aproximación de trifonemas
- iwsppenalty -70.0# transition penalty for the appended sp models
- iwsp# append a skippable sp model at all word ends
- penalty1 5.0# word insertion penalty for grammar (pass1)
- penalty2 20.0# word insertion penalty for grammar (pass2)
- b2 200# beam width on 2nd pass (#words)
- sb 200.0# score b1eam envelope threshold
- n 1# num of sentences to find

y se ejecuta así:

```
julius -input mic -C gpyvozcontrol.jconf
```

Los primeros 2-3 segundos del discurso no son reconocidos, mientras Julius[1] ajusta sus niveles de reconocimiento. A continuaci—ó—n, se reconocen las frases de la gramática creada en el paso 1.

En este punto se ha logrado construir la gramática y el modelo acústico necesarios para el desarrollo del Sistema de órdenes de voz "gpyvozcontrol".

3.2. Creación del modelo acústico con CMUSphinx

3.2.1. Bibliotecas (libraries) de software necesarias

Los programas necesarios para crear este modelo acústico con el kit de herramientas de CMUSphinx[3] y utilizados para este proyecto son:

1. SphinxBase Es un conjunto de bibliotecas utilizadas para los proyectos de CMUSphinx. Se puede descargar desde: `http://sourceforge.net/projects/cmusphinx/files/sphinxbase/5prealpha`.

2. SphinxTrain Es una suite de herramientas que llevan a cabo el entrenamiento del modelo acústico. Se puede descargar desde: `http://sourceforge.net/projects/cmusphinx/files/sphinxtrain/5prealpha`.
   Se recomienda descargar el tar.gz desde: `http://cmusphinx.org/download/nightly/SphinxTrain.nightly.tar.gz`

3. Sphinx3 Es un reconocedor de voz ligeramente más lento que Sphinx 2 pero más preciso. Se puede descargar con svn desde: `https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/sphinx3`

4. PocketSphinx Es un reconocedor de voz que puede ser utilizado en sistemas embebidos. Se puede descargar desde: `http://sourceforge.net/projects/cmusphinx/files/pocketsphinx/5prealpha`

en [30] están los pasos a seguir para instalar dichas bibliotecas.

3.2.2. Creación de la gramática

Para utilizar una gramática en "gpyvozcontrol" con CMUSphinx[3], se crea el archivo "gpyvozcontrol.gram'y que contiene:

```
#JSGF V1.0;

grammar gpyvozcontrol;

public <gpyvozcontrol-listens> = <comando>;

<comando> = vozcontrol | <accion> (editor de <edicion> | <
    programa_o> | carpeta <archivo>) | <accion_v> | <accion_v>
    (ventana | todas las ventanas) | <lamina> <accion_r> | <
    accion_l> <lamina> | <accion_rp> volumen al <sistema> | <
    accion_rp_1> al reproductor | <accion_s> (sistema | equipo)
     |  presentar (proyecto| tesis) | <accion_a> <
    programa_accesibilidad> | <cambiar> a la lamina (<numero5>
    | <numero9> | <numero10> | dieci <numero9> | veinte |
    veinti (<numero5> |<numero9>) | <numero100> | <numero100> y
     (<numero5> | <numero9>)) | <decir> <date> | titulo de la (
    cancion | ventana);
<accion> = abrir | abre ;
<accion_v> = cerrar | cierra | minimizar | minimiza |
    maximizar | maximiza;
<accion_r> = siguiente | anterior  | actual;
<accion_l> =  primera | ultima ;
<accion_rp> = bajar | subir ;
<accion_rp_1> = cierra | cerrar | pausa | pausar ;
<accion_s> = apagar | reiniciar ;
<accion_a> = activar | activa | desactivar | desactiva ;
<archivo> = videos | musica | documentos | descargas |
    personal ;
<lamina> = cancion | lamina ;
<edicion> = texto | documento | hoja | presentacion ;
```

```
<programa_o> = terminal | consola | texto | documento |
   calculadora | navegador | reproductor ;
<sistema> = equipo | reproductor | sistema ;
<programa_accesibilidad> = orca | teclado | magnificador ;
<cambiar> = cambia | cambiar;
<numero5> = uno | dos | tres | cuatro | cinco;
<numero9> = seis | siete | ocho | nueve;
<numero10> = diez | once | doce | trece | catorce | quince;
<numero100> = treinta | cuarenta | cincuenta | sesenta |
   setenta | ochenta | noventa;
<date> = hora | fecha;
<decir> = dime | decir;
```

donde están todas las instrucciones que reconocerá el sistema de órdenes de voz.

3.2.3. Preparación de datos de entrenamiento

Una base de datos de entrenamiento es un conjunto de muestras de voz capturadas en señales de audio, que el software de "entrenamiento", al que llamamos el entrenador, aprende los parámetros de los modelos de las unidades de sonido (fonemas). Contiene la información necesaria para extraer las estadisticas del discurso en forma de modelo acústico.

Es necesario decirle al entrenador las unidades de sonido (los fonemas) que se desean aprender y por lo menos la secuencia en que se producen en cada señal de voz en la base de datos de entrenamiento. Esto se proporciona al entrenador a través de un archivo llamado el archivo de transcripción, en el que la secuencia de palabras y los sonidos no vocales se escriben exactamente como se producen en una señal de voz, seguido de una etiqueta que se puede utilizar para asociar esta secuencia con la señal de voz correspondiente.

El entrenador entonces ve en un diccionario que mapea cada palabra a una secuencia de unidades de sonido, para derivar la secuencia de unidades de sonido asociados con cada señal.

Archivos de configuración

Se crea los siguientes archivos de configuración para entrenar el modelo acústico, utilizado en el sistema de órdenes de voz "gpyvozcontrol":

– Diccionario fonetico (gpyvozcontrol.dic): tiene todas las palabras que están en el archivo de transcripción, una por linea con su pronunciación fonetica, el formato es:

```
palabra_1        fonemas_palabra_1
palabra_2        fonemas_palabra_2
palabra_3        fonemas_palabra_3
palabra_4        fonemas_palabra_4
palabra_5        fonemas_palabra_5
...      ...
...      ...
```

para el sistema de órdenes de voz es (solo se muestran algunas palabras):

```
a a
aaronita  a a r o n i t a
aarónico  a a r o n i k o
aba  a b a
ababa  a b a b a
...
...
b  b e
baalita  b a a l i t a
baba  b a b a
babada  b a b a d a
babadero  b a b a d e r o
...
...
```

```
ca  k a
caaminí  k a a m i n i
cabadelante  k a b a d e l a n t e
cabal  k a b a l
cabalar  k a b a l a r
...
...
```

– Archivo de fonemas (gpyvozcontrol.phone): tiene todos los fonemas utilizados en el diccionario, uno por linea, más el fonema especial 'sil'para el silencio, el formato es:

```
fonema_1
fonema_2
fonema_3
fonema_4
fonema_5
...
...
SIL
```

que para el sistema "gpyvozcontrol" es:

```
a
b
d
e
f
g
...
...
SIL
```

– Archivo de relleno (gpyvozcontrol.filler): contiene los fonemas no cubiertos por el modelo de lenguaje como risas, aliento, entre otros o simplemente el silencio, a continuación se muestra el archivo creado para “gpyvozcontrol”:

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

– - Archivo de transcripción (gpyvozcontrol_train.transcription y gpyvozcontrol_test.transcription): es un archivo que contiene las transcripciones de entrenamiento y de pruebas para cada archivo de audio. Se pueden ver a continuación los archivos creados para el sistema:

- Archivo gpyvozcontrol_train.transcription

<s> entre los libros infantiles más leídos está el cuento de la caperucita roja </s> (01)
<s> el inspector carlos ha tenido mucho peso en la resolución del caso </s> (02)
<s> a lo lejos se divisaba una luz azulada que se movía en zigzag </s> (03)
<s> siempre se ha dicho que por la boca muere el pez </s> (04)
<s> clavó el gusano en el anzuelo </s> (05)
<s> las autoridades sanitarias advierten que el tabaco perjudica seriamente la salud </s> (06)
’<s> en el techo el yeso estaba agrietado </s> (07)
<s> mi madre me ha dicho que coma todo lo que tengo en el plato </s> (08)
<s> el mago parecía de goma se metió en la caja en un abrir y cerrar de ojos </s> (09)
<s> el padre dio como dote unos zapatos y una colección de sellos muy cara </s> (10)
<s> que uno más uno son tres es una afirmación falsa </s> (11)

<s> la jota es un baile regional </s> (12)

<s> el chico miró al asno con desdén </s> (13)

<s> mi madre me mima mucho </s> (14)

<s> no te voy a dar nada de nada </s> (15)

<s> el coche estaba cubierto por una lona de color verde </s> (16)

<s> el llanto del niño me rompía el corazón </s> (17)

<s> mi carro me lo robaron </s> (18)

<s> uno de los huesos más largos del cuerpo es el fémur </s> (19)

<s> en esa aula doy clases de danza </s> (20)

...

...

- Archivo gpyvozcontrol_test.transcription

<s> abrir reproductor </s> (01)

<s> cierra el reproductor </s> (02)

<s> activa orca </s> (03)

<s> reducir la ventana </s> (04)

<s> bajar volumen al equipo </s> (05)

<s> abrir calculadora </s> (06)

<s> minimizar las ventanas </s> (07)

<s> presentar proyecto </s> (08)

<s> cambiar a la lámina ochenta y cuatro </s> (09)

<s> vozcontrol </s> (10)

<s> maximizar las ventanas </s> (11)

<s> maximiza las ventanas </s> (12)

<s> apagar sistema </s> (13)

<s> subir volumen al equipo </s> (14)

<s> cambiar a la lámina once </s> (15)

<s> abrir mensajería </s> (16)

<s> cerrar el reproductor </s> (17)

<s> reduce todas la ventanas </s> (18)
<s> cambiar a la lámina cincuenta y seis </s> (19)
<s> primera canción </s> (20)
...
...

Es importante resaltar que cada frase comienza con <s> y termina con </s> seguido con el identificador del audio entre paréntesis (solo el nombre del audio, no hace falta el camino o la dirección de la carpeta). Es fundamental que coincidan los identificadores de los archivos, Fileids, con los archivos de transcripción (estos dos archivos deben contener la misma cantidad de lineas).

– Archivo Fileids (gpyvozcontrol_train.fileids y gpyvozcontrol_test.fileids): es un archivo que contiene los nombre de los audios de entrenamiento y pruebas, uno por línea, que para "gpyvozcontrol es:

  - gpyvozcontrol_train.fileids

  ramirezm/01
  ramirez/02
  ramirezm/03
  ...
  ...
  rincony/01
  rincony/02
  rincony/03
  ...
  ...
  quinteron/01
  quinteron/02
  quinteron/03
  ...

...

- gpyvozcontrol_test.fileids

prueba1/01
prueba1/02
prueba1/03
...
...
prueba2/01
prueba2/02
prueba2/03
...
...
prueba3/01
prueba3/02
prueba3/03
...
...

Archivos de audio

Revisar la sección 3.3 de este capítulo, además estos archivos están en la carpeta "wav".

Configuración para la creación del modelo acústico

A continuación se describe el procedimiento realizado para la creación del modelo acústico.

– Se crea la carpeta "gpyvozcontrol", se entra a la misma y se ejecuta:

```
sphinxtrain -task gpyvozcontrol setup
```

esto genera la sub-carpeta “etc” con los archivos de configuración necesarios para el entrenamiento, además se copian los archivos de configuración del paso anterior (gpyvozcontrol_train.transcription, gpyvozcontrol_test.transcription, gpyvozcontrol_train.fileids, gpyvozcontrol_test.fileids, gpyvozcontrol.dic, gpyvozcontrol.phone y gpyvozcontrol.filler) dentro de la sub-carpeta.

Se modifica el archivo “etc/sphinx_train.sfg”, según corresponda y quede de la siguiente forma:

```
\$CFG\_WAVFILES\_DIR = "\$CFG\_BASE\_DIR/wav";
\$CFG\_WAVFILE\_EXTENSION = 'wav';
\$CFG\_WAVFILE\_TYPE = 'mswav'; # one of nist, mswav, raw
\$CFG\_HMM\_TYPE = '.cont.'; # Sphinx 4, PocketSphinx
\$CFG\_HMM\_TYPE = '.semi.'; # PocketSphinx
\$CFG\_N\_TIED\_STATES = 200;
```

3.2.4. Entrenamiento del modelo acústico

Ya con los cambios mencionados, se procede al entrenamiento ejecutando:

```
sphinxtrain run
```

esto pasará por todos los procesos para entrenar el modelo, quedando la siguiente estructura:

```
bwaccumdir/
etc/
feat/
logdir/
model\_architecture/
model\_parameters/
```

```
qmanager/
result/
trees/
wav/
gpyvozcontrol.html
```

El archivo "gpyvozcontrol.html" tiene registrada toda la información de los procesos realizados en el entrenamiento del modelo acústico.

El modelo acústico creado esta ubicado en la carpeta generada "model_parameters/gpyvozcontrol.cd_semi_200/" y contiene los siguientes archivos:

- feat.params.
- mdef.
- means.
- mixture_weights.
- noisedict.
- sendump.
- transition_matrices.
- variances.

3.2.5. Ejecución de PocketSphinx

Para probar el modelo acústico creado, se ejecuta:

```
pocketsphinx_continuous -inmic yes -hmm model_parameters/
   gpyvozcontrol.cd_semi_200/ -lm etc/gpyvozcontrol.lm.DMP -dict
    etc/gpyvozcontrol.dic
```

También se puede probar el modelo acústico con una gramática, por lo que se ejecuta:

```
pocketsphinx_continuous -inmic yes -hmm model_parameters/
   gpyvozcontrol.cd_semi_200/ -dict etc/ordenes.dic -jsgf
   ordenes.gram
```

Para que pocketsphinx reconociera la entrada de voz, se tuvo que configurar el volumen del microfóno a un 25 % - 30 %. Volúmenes superiores bloquearon completamente el reconocimiento. Probablemente un problema de configuración del software de base.

Hasta este punto se ha logrado construir la gramática y el modelo acústico necesario para el desarrollo del Sistema de órdenes de voz "gpyvozcontrol".

3.3. Grabaciones

Se grabaron 34 personas (14 mujeres y 20 hombres) para un total de 5400 archivos de audio para el entrenamiento de los modelos acústicos de Julius[1] y CMUSphinx[3], con las siguientes características:

- Formato MSWav
- Frecuencia de muestra 16 Khz
- A 16-bits.
- Stereo aplicación de escritorio.

de las cuales 2380 son grabaciones femeninas y 3020 son grabaciones masculinas. También se resalta el hecho que la mayoría de los acentos son de personas residentes en Mérida.

Es importante resaltar que 15 personas fueron grabadas en un primer momento con 200 frases, luego reajustando las frases para reforzar algunos fonemas de la gramática. Las otras 19 solo grabaron 125 entre una combinación de las ya existentes como otras nuevas. Por eso la cantidad de 282 frases en total.

Otra cosa que se debe mencionar, es la eliminación del ruido de fondo de un gran número de grabaciones, ya que deterioraban la calidad de reconocimiento en los modelos.

Capítulo 4

Sistema de órdenes de voz

Ya con los modelos acústicos creados que son el motor principal para el reconocimiento de las órdenes, se crea el Sistema de órdenes de voz en software libre "gpyvozcontrol" para cada uno de los modelos. Para esto se toma como base el ejemplo realizado por RainCT "Writing a command and control application with voice recognition"[31], el cual consiste en manipular el reproductor de música con la voz del usuario utilizando la biblioteca Julius.

El nombre "gpyvozcontrol" surge de la combinación de las letras "g" por el escritorio Gnome, "py" de Python, ya que esta implementado en este lenguaje y "vozcontrol" porque es con la voz del usuario que se controla el equipo.

Es importante resaltar el hecho que se ha desarrollado un sistema completo de órdenes de voz para cada uno de los modelos acústicos creados en las secciones 3.1 y 3.2, que funcionan de manera separada con cada biblioteca.

4.1. Lista de órdenes

Las órdenes que "gpyvozcontrol" puede reconocer son las siguientes:

- Abrir los programas: Calculadora, Cliente de correo, Navegador web, Mensajería, Terminal, Editor de documentos, Editor de texto, Carpeta de archivos, Reproductor de música, Visualizador de PDF Evince. Mediante la instrucción:

```
abrir o abre programa
```

– Bajar y subir el volumen del sistema. Mediante la instrucción:

```
bajar o subir volumen al sistema o equipo
```

– Saber la hora y la fecha. Mediante la orden:

```
decir o dime la hora
decir o dime la fecha
```

– Ejecutar las acciones de minimizar, maximizar, ampliar, reducir y cerrar las ventanas. Mediante las instrucciones:

```
minimizar o minimiza la ventana
miminizar o minimiza todas las ventanas
maximizar o maximiza la ventana
maximizar o maximiza todas las ventanas
ampliar o amplia la ventana
reducir o reduce la ventana
cerrar o cierra la ventana
tambien se permite solo decir: minimizar,
   maximizar, reducir, ampliar y cerrar.
```

– Saber título de la o las ventanas activas, con el comando de voz:

```
t|  |tulo de la ventana
decir o dime ventanas activas
```

– Manipular el Evince (visualizador de PDF), mediante la orden:

```
lámina siguiente o anterior
```

```
primera o última lamina
cambiar o cambia a la lámina n mero
lámina actual
```

– Manipular el reproductor de música, con las ordenes siguientes:

```
abrir o abre reproductor
canción siguiente o anterior
cambiar o pr xima canción
bajar o subir volumen al reproductor
pausar o pausa el reproductor
título de la canción
```

4.2. Biblioteca libwnck

Para poder manipular las ventanas del entorno gráfico (minimizar, maximizar, ampliar, reducir y cerrar), se utilizó la biblioteca libwnck[32], del ambiente gráfico GNOME, que permite administrar ventanas y espacios de trabajo a través de busqueda, lista de tareas, entre otras actividades.

A continuación se colocan las funciones que realizan ese trabajo:

```
def Acciones(self, screen):
        while gtk.events_pending():
                gtk.main_iteration()
                ventana = wnck.Screen.force_update(screen)
                windows = screen.get_windows()
                for w in windows:
                        return windows
```

que retorna las ventanas activas en el sistema y que son recibidas por la función siguiente:

```
def Actions(self, windows, accion, n):
        for w in windows:
                window = wnck.Screen.get_active_window(w.
                    get_screen())

        if accion == 'maximizar' or accion == 'maximiza':
                if n == 2:
                        win =  wnck.Screen.get_windows(wnck.
                            screen_get_default())
                        for w in win:
                                w.unminimize(0)
                        #window.unminimize(0)
                        print "Maximizando la ventana "+window
                            .get_name()
                        Reproducir("Se maximiza la ventana "+
                            window.get_name())
                if n == 1:
                        print "Maximizando todas las ventanas"
                        win =  wnck.Screen.get_windows(wnck.
                            screen_get_default())
                        for w in win:
                                w.unminimize(0)
                        Reproducir("Se maximizan todas las
                            ventanas")
        if accion == 'minimizar' or accion == 'minimiza':
                if n == 2:
                        window.minimize()
                        print "Minimizando la ventana "+window
                            .get_name()
```

```
                    Reproducir("Se minimiza la ventana "+
                        window.get_name())
            if n == 1:
                    print "Minimizando todas las ventanas"
                    for w in windows:
                            w.minimize()
                    Reproducir("Se minimizan todas las
                        ventanas")
    if accion == 'cerrar' or accion == 'cierra':
            if n == 2:
                    window.close(int(time.time()))
                    print 'Cerrando la ventana '+window.
                        get_name()
                    Reproducir('Cerrando la ventana '+
                        window.get_name())
            if n == 1:
                    win =  wnck.Screen.get_windows(wnck.
                        screen_get_default())
                    for w in win:
                            w.close(int(time.time()))
                    print "Se cierran todas las ventanas"
                    Reproducir("Se cierran todas las
                        ventanas")
    if accion == 'ampliar' or accion == 'amplia':
            if n == 2:
                    window.maximize()
                    print "Ampliando la ventana "+window.
                        get_name()
                    Reproducir("Se ampl a la ventana "+
                        window.get_name())
            if n == 1:
```

```
                print "Ampliando todas las ventanas"
                for w in windows:
                        w.maximize(0)
                Reproducir('Se ampl an todas las
                   ventanas')
        if accion == 'reducir' or accion == 'reduce':
                if n == 2:
                window.unmaximize()
                print "Se reduce la ventana "+window.
                   get_name()
                Reproducir("Se reduce la ventana "+
                   window.get_name())
                if n == 1:
                print "Reduce todas las ventanas"
                for w in windows:
                        w.unmaximize(0)
                Reproducir('Se Reducen todas las
                   ventanas')
        if accion == 'titulo':
                if n == 1:
                win =  wnck.Screen.get_windows(wnck.
                   screen_get_default())
                print "La ventana activa es "+window.
                   get_name()
                Reproducir("La ventana activa es "+
                   window.get_name())
                else:
                Reproducir("Las ventanas activas son:"
                   )
                for w in windows:
                        Reproducir(" "+w.get_name())
```

la cual ejecuta la instrucción.

4.3. Funcionamiento

"Gpyvozcontrol", en cualquiera de sus versiones, funciona de la siguiente forma: El programa captura una señal de audio por medio de un micrófono. Esta es convertida mediante la biblioteca Julius o CMUSphinx en una señal de texto. Luego este texto se envía a la clase principal que ejecuta la orden invocando al sistema operativo o al entorno gráfico.

Antes de comenzar a pronunciar cualquier instrucción, es obligatorio decir la palabra clave "vozcontrol". Esto activa un pequeño sonido (bandera), lo que indica que se puede pronunciar la instrucción a ejecutar. Luego que se ha pronunciado la orden el sistema reproducirá el siguiente mensaje: "espere un momento". Cuando se haya ejecutado la instrucción el sistema reproducirá un mensaje (el mensaje depende de la acción realizada). Cuando no reconoce la orden, el sistema emite el mensaje de audio: "Ups! no entiendo la orden" y se queda en espera de la palabra clave antes para ejecutar la próxima instrucción. Por ejemplo:

usuario: "vozcontrol"

sistema: (un sonido indicando que se puede pronunciar la instrucción)

usuario: abrir editor de texto

sistema: mensaje "espere un momento"

sistema: (ejecuta la orden)

sistema: mensaje "la aplicación editor de texto fue abierta"

sistema: ...(esperando la palabra clave "vozcontrol")

Noten que el sistema reporta, en audio, el resultado de la acción solicitada.

4.4. Archivos y funcionamiento del sistema gpyVozControl

Los archivos que comprende este desarrollo son:

- gpyvozcontrol.py: es el archivo principal con el que se comunica la biblioteca Julius y CMUSphinx para ejecutar el comando del usuario
- reproductor.py: es la clase que permite manipular el reproductor.
- evince.py: es el que permite manipular el visualizador de PDF.
- sistema.py: permite manipular algunas acciones del sistema.
- speak.py: permite reproducir los audios para indicar al usuario que eventos se realizan.
- aplicaciones.py: permite saber que programa desea abrir el usuario.
- correo.py: es el que permite abrir el cliente de correo.
- multimedia.py: permite saber que reproductor esta instalado en el sistema.
- clientecorreo.py: permite saber cual cliente de correo está instalado.
- numero.py: convierte los números de formato letra a dígito.

En gpyvozcontrol.py se implementa la clase CommandAndControl que contiene los métodos de nivel superior que controlan todo el sistema e invocan a los objetos que encapsulan las funcionalidades particulares y las invocaciones al sistema operativo y a su gestor de interfaz gráfica. Para ilustrar las dependencias funcionales internas se muestra a continuación el diagrama de clases del sistema.

4.5. Diagrama de clases

En la imagen 4.1 se muestra el diagrama de clases para “gpyvozcontrol”:

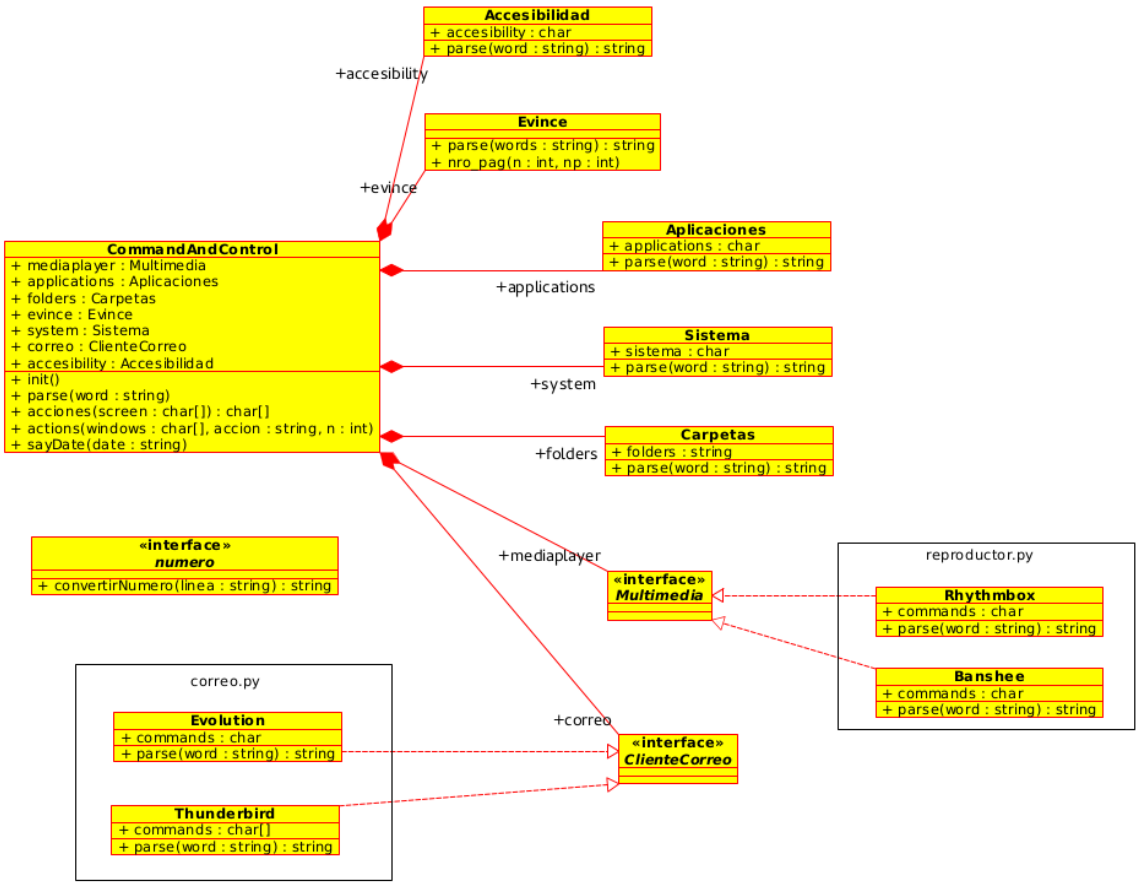

Figura 4.1: Diagrama de clases de gpyvozcontrol

Las asociaciones entre clases que se muestran en el diagrama corresponden a caminos de invocación de acciones que se realizan bien en el sistema operativo (o en su gestor gráfico) o bien en otras aplicaciones. Noten que aplicaciones genéricas como el reproductor de audio y el lector de correo pueden tener realizaciones alternativas, pero cuentan con un interfaz común desde el sistema.

4.6. Repositorio del proyecto

Con la instrucción del protocolo git para gestión de proyectos, se puede descargar el proyecto desde el repositorio BitBucket, así:

```
git clone https://usuarioousuaria@bitbucket.org/gproyectogrado
    /gpyvozcontrol.git
```

en donde la dirección `https://usuarioousuaria@bitbucket.org/gproyectogrado/gpyvozcontrol.git` incluye los detalles del proyecto, pero la etiqueta usuarioosuaria debe corresponder al identificador de un usuario o usuaria real del repositorio BitBucket.

Allí se encuentran los fuentes y archivos creados con las bibliotecas Julius[1] y CMUSphinx[3], incluyendo los script, modelos acústicos y el sistema de órdenes de voz creados por el autor de este proyecto.

Capítulo 5

Resultados

5.1. Campaña de Pruebas

Para verificar el nivel de logro alcanzado con la implementación del sistema GpyVozControl y comparar sistemáticamente las dos bibliotecas de soporte que utiliza, se realizaron 3 conjuntos de pruebas a los modelos acústicos creados con las bibliotecas Julius[1] y CMUSphinx[3], cada una con 100 audios grabados, de las cuales 2 pruebas se grabaron con un micrófono externo y 1 prueba con el micrófono integrado en el equipo. Las pruebas fueron realizadas por dos voces masculinas y una voz femenina. A continuación se muestra un fragmento de las frases grabadas para las pruebas (el total de los archivos de prueba se encuentra en el repositorio del proyecto):

*/01 abrir reproductor

*/02 cierra el reproductor

*/03 activa orca

*/04 reducir la ventanas

*/05 bajar volumen al equipo

*/06 abrir calculadora

*/07 minimizar las ventanas

*/08 presentar proyecto

*/09 cambiar a la lamina ochenta y cuatro

*/10 vozcontrol

*/11 maximizar las ventanas
*/12 maximiza las ventanas
*/13 apagar sistema
*/14 subir volumen al equipo
*/15 cambiar a la lamina once
*/16 abrir mensajeria
*/17 cerrar el reproductor
*/18 reduce todas la ventanas
*/19 cambiar a la lamina cincuenta y seis
*/20 primera cancion
...
...

En las siguientes secciones se describe el proceso llevado a cabo para cada biblioteca.

Pruebas y resultados en Julius

– Ejecución de pruebas

Para realizar pruebas al modelo acústico creado con la biblioteca Julius[1], se siguió el libro de HTK "HTKBOOK"[23], sección 3.4.

HTK[6] necesita de una red de palabras, la cual se crea a partir de la gramática utilizada, representada como se muestra a continuación (este archivo se debe crear. Para el sistema de órdenes de voz se llama "gramatica"):

$ACCION = abrir | abre;
$ACCION_V = cerrar | cierra | maximiza[r] | minimiza[r] | amplía[r] | reducir | reduce;
$EDICION = presentaci|ó|n | texto | documento | hoja;
$PROGRAMA_O = calculadora | navegador | mensajería | terminal | consola | correo;
$SISTEMA = equipo | sistema;
$ACCION_S = apagar | reiniciar;
$ARCHIVO = personal | videos | música | descargas | documentos;
$LA = la | las;
$ACCION_R = siguiente | anterior | detener | reproducir;

$ACCION_L = primera | última | cambiar | próxima;
$ACCION_RP = bajar | subir | silenciar;
$ACCION_RP_1 = cerrar | cierra | pausa[r];
$LAMINA = lámina | canción;
$TESIS = tesis | proyecto;
$ACTIVO = canción | ventana;
$DECIR = decir | dime;
$DIGIT5 = uno | dos | tres | cuatro | cinco;
$DIGIT9 = seis | siete | ocho | nueve;
$DIGIT10 = diez | once | doce | trece | catorce | quince;
$DIGIT100 = veinte | treinta | cuarenta | cincuenta | sesenta | setenta | ochenta | noventa;
$ACCION_A = activa[r] | desactiva[r];
$ACCESIBILIDAD = orca | teclado | magnificador;
$DATE = hora | fecha;
(SENT-START (vozcontrol | $ACCION (editor de $EDICION | $PROGRAMA_O | carpeta $ARCHIVO | reproductor) | $ACCION_V [todas] $LA ventanas] | $ACCION_R $LAMINA | $ACCION_RP volumen al (reproductor | $SISTEMA) | título de la $ACTIVO | $DECIR la $DATE | $ACCION_L $LAMINA | $ACCION_RP_1 el reproductor | $ACCION_S $SISTEMA | presentar $TESIS | $LAMINA $ACCION_R | cambiar a la lámina ($DIGIT5 | $DIGIT9 | $DIGIT10 | $DIGIT100 [y ($DIGIT5 | $DIGIT9)]) | $ACCION_A $ACCESIBILIDAD) SENT-END)

donde '|'(barras verticales) son elementos alternativos y '[ ]'(corchetes) son elementos opcionales. Con el archivo "gramatica" y el comando "HParse" se crea la red de palabras:

```
HParse gramatica RedPalabras
```

lo que genera el archivo "RedPalabras".

Teniendo que "tes.scp" contiene la lista de los archivos de pruebas codificadas (es parecido al archivo "train.scp" creado en el paso 6 descrito en un capítulo anterior),

entonces cada archivo de prueba es reconocido ejecutando el siguiente comando:

```
HVite -T 1 -C ../../../input_files/config -H ../../hmm15/macros
    -H ../../hmm15/hmmdefs -S tes.scp \
-l '*' -i reconocimiento -w ../RedPalabras \
-p 0.0 -s 5.0 ../../../lexicon/VoxForgeDict.txt ../../tiedlist
```

lo que genera el archivo "reconocimiento", los archivos "tiedlist", "hmm15/macros" y "hmm15/hmmdefs" que fueron creados con el modelo acústico.

Ahora para obtener los resultados reales de prueba, se ejecuta:

```
HResults -I tes.mlf tiedlist reconocimiento
```

lo que da el rendimiento real del modelo acústico, donde el archivo "tes.mlf" contiene transcripciones de nivel de palabra de cada archivo de prueba (descrito en el paso 4).

Los comandos anteriormente mencionados se ejecutan mediante el siguiente script:

```
#! /usr/bin/env python
# coding: utf-8

import os

os.system("HVite -T 1 -C ../../../input_files/config -H ../../
   hmm15/macros -H ../../hmm15/hmmdefs -S tes.scp \
-l '*' -i reconocimiento -w ../RedPalabras \
-p 0.0 -s 5.0 ../../../lexicon/VoxForgeDict.txt ../../tiedlist
   ")

os.system('HResults -I tes.mlf ../tiedlist reconocimiento')
```

que fue implementado en python y se llama "tesdepruebas.py".

Y las corridas de los 3 conjuntos de prueba se realiza mediante el script "corridaTes.sh" y que contiene:

```
#! /bin/bash
# coding: utf-8

echo "Script para ejecutar todas las pruebas"
echo ""
echo "Corriendo la prueba 1"
cd prueba1/
python ../tesdepruebas.py > ../resultado1
cd ..

echo "Corriendo la prueba 2"
cd prueba2/
python ../tesdepruebas.py > ../resultado2
cd ..

echo "Corriendo la prueba 3"
cd prueba3/
python ../tesdepruebas.py > ../resultado3
cd ..
```

– Resultados

Estos son los resultados obtenidos para cada una de las pruebas realizadas en el modelo acústico de Julius[1]:

1. Prueba 1: Masculino (micrófono externo)

```
====================== HTK Results Analysis =======================
  Date: Sun Dec 20 14:47:06 2015
  Ref : tes.mlf
  Rec : reconocimiento
```

```
---------------------- Overall Results -------------------------
SENT: %Correct=52.00 [H=52, S=48, N=100]
WORD: %Corr=74.84, Acc=74.84 [H=235, D=9, S=70, I=0, N=314]
================================================================
```

dando un 52.0 % de frases reconocidas de un total de 100 y un 74.84 % de palabras reconocidas de un total de 314.

Donde: "H" son las palabras reconocidas, "D" los errores de eliminación, "S" los errores de sustitución, "I" los errores de inserción y "N" el número total de palabras. Esos errores se refieren siempre a acciones del sistema que no corresponden con lo que debe hacer. Por ejemplo, reemplazar una palabra por otra diferente es un error de sustitución, eliminar la palabra es uno de eliminación y agregar una palabra sin necesidad es un error de inserción.

2. Prueba 2: Femenino (micrófono externo)

```
====================== HTK Results Analysis =======================
  Date: Sun Dec 20 14:47:12 2015
  Ref : tes.mlf
  Rec : reconocimiento
---------------------- Overall Results -------------------------
SENT: %Correct=44.00 [H=44, S=56, N=100]
WORD: %Corr=59.24, Acc=59.24 [H=186, D=39, S=89, I=0, N=314]
================================================================
```

dando un 44.0 % de frases reconocidas de un total de 100 y un 59.24 % de palabras reconocidas de un total de 314.

3. Prueba 3: Masculino (micrófono interno)

```
====================== HTK Results Analysis =======================
  Date: Sun Dec 20 14:47:18 2015
  Ref : tes.mlf
```

```
  Rec : reconocimiento
---------------------- Overall Results -------------------------
SENT: %Correct=61.00 [H=61, S=39, N=100]
WORD: %Corr=81.21, Acc=80.25 [H=255, D=6, S=53, I=3, N=314]
================================================================
```

dando un 61 % de frases reconocidas de un total de 100 y un 81,21 % de palabras reconocidas de un total de 314.

Pruebas y resultados en CMUSphinx

– Ejecución de pruebas

Para realizar las pruebas al modelo acústico creado con la biblioteca CMUSphinx[3], es necesario tener un modelo de lenguaje, para esto se sigue el tutorial "Reconocimiento de voz"[33]. A continuación se detalla como se crea el modelo de lenguaje.

Estos son los programas necesarios para crear el modelo de lenguaje:

1. Kit de modelo de lenguaje: `http://svr-www.eng.cam.ac.uk/~prc14/CMU-Cam_Toolkit_v2.tar.gz`
2. Convertidor de modelo de lenguaje: `http://www.speech.cs.cmu.edu/sphinx/download/nightly/lm3g2dmp.nightly.tar.gz`

Para compilar dichos programas, se deben seguir los pasos en el archivo "README" de cada carpeta.

Ahora bien, para la creación del modelo de lenguaje, se crea un archivo de texto formateando las transcripciones del archivo "etc/gpyvozcontrol_train.transcription" eliminando los identificadores. El siguiente script "corpus.py", creado para este proyecto, realiza esta acción:

```
#! /usr/bin/env python
# coding: utf-8

import re
```

```
f = open('etc/gpyvozcontrol_train.transcription', 'r')
c = open('lm/train.corpus', 'w')
t = ""

for a in f:
        t = re.sub('[]\(\d\)]', '', a)
        c.write(t)

f.close()
c.close()
```

Con esto se crea el archivo de texto "train.corpus" en la carpeta "lm". Luego se crea el archivo "con.ccs" con lo siguiente:

```
<s>
```

y se crea el modelo, ejecutando el código:

```
cat train.corpus | ../../programas/CMU-Cam_Toolkit_v2/bin/
   text2wfreq | sort -rn -k 2 > train.wfreq

cat train.wfreq | ../../programas/CMU-Cam_Toolkit_v2/bin/
   wfreq2vocab -gt 0 > train.vocab

cat train.corpus | ../../programas/CMU-Cam_Toolkit_v2/bin/
   text2wngram -n 3 -temp /tmp >  train.w3gram

cat train.w3gram | ../../programas/CMU-Cam_Toolkit_v2/bin/
   wngram2idngram -n 3 -vocab train.vocab -temp /tmp > train.
   id3gram
```

```
../../programas/CMU-Cam_Toolkit_v2/bin/idngram2lm -idngram
   train.id3gram -vocab train.vocab -context con.ccs -
   witten_bell -n 3 -vocab_type 1 -arpa train.lm
```

Por último, se transforma el modelo a formato "DMP":

```
../../programas/lm3g2dmp/lm3g2dmp train.lm ../etc/
```

esto genera el archivo "train.lm.DMP", que se renombra a "gpyvozcontrol.lm.DMP" y se coloca en la carpeta "etc/".

Ya teniendo los archivos "etc/gpyvozcontrol_test.transcription" y "etc/gpyvozcontrol_test.fileids", creados en la sección 3.2.2, solo queda ejecutar:

```
sphinxtrain -s decode run
```

realizando esta operación para cada una de las pruebas.

Resultados

Estos son los resultados obtenidos para cada una de las pruebas realizadas en el modelo acústico de CMUSphinx[3]:

1. Prueba 1: Masculino (micrófono externo)

   En el siguiente fragmento se puede observar lo que reconoce el modelo:

```
ABRIR   reproductor  (prueba1-01)
ALGUIEN reproductor  (prueba1-01)
Words: 2 Correct: 1 Errors: 1 Percent correct = 50.00% Error = 50.00% Accuracy = 50.00%
Insertions: 0 Deletions: 0 Substitutions: 1
cierra el  reproductor  (prueba1-02)
cierra el  reproductor  (prueba1-02)
Words: 3 Correct: 3 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
ACTIVA  orca  (prueba1-03)
ACTIVAR orca  (prueba1-03)
Words: 2 Correct: 1 Errors: 1 Percent correct = 50.00% Error = 50.00% Accuracy = 50.00%
Insertions: 0 Deletions: 0 Substitutions: 1
reducir LA  VENTANA   (prueba1-04)
reducir LAS VENTANAS  (prueba1-04)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 0 Substitutions: 2
***   BAJAR VOLUMEN AL  equipo  (prueba1-05)
VAMOS UN    EN      EL  equipo  (prueba1-05)
Words: 4 Correct: 1 Errors: 4 Percent correct = 25.00% Error = 100.00% Accuracy = 0.00%
Insertions: 1 Deletions: 0 Substitutions: 3
```

```
abrir calculadora  (prueba1-06)
abrir calculadora  (prueba1-06)
Words: 2 Correct: 2 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
```

lo que da como resultado:

```
TOTAL Words: 314 Correct: 168 Errors: 167
TOTAL Percent correct = 53.50% Error = 53.18% Accuracy = 46.82%
TOTAL Insertions: 21 Deletions: 17 Substitutions: 129
```

2. Prueba 2: Femenino (micrófono externo)

   Se puede observar un fracmento de lo que reconoce el modelo:

```
ABRIR   REPRODUCTOR  (prueba2-01)
AMPLIAR TODO         (prueba2-01)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 2
cierra el  reproductor  (prueba2-02)
cierra el  reproductor  (prueba2-02)
Words: 3 Correct: 3 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
ACTIVA ORCA   (prueba2-03)
AQUÍ   AHORA  (prueba2-03)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 2
reducir la  ventana  (prueba2-04)
reducir la  ventana  (prueba2-04)
Words: 3 Correct: 3 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
*** BAJAR VOLUMEN AL  equipo  (prueba2-05)
VA  ÁRBOL NOVENTA EL  equipo  (prueba2-05)
Words: 4 Correct: 1 Errors: 4 Percent correct = 25.00% Error = 100.00% Accuracy = 0.00%
Insertions: 1 Deletions: 0 Substitutions: 3
abrir calculadora  (prueba2-06)
abrir calculadora  (prueba2-06)
Words: 2 Correct: 2 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
```

   lo que da como resultado:

```
TOTAL Words: 314 Correct: 135 Errors: 205
TOTAL Percent correct = 42.99% Error = 65.29% Accuracy = 34.71%
TOTAL Insertions: 26 Deletions: 36 Substitutions: 143
```

3. Prueba 3: Masculino (micrófono interno)

   Se puede observar un fragmento de lo que reconoce el modelo:

```
abrir *** reproductor  (prueba3-01)
abrir EL  reproductor  (prueba3-01)
Words: 2 Correct: 2 Errors: 1 Percent correct = 100.00% Error = 50.00% Accuracy = 50.00%
Insertions: 1 Deletions: 0 Substitutions: 0
cierra el  reproductor  (prueba3-02)
cierra el  reproductor  (prueba3-02)
Words: 3 Correct: 3 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
ACTIVA    ORCA     (prueba3-03)
ACTIVARON PORTERO  (prueba3-03)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 2
reducir LA  VENTANA  (prueba3-04)
reducir LO  ROBARON  (prueba3-04)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 0 Substitutions: 2
*** BAJAR VOLUMEN AL  equipo  (prueba3-05)
VA  ÁRBOL EL      PAN equipo  (prueba3-05)
Words: 4 Correct: 1 Errors: 4 Percent correct = 25.00% Error = 100.00% Accuracy = 0.00%
Insertions: 1 Deletions: 0 Substitutions: 3
abrir calculadora  (prueba3-06)
abrir calculadora  (prueba3-06)
Words: 2 Correct: 2 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
```

lo que da como resultado:

```
TOTAL Words: 314 Correct: 181 Errors: 175
TOTAL Percent correct = 57.64% Error = 55.73% Accuracy = 44.27%
TOTAL Insertions: 42 Deletions: 16 Substitutions: 117
```

Las comparaciones a nivel de palabras, que ambas plataformas realizan a su manera, muestran que Julius-HTK alcanza un mejor rendimiento que CMUSphinx en ese contexto restringido.

5.2. Análisis de Resultados

De los resultados obtenidos para los modelos acústicos de Julius[1] y CMUSphinx[3], se pueden hacer dos análisis, uno con respecto al mismo modelo y el porqué de la diferencia en las pruebas y el otro, con respecto a la comparación entre ellos.

5.2.1. Diferencia de las pruebas para el mismo modelo

Si se observan las pruebas realizadas a los modelos de Julius[1] y CMUSphinx[3], se puede notar algunas diferencias entre cada una de ellas. Se toman como ejemplo las pruebas realizadas al modelo acústico creado en Julius[1] para hacer el análisis y que aplica para los dos casos. En la tabla 5.2.1 se pueden ver los resultados:

Prueba	% frases reconocidas	% palabras reconocidas
1	52.0	74.84
2	44.0	59.24
3	61.0	81,21

Las diferencias que se pueden notar en la tabla 5.2.1 y que también se pueden aplicar para el modelo acústico de CMUSphinx podrían deberse a:

- Las grabaciones de las pruebas 1 y 2 fueron realizadas con una lectura rápida.
- La prueba 2 se grabó a una distancia mayor a la ubicación del micrófono, lo que da un mayor error de reconocimiento con respecto a las pruebas 1 y 3.
- La prueba 3, aunque se hizo con el micrófono interno se grabó con una pronunciación lenta, por eso su mayor exactitud con respecto a las pruebas 1 y 2.
- La precisión en la pronunciación correcta de las frases.
- La falta de un número mayor de grabaciones.

5.2.2. Diferencia de las pruebas entre modelos

Se puede notar una gran diferencia de reconocimiento entre modelos. En la tabla 5.2.2 se puede observar:

Prueba	% palabra reconocida (Julius)	% palabra reconocidas (CMUSphinx)
1	74.84	53.50
2	59.24	42.99
3	81,21	57.64

la tabla 5.2.2 muestra que el modelo acústico creado con la biblioteca Julius reconoce mucho mejor que el modelo creado con la biblioteca CMUSphinx. Las razones de estas diferencias podrían ser:

- Para las pruebas del modelo creado con Julius, se utiliza una gramática, lo que ajusta las grabaciones a lo que se está reconociendo.
- Las pruebas del modelo CMUSphinx utiliza el modelo de lenguaje de todas las transcripciones utilizadas en la creación del modelo, no teniendo una gramática a seguir, lo que puede ocasionar mas % de error.

Con modelos acústicos creados solo con los audios de las 15 personas (grabaron 200 frases cada una) y solo 19 personas (125 frases cada una), cada conjunto por separado, se obtuvo mejores resultados que con todo el conjunto de audios. La diferencia está entre un 2 % y 6 % aproximadamente. Es decir, el modelo creado con todas las grabaciones es menos eficiente. No obstante los resultados de esta prueba sistemática, en el uso cotidiano el modelo acústico creado con la biblioteca CMUSphinx[3] reconoce mucho mejor que el creado con la biblioteca Julius[1]. Obviamente se requiere ampliar la gama de pruebas para identificar aspectos del reconocimiento que no parecieran reflejarse acá.

Capítulo 6

Conclusión

Ya realizada la investigación y desarrollo del Sistema de órdenes de voz en software libre “Gpyvozcontrol”, se concluye lo siguiente:

Se identificaron y estudiaron dos bibliotecas que permiten realizar reconocimiento de voz en software libre: a saber el motor de voz Julius[1] que es un sistema de reconocimiento preciso en tiempo real, de alta velocidad, basado en la estrategia de doble paso y el motor CMUSphinx[3] de la Universidad Carnegie Mellon, un conjunto de herramientas para el reconocimiento de voz que permite desarrollar juegos, entre otras aplicaciones.

También se identificó e incorporó al projecto la biblioteca WNCK[32], que permite saber cuáles ventanas del sistema están activas, además de poder realizar las acciones de minimizar, maximizar, ampliar, reducir y cerrarlas ventanas del sistema.

Se logró grabar a 34 personas (14 mujeres y 20 hombres). Es decir, se construyó un corpus de audio con un total de 5400 archivos de audio en español a 16000 bits, en canal estereo y 16 Khz para alimentar el modelo acústico tanto de la biblioteca Julius[1] como CMUSphinx[3].

Se crearon los modelos acústicos con las bibliotecas Julius[1] y CMUSphinx[3] que permiten reconocer las instrucciones de la gramática creada y que es utilizado por el Sistema de órdenes de voz. Se destacan cuatro pasos importantes en la creación de dichos modelos como son: la creación de la gramática, los archivos de configuración (archivos de pronunciación y transcripción), llamado modelo de lenguaje, codificación

de los archivos de audio y la creación del modelo.

Entre las instrucciones que realiza “gpyvozcontrol” están: abrir aplicaciones (cliente de correo y mensajería, calculadora, editor de documentos y texto, hoja de calculo, editor de presentaciones, terminal, navegador, carpeta de archivos), también las acciones de minimizar, maximizar, ampliar, reducir y cerrar la ventana activa en la interfaz gráfica. Por último, el sistema permite manipular el reproductor de música (abrir el reproductor, ir a la siguiente o anterior canción) y el visualizador de PDF Evince (cambiar de lámina (siguiente o anterior), ir a una lámina específica).

El sistema de órdenes de voz fue creado con el lenguaje python, utilizando los modelos acústicos e integrados con las gramática creada. El sistema espera la palabra clave “vozcontrol” para luego esperar por la instrucción que desea realizar el usuario, este proceso se repite continuamente. “Gpyvozcontrol’también notifica con mensajes auditivos las acciones que realiza el usuario, lo que ayuda a una persona con discapacidad visual saber que ocurre en el sistema.

Otro punto importante es que las diferencias de los resultados obtenidos en las pruebas realizadas a los modelos acústicos se pueden deber principalmente a la calidad de las grabaciones, a la incorrecta pronunciación de las frases, la no uniformidad de las frases grabadas (se hizo dos grupos de grabaciones), la falta de un número mayor de audios para el entrenamiento. En todo caso, el nivel de reconocimiento con una herramienta prototipo como la que se describe y con un corpus de audio todavía limitado, supera el 50 %. La comprensión del diseño profundo de las bibliotecas de soporte y de los parámetros de cada una seguramente permitirán mejoras en ese nivel de reconocimiento en el corto plazo.

Aunque falta mucho por desarrollar en este sistema, se ha cumplido con los objetivos de este proyecto luego de un arduo trabajo de investigación y desarrollo, logrando construir un sistema que permite a una usuaria o usuario ejecutar con su voz algunas órdenes de voz en el escritorio de Gnome, además de manipular el evince y el reproductor de música, con notificaciones auditivas para ayudar a las personas con discapacidad visual. Esta una de las funcionalidades principales de este desarrollo, ya que corresponde con pautas para el acceso universal de cualquier persona y que siempre se deben tener en cuenta en todo sistema accesible.

Más allá de los objetivos planteados al inicio de la investigación, el desarrollo de "gpyvozcontrol" ha sido logrado con la variedad venezolana del español con sus características que le son propias en prosodia, tono y acento. Ello, sin lugar a dudas, tiene gran importancia no solo para estimular estudios con otras variedades geográficas del español, sino para la construcción de sistemas de síntesis de habla a partir de los monofonos y trifonos que ya posee la herramienta y que, de replicarse en otros países, generarían herramientas con mayor robustez, gratuitas y de gran utilidad para personas con discapacidad.

Queda pendiente desarrollar extensiones al sistema que permitan ejecutar acciones como el guardar un texto, abrir un documento especifico, hacer dictado, busquedas en internet, entre otras muchas. Al ofrecer el sistema como software libre se debe entender que tales extensiones son bienvenidas y serán acompañadas. Con esto, confiamos asegurar el continuo desarrollo del proyecto, ya sea por universidades, instituciones o desarrolladores voluntarios. Se estima que en ese proceso continuo un equipo de desarrolladores pueda enfrentar dificultades mayores como la gestión precisa de entornos gráficos que incluyan manejo de eventos, notificaciones, ventanas emergentes e interrupciones.

Bibliografía

[1] Decodificador julius. página: http://julius.sourceforge.jp/en_index.php. [Internet; Revisado Marzo-2015].

[2] julius/readme. PÍdisponible en: https://github.com/julius-speech/julius/blob/master/README.md. [Internet; Revisado Diciembre-2015].

[3] Cmu sphinx. página: http://cmusphinx.sourceforge.net/. [Internet; Revisado Enero-2015].

[4] Wikipedia. Scrum — wikipedia, la enciclopedia libre, 2015. [Internet; descargado Febrero-2015].

[5] Tutorial: Create acoustic model - manually. página: http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial. [Internet; Revisado Enero-2015].

[6] The hidden markov model toolkit (htk). página: http://htk.eng.cam.ac.uk/. [Internet; Revisado Febrero-2015].

[7] Training acoustic model for cmusphinx. página: http://cmusphinx.sourceforge.net/wiki/tutorialamtesting. [Internet; Revisado Enero-2015].

[8] Cvoicecontrol. Página: http://www.kiecza.net/daniel/linux/cvoicecontrol/index.htmltoc2. [Internet; Revisado Marzo-2015].

[9] Perlbox. Página: http://perlbox.sourceforge.net/. [Internet; Revisado Marzo-2015].

[10] Perlboxvoiceserver. Página: http://perlbox.sourceforge.net/pbtk/api/. [Internet; Revisado Marzo-2015].

[11] Google summer of code. Página: https://www.google-melange.com/gsoc/homepage/google/gsoc2015. [Internet; Revisado Mayo-2015].

[12] Gnome-voice-control. Página: https://wiki.gnome.org/Projects/GnomeVoiceControl. [Internet; Revisado Abril-2015].

[13] Repositorio de gnome-voice-control en ubuntu. página: https://code.launchpad.net/ jazzva/gnome-voice-control/ubuntu. [Internet; Revisado Mayo-2015].

[14] google2ubuntu. Git: https://github.com/benoitfragit/google2ubuntu. [Internet; Revisado Junio-2015].

[15] Google voice. PÍdisponible en: https://developers.google.com/voice-actions/interaction/. [Internet; Revisado Octubre-2015].

[16] Google apis terms of service. PÍdisponible en: https://developers.google.com/terms/. [Internet; Revisado Octubre-2015].

[17] Simon. Página: https://simon.kde.org/. [Internet; Revisado Mayo-2015].

[18] Peter Grasch. Blog: http://grasch.net/ and https://simon.kde.org/aggregator/sources/1. [Internet; Revisado Mayo-2015].

[19] Creación de un corpus para cmu sphinx. página: http://www.speech.cs.cmu.edu/tools/lmtool-new.html. [Internet; Revisado Marzo-2015].

[20] Building language model. página: http://cmusphinx.sourceforge.net/wiki/tutoriallm. [Internet; Revisado Abril-2015].

[21] A. Bonnet, J. Gutierres, and H. Hernández. Reconocedor automático de comandos por medio del habla para las funciones de un automóvil. Proyecto de grado, Instituto Politécnico Metropolitano, Ingeniería en Comunicaciones y Electrónica. [Internet; Descargado Junio-2015].

[22] Reconocimiento de voz en castellano para linux. página: http://ubanov.wordpress.com/2008/11/28/reconocimiento-de-voz-en-castellano/. [Internet; Revisado Septiembre-2015].

[23] Wikipedia. Modelo oculto de márkov — wikipedia, la enciclopedia libre, 2015. [Internet; descargado Marzo-2015].

[24] Manual de julia. página: http://julia-es-la.readthedocs.org/es/latest/manual/introduction. [Internet; Revisado Octubre-2015].

[25] Wikipedia. Mfcc — wikipedia, la enciclopedia libre, 2015. [Internet; descargado Octubre-2015].

[26] Marta Duxans Barrobés, Helenca;Ruiz Costa. Reconocimiento automático del habla. Disponible en: https://www.exabyteinformatica.com/uoc/Audio/Procesamiento_de_audio/Procesamiento_de_a [Internet; Revisado Octubre-2015].

[27] Htk book. página: http://www.ee.columbia.edu/ln/LabROSA/doc/HTKBook21/HTKBook.htm [Internet; Revisado Febrero-2015].

[28] How to create tree.hed - questions. página: http://www.voxforge.org/home/docs/faq/faq/how-to-create-tree.hed-questions. [Internet; Revisado Octubre-2015].

[29] Wiki: Acoustictreequestions. página: http://www.dev.voxforge.org/projects/Main/wiki/Acoustic [Internet; Revisado Octubre-2015].

[30] Learning to use the cmu sphinx automatic speech recognition system. página: http://www.speech.cs.cmu.edu/sphinx/tutorial.html. [Internet; Revisado Agosto-2015].

[31] Writing a command and control application. página: http://bloc.eurion.net/archives/2008/writing-a-command-and-control-application-with-voice-recognition/. [Internet; Revisado Febrero-2015].

[32] Wnck 3.0. Página: https://lazka.github.io/pgi-docs/Wnck-3.0/index.html. [Internet; Revisado Octubre-2015].

[33] Reconocimiento de voz. página: http://golem-iimas.wikidot.com/reconocedor-de-voz. [Internet; Revisado Septiembre-2015].

Anexos

Apéndice A

Gramática en Julius gpyVozControl

A.1. Archivo gpyvozcontrol.grammar

```
###############################################
##
## archivo: gpyvozcontrol.grammar
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
```

```
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR. Revisar
## la Licencia Publica General GNU Affero para mayor información.
##
## Debería haber recibido una copia de la Licencia Pública General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################
S: SN_B SENT SN_E
SENT: NOMBRE
SENT: ACCION EDITOR DE EDICION
SENT: ACCION PROGRAMA_O
SENT: ACCION  CARPETA ARCHIVO
SENT: ACCION  CARPETA DE ARCHIVO
SENT: ACCION_V LA VENTANA
SENT: ACCION_V TODAS LA VENTANA
SENT: ACCION_V
SENT: ACCION_V TODAS
SENT: ACCION REPRODUCTOR
SENT: ACCION_R LAMINA
SENT: ACCION_RP VOLUMEN AL REPRODUCTOR
SENT: ACCION_RP VOLUMEN AL SISTEMA
SENT: TITULO DE LA ACTIVO
SENT: DECIR LA DATE
```

SENT: ACCION_L LAMINA

SENT: ACCION_RP_1 AL REPRODUCTOR

SENT: ACCION_RP VOLUMEN AL SISTEMA

SENT: ACCION_S SISTEMA

SENT: PRESENTAR TESIS

SENT: LAMINA ACCION_R

SENT: DECIR VENTANA ACTIVAS

SENT: CAMBIAR A LAMINA NUMEROS

SENT: ACCION_A ACCESIBILIDAD

NUMEROS: NUMERO

NUMERO: DIGIT0

NUMERO: DIGIT5

NUMERO: DIGIT9

NUMERO: DIGIT20

NUMERO: DIGIT2X DIGIT5

NUMERO: DIGIT2X DIGIT9

NUMERO: DIGIT10

NUMERO: DIECI DIGIT5

NUMERO: DIECI DIGIT9

NUMERO: DIGIT30

NUMERO: DIGIT30 Y DIGIT5

NUMERO: DIGIT30 Y DIGIT9

NUMERO: DIGIT40

NUMERO: DIGIT40 Y DIGIT5

NUMERO: DIGIT40 Y DIGIT9

NUMERO: DIGIT50

NUMERO: DIGIT50 Y DIGIT5

NUMERO: DIGIT50 Y DIGIT9

NUMERO: DIGIT60

NUMERO: DIGIT60 Y DIGIT5

NUMERO: DIGIT60 Y DIGIT9

NUMERO: DIGIT70
NUMERO: DIGIT70 Y DIGIT5
NUMERO: DIGIT70 Y DIGIT9
NUMERO: DIGIT80
NUMERO: DIGIT80 Y DIGIT5
NUMERO: DIGIT80 Y DIGIT9
NUMERO: DIGIT90
NUMERO: DIGIT90 Y DIGIT5
NUMERO: DIGIT90 Y DIGIT9

A.2. Archivo gpyvozcontrol.voca

```
##############################################
##
## archivo: gpyvozcontrol.voca
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
```

```
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################
% SN_B
<s>         sil

% SN_E
</s>         sil

% ACCION
abrir a b r i r
abre a b r e

% ACCION_V
cerrar th e rr a r
cierra th i e rr a
maximizar m a k s i m i th a r
maximiza m a k s i th a
minimizar m i n i m i th a r
```

```
minimiza m i n i m i th a
ampliar a m p l i a r
amplia a m p i a
reducir r e d u th i r
reduce r e d u th e
cancelar k a n th e l a r
cancela k a n th e l a

%EDICION
presentacion p r e s e n t a th i o n
texto t e k s t o
documento d o k u m e n t o s
hoja o x a

%PROGRAMA_O
calculadora k a l k u l a d o r a
navegador n a b e g a d o r
mensajeria m e n s a x e r i a
terminal t e r m i n a l
consola k o n s o l a
correo k o rr e o
gmail g m a i l
proyecto p r o ll e k t o
wiki u i k i

%REPRODUCTOR
reproductor r e p r o d u k t o r

%SISTEMA
equipo e k i p o
sistema s i s t e m a
```

```
% ACCION_S
apagar a p a g a r
reiniciar r e i n i th i a r

% ARCHIVO
personal p e r s o n a l
videos b i d e o s
musica m u s i k a
descargas d e s k a r g a s
documentos d o k u m e n t o s

% EDITOR
editor e d i t o r

% DE
de d e

% LA
la l a
las l a s

% AL
al a l

% CARPETA
carpeta k a r p e t a

% VENTANA
ventana b e n t a n a
ventanas b e n t a n a s
```

```
%TODAS
todas t o d a s
todo t o d o

%ACCION_R
siguiente s i g i e n t e
anterior a n t e r i o r
detener d e t e n e r
reproducir r e p r o d u th i r
actual a k t u a l

%ACCION_L
primera p r i m e r a
ultima u l t i m a
cambiar k a m b i a r
proxima p r o k s i m a

%ACCION_RP
bajar b a x a r
subir s u b i r
silenciar s i l e n th i a r

%ACCION_RP_1
cerrar th e rr a r
cierra th i e rr a
pausa p a u s a
pausar p a u s a r

%VOLUMEN
volumen b o l u m e n
```

```
%PRESENTAR
presentar p r e s e n t a r
mostrar m o s t r a r

%A
a a

%LAMINA
lamina l a m i n a
cancion k a n th i o n

%CAMBIAR
cambiar k a m b i a r
cambia k a m b i a

%TESIS
tesis t e s i s
proyecto p r o ll e k t o

%Y
y i

%DIGIT0
cero th e r o

%DIGIT5
uno u n o
dos d o s
tres t r e s
cuatro k u a t r o
```

```
cinco th i n k o

% DIGIT9
seis s e i s
siete s i e t e
ocho o ch o
nueve n u e b e

% DIGIT10
diez d i e th
once o n th e
doce d o th e
trece t r e th e
catorce k a t o r th e
quince k i n th e

% DIECI
dieci d i e th i

% DIGIT20
veinte b e i n t e

% DIGIT2X
veinti b e i n t i

% DIGIT30
treinta t r e i n t a

% DIGIT40
cuarenta k u a r e n t a
```

```
% DIGIT50
cincuenta th i n k u e n t a

% DIGIT60
sesenta s e s e n t a

% DIGIT70
setenta s e t e n t a

% DIGIT80
ochenta o ch e n t a

% DIGIT90
noventa n o b e n t a

% ACCION_A
activar a k t i b a r
activa a k t i b a
desactivar d e s a k t i b a r
desactiva d e s a k t i b a

% ACTIVAS
activas a k t i b a s

% ACCESIBILIDAD
orca o r k a
teclado t e k l a d o
magnificador m a g n i f i k a d o r

% DECIR
decir d e th i r
```

```
dime d i m e

%TITULO
titulo t i t u l o

%ACTIVO
cancion k a n th i o n
ventana b e n t a n a

%DATE
hora o r a
fecha f e ch a

%NOMBRE
vozcontrol b o th k o n t r o l
```

A.3. Script de entrenamiento para Julius

```
#! /bin/bash
# coding: utf-8
##########################################
##
## archivo: entrenamientoJulius.sh
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
```

```
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
##########################################

# generamos los archivos dict, term, dfa
#mkdfa.pl ordenes

echo "Creamos una lista de palabras"
#perl ../HTK_scripts/prompts2wlist prompts.txt wlist
../julia/bin/julia ../bin/prompts2wlist.jl prompts.txt wList
```

```
#echo "Agregando SENT-END y SENT-START a wlist"
#echo SENT-END > wlistTemp
#echo SENT-START >> wlistTemp
#cat wlist >> wlistTemp
#rm wlist
#mv wlistTemp wlist

echo "Creando monophones1"
HDMan -A -D -T 1 -m -w wList -n monophones1 -i -l dlog dict
    ../lexicon/VoxForgeDict.txt

echo "Creando monophones0 copiando a monophones1 y eliminando
    'sp'"
cat monophones1 | grep -v sp >monophones0
#cp monophones1 monophones0

echo "Paso 3 - Creando words.mlf"
#perl ../HTK_scripts/prompts2mlf words.mlf prompts.txt
../julia/bin/julia ../bin/prompts2mlf.jl prompts.txt words.mlf
HLEd -A -D -T 1 -l '*' -d dict -i phones0.mlf ../input_files/
    mkphones0.led words.mlf

echo "Paso 4"
HLEd -A -D -T 1 -l '*' -d dict -i phones1.mlf ../input_files/
    mkphones1.led words.mlf

echo "Paso 5 - convirtiendo de wav a mfc"
HCopy -A -D -T 1 -C ../input_files/wav_config -S codetrain.scp

echo "Paso 6 - eliminando las carpetas hmm viejas"
rm -Rf hmm[0-9] hmm1[0-5]
```

```
echo "Creando las carpetas hmm0...hmm15"
mkdir hmm0 hmm1 hmm2 hmm3 hmm4 hmm5 hmm6 hmm7 hmm8 hmm9 hmm10
    hmm11 hmm12 hmm13 hmm14 hmm15

HCompV -A -D -T 1 -C ../input_files/config -f 0.01 -m -S train
    .scp -M hmm0 ../input_files/proto

echo "Creando en hmm0: hmdefs y macros"
../script/./hmmdefs.py
../script/./macros.py

echo "Restimando los modelos hmmdefs y macros en hmm1, hmm2 y
    hmm3"
HERest -A -D -T 1 -C ../input_files/config -I phones0.mlf -t
    250.0 150.0 1000.0 -S train.scp -H hmm0/macros -H hmm0/
    hmmdefs -M hmm1 monophones0
HERest -A -D -T 1 -C ../input_files/config -I phones0.mlf -t
    250.0 150.0 1000.0 -S train.scp -H hmm1/macros -H hmm1/
    hmmdefs -M hmm2 monophones0
HERest -A -D -T 1 -C ../input_files/config -I phones0.mlf -t
    250.0 150.0 1000.0 -S train.scp -H hmm2/macros -H hmm2/
    hmmdefs -M hmm3 monophones0

echo "Paso 7- copiando hmm3 a hmm4"
cp hmm3/* hmm4/

echo "Paso 8 - Agregando estado 'sp'"
../script/./sp.py

echo "Paso 9- hmm5"
```

```
HHEd -A -D -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 ../
    input_files/sil.hed monophones1

echo"Paso 10 - hmm6 y hmm7"
HERest -A -D -T 1 -C ../input_files/config -I phones1.mlf -t
    250.0 150.0 3000.0 -S train.scp -H hmm5/macros -H hmm5/
    hmmdefs -M hmm6 monophones1
HERest -A -D -T 1 -C ../input_files/config -I phones1.mlf -t
    250.0 150.0 3000.0 -S train.scp -H hmm6/macros -H hmm6/
    hmmdefs -M hmm7 monophones1

echo "Paso 11 - Alineación"
HVite -A -D -T 1 -l '*' -o SWT -b SENT-END -C ../input_files/
    config -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m -t
    250.0 150.0 1000.0 -y lab -a -I words.mlf -S train.scp dict
     monophones1> hvite_log

echo "Paso 12- hmm8 y hmm9"
HERest -A -D -T 1 -C ../input_files/config -I aligned.mlf -t
    250.0 150.0 3000.0 -S train.scp -H hmm7/macros -H hmm7/
    hmmdefs -M hmm8 monophones1
HERest -A -D -T 1 -C ../input_files/config -I aligned.mlf -t
    250.0 150.0 3000.0 -S train.scp -H hmm8/macros -H hmm8/
    hmmdefs -M hmm9 monophones1

echo "Paso 13"
HLEd -A -D -T 1 -n triphones1 -l '*' -i wintri.mlf ../
    input_files/mktri.led aligned.mlf

echo "Paso 14: creando el archivo 'mktri.hed'"
```

```
../julia/bin/julia ../bin/mktrihed.jl monophones1 triphones1
    mktri.hed

echo "creamos hmdfes y macros para la carpeta hmm10"
HHEd -A -D -T 1 -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.
    hed monophones1

echo "Para hmm11 y hmm12"
HERest  -A -D -T 1 -C ../input_files/config -I wintri.mlf -t
    250.0 150.0 3000.0 -s stats -S train.scp -H hmm10/macros -H
     hmm10/hmmdefs -M hmm11 triphones1
HERest  -A -D -T 1 -C ../input_files/config -I wintri.mlf -t
    250.0 150.0 3000.0 -s stats -S train.scp -H hmm11/macros -H
     hmm11/hmmdefs -M hmm12 triphones1

echo "Paso 15- triphones"
HDMan -A -D -T 1 -b sp -n fulllist0 -g ../input_files/
    maketriphones.ded -l flog dict-tri ../lexicon/VoxForgeDict.
    txt
../julia/bin/julia ../bin/fixfulllist.jl fulllist0 monophones0
     fulllist

echo  "copiamos tree1.hed a tree.hed"
cat ../input_files/tree1.hed > tree.hed
#perl ../HTK_scripts/mkclscript.prl TB 350 monophones0 >> tree
    .hed
#cat tree >> tree.hed
../julia/bin/julia ../bin/mkclscript.jl monophones0 tree.hed

echo "para hmm13"
```

```
HHEd -A -D -T 1 -H hmm12/macros -H hmm12/hmmdefs -M hmm13 tree
    .hed triphones1

echo "para hmm14 y hmm15"
HERest -A -D -T 1 -T 1 -C ../input_files/config -I wintri.mlf
     -t 250.0 150.0 3000.0 -S train.scp -H hmm13/macros -H
    hmm13/hmmdefs -M hmm14 tiedlist
HERest -A -D -T 1 -T 1 -C ../input_files/config -I wintri.mlf
     -t 250.0 150.0 3000.0 -S train.scp -H hmm14/macros -H
    hmm14/hmmdefs -M hmm15 tiedlist

echo "Listo, puedes usar 'hmm15/hmmdes' y 'tiedlist' para
    hacer reconocimiento"
echo "Solo ejecuta:
julius -input mic -C julian.jconf"
```

Apéndice B

Archivos del sistema gpyVozControl

B.1. Archivo principal gpyvozcontrol.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: gpyvozcontrol.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
```

```
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

import sys
import os
import pyatspi
import wnck
import pygtk
import gtk
import time
import locale
import webbrowser
from multimedia import Multimedia
from aplicaciones import Aplicaciones, Accesibilidad, Carpetas
from evince import Evince
from speak import Reproducir
```

```
from sistema import Sistema
from clientecorreo import ClienteDeCorreo
from numero import ConvertirNumero

class CommandAndControl:
  def __init__(self, file_object):
    self.mediaplayer = Multimedia()
    self.aplications = Aplicaciones()
    self.folders = Carpetas()
    self.evince = Evince()
    self.system = Sistema()
    self.correo = ClienteDeCorreo()

    os.system('amixer sset "Capture" 49800')
    print 'GPYVOZCONTROL - Sistema de órdenes de voz en
       español con Julius'
    Reproducir('Bienvenido a GPYVOZCONTROL, el sistema de
       órdenes de voz en español con Julius')
    Reproducir('Espera mientras se carga el modelo acústico')
    for i in range(1, 4):
      Reproducir('%s' % str(4-i))
      time.sleep(1)
    Reproducir('Listopara recibir órdenes, ya puedes comenzar'
       )

    startstring = 'sentence1: <s> '
    endstring = ' </s>'

    while 1:
      line = file_object.readline()
      screen = None
```

```
        screen = wnck.screen_get_default()

        if not line:
          break
        if 'missing phones' in line.lower():
          print 'Error: Missing phonemes for the used grammar
             file.'
          sys.exit(1)
        if line.startswith(startstring) and line.strip().
           endswith(endstring):
          self.parse(line.strip('\n')[len(startstring):-len(
             endstring)], screen)

    def parse(self, line, screen):
      # Parse the input
      params = [param.lower() for param in line.split() if param
         ]
      if not '-q' in sys.argv and not '--quiet' in sys.argv:
        print 'Entrada reconocida:', ' '.join(params).
           capitalize()
        print "La primera palabra es: ", params[0]
        f = open('.orden', 'r')
        self.k = int(f.read())
        f.close()

      # Activa la bandera
      if params[-1] == 'vozcontrol':
        os.system('aplay .wavs/tono2.wav &')
        os.system('echo 1 > .orden')

      # Ejecuta el comando si es reconocido.
```

```
if self.k == 1:
  Reproducir('Espera un momento')
  if params[-1] == 'cancelar' or params[-1] == 'cancela':
    Reproducir('No se ejecuta ninguna orden')
  elif params[-1] == 'reproductor':
    command = self.mediaplayer.parse(params[0])
    if params[0] == 'abrir' or params[0] == 'abre':
      os.system(command)
      Reproducir('%s fue activado' %self.mediaplayer.name
          )
    elif command:
      if os.system('ps -A | grep rhythmbox > /dev/null')
         == 0 or os.system('ps -A | grep banshee > /dev/
         null') == 0:
        os.system(command)
        Reproducir('%s %s a %s' %(params[0], params[1],
            self.mediaplayer.name))
      else:
        print'No tienes un reproductor abierto para
            realizar esa acción'
        Reproducir('No tienes un reproductor abierto para
            realizar esa acción')
    elif not '-q' in sys.argv and not '--quiet' in sys.
       argv:
      print 'Acción no válida para %s' %self.mediaplayer.
          name
      Reproducir('Acción no válida para %s' %self.
          mediaplayer.name)
  elif params[0] == 'cancion':
    command = self.mediaplayer.parse(params[-1])
```

```
        if os.system('ps -A | grep rhythmbox > /dev/null') ==
           0 or os.system('ps -A | grep banshee > /dev/null')
           == 0:
          os.system(command)
          Reproducir('Reproduciendo la %s %s' %(params[-1],
             params[0]))
        else:
          print'No tienes un reproductor abierto'
          Reproducir('No tienes un reproductor abierto para
             realizar esa acción')
      elif params[-1] == 'cancion':
        command = self.mediaplayer.parse(params[0])
        if os.system('ps -A | grep rhythmbox > /dev/null') ==
           0 or os.system('ps -A | grep banshee > /dev/null')
           == 0:
          if params[0] == 'próximo' or params[0] == 'cambiar':
            os.system(command)
            Reproducir('Se cambia la canción')
          else:
            print "No entiendo la orden"
            Reproducir("Ups! no entiendo la orden")
        else:
          print'No tienes un reproductor abierto'
          Reproducir('No tienes un reproductor abierto para
             realizar esa acción')
      elif params[0] == 'dime' or params[0] == 'decir':
        if params[-1] =='hora' or params[-1] == 'fecha':
          self.SayDate(params[-1])
        else:
          self.Actions(self.Acciones(screen), 'titulo', 2)
      elif params[0] == 'titulo':
```

```
        if params[-1] == 'cancion':
            command = self.mediaplayer.parse(params[0])
            if os.system('ps -A | grep rhythmbox > /dev/null')
               == 0 or os.system('ps -A | grep banshee > /dev/
               null') == 0:
                titulo = commands.getoutput(command)
                Reproducir(str(titulo))
            else:
                print'No tienes un reproductor abierto'
                Reproducir('No tienes un reproductor abierto para
                   realizar esa acción')
        else:
            self.Actions(self.Acciones(screen), 'titulo', 1)
    elif params[0] == 'abrir' or params[0] == 'abre':
        if params[1] == 'carpeta':
            command = self.folders.parse(params[-1])
            os.system(command)
            Reproducir('Carpeta %s fue abierta' %params[-1])
        elif params[-1] == 'correo':
            command = self.correo.parse(params[0])
            os.system(command)
            Reproducir('El cliente de correo %s fue abierto' %
               self.correo.name)
        elif params[-1] == 'proyecto' or params[-1] == 'wiki'
           or params[-1] == 'navegador' or params[-1] == '
           gmail':
            command = self.aplications.parse(params[-1])
            webbrowser.open_new_tab(command)
            Reproducir('La página del %s fue abierta' %params
               [-1])
        else:
```

```
            command = self.aplications.parse(params[-1])
            os.system(command)
            Reproducir('La aplicación %s fue abierta' %params
                [-1])
        elif params[-1] == 'sistema' or params[-1] == 'equipo':
          command = self.system.parse(params[0])
          os.system(command)
          if params[0] == 'apagar' or params[0] == 'reiniciar':
            Reproducir('El %s se va a %s' %(params[-1], params
                [0]))
          else:
            #texto = '%s %s al %s' %(params[0], params[1],
                params[-1])
            Reproducir('%s %s al %s' %(params[0], params[1],
                params[-1]))
        elif params[0] == 'presentar':
          os.system('evince -f -s /home/joenco/Entrega-1.pdf &')
          Reproducir('Proyecto de grado fue abierto')
        elif params[0] == 'lamina':
          command = self.evince.parse(params[-1])
          if os.system('ps -A | grep evince > /dev/null') == 0:
            os.system(command[0])
            Reproducir(command[1])
          else:
            print'No tienes un Evince abierto'
            Reproducir('No tienes un Evince abierto')
        elif params[-1] == 'lamina':
          command = self.evince.parse(params[0])
          if os.system('ps -A | grep evince > /dev/null') == 0:
            os.system(command[0])
            Reproducir(command[1])
```

```
        else:
            print 'No tienes un Evince abierto'
            Reproducir('No tienes un Evince abierto')
        elif params[0] == 'maximizar' or params[0] == 'maximiza'
            or params[0] == 'minimizar' or params[0] == '
            minimiza' or params[0] == 'reducir' or params[0] == '
            reduce' or params[0] == 'ampliar' or params[0] == '
            amplia' or params[0] == 'cerrar' or params[0] == '
            cierra':
            if params[-1] != 'maximizar' and params[-1] != '
                maximiza' and params[-1] != 'minimizar' and params
                [-1] != 'minimiza' and params[-1] != 'reducir' and
                params[-1] != 'reduce' and params[-1] != 'ampliar'
                and params[-1] != 'amplia' and params[-1] != '
                cerrar' and params[-1] != 'cierra':
                if params[1] == 'todas' or params[1] == 'todo':
                    self.Actions(self.Acciones(screen), params[0], 1)
                    return True
                else:
                    self.Actions(self.Acciones(screen), params[0], 2)
                    return True
            else:
                self.Actions(self.Acciones(screen), params[0], 2)
                return True
        elif params[0] == 'cambiar' or params[0] == 'cambia':
            linea = ConvertirNumero(line)
            words = [param.lower() for param in linea.split() if
                param]
            command = self.evince.parse(words[-1])
            if os.system('ps -A | grep evince > /dev/null') == 0:
                os.system(command[0])
```

```
            Reproducir(command[1])
          else:
            print'No tienes un Evince abierto'
            Reproducir('No tienes un Evince abierto')
        else:
          print '¡Ups! No entiendo la orden'
          Reproducir('¡Ups! No entiendo la orden')
        os.system('echo 0 > .orden')
        screen = None

    def Acciones(self, screen):
      while gtk.events_pending():
        gtk.main_iteration()
        print "... force update ..."
        ventana = wnck.Screen.force_update(screen)
        print ventana
        print "... force update ..."
        windows = screen.get_windows()
        for w in windows:
          return windows

    def Actions(self, windows, accion, n):
      for w in windows:
        window = wnck.Screen.get_active_window(w.get_screen())

      if accion == 'maximizar' or accion == 'maximiza':
        if n == 2:
          win =  wnck.Screen.get_windows(wnck.screen_get_default
              ())
          for w in win:
            w.unminimize(0)
```

```
        #window.unminimize(0)
        print "Maximizando la ventana "+window.get_name()
        Reproducir("Se maximiza la ventana "+window.get_name()
            )
      if n == 1:
        print "Maximizando todas las ventanas"
        win =  wnck.Screen.get_windows(wnck.screen_get_default
            ())
        for w in win:
          w.unminimize(0)
        Reproducir("Se maximizan todas las ventanas")
    if accion == 'minimizar' or accion == 'minimiza':
      if n == 2:
        window.minimize()
        print "Minimizando la ventana "+window.get_name()
        Reproducir("Se minimiza la ventana "+window.get_name()
            )
      if n == 1:
        print "Minimizando todas las ventanas"
        for w in windows:
          w.minimize()
        Reproducir("Se minimizan todas las ventanas")
    if accion == 'cerrar' or accion == 'cierra':
      if n == 2:
        window.close(int(time.time()))
        print 'Cerrando la ventana '+window.get_name()
        Reproducir('Cerrando la ventana '+window.get_name())
      if n == 1:
        win =  wnck.Screen.get_windows(wnck.screen_get_default
            ())
        for w in win:
```

```
            w.close(int(time.time()))
        print "Se cierran todas las ventanas"
        Reproducir("Se cierran todas las ventanas")
    if accion == 'ampliar' or accion == 'amplia':
      if n == 2:
        window.maximize()
        print "Ampliando la ventana "+window.get_name()
        Reproducir("Se amplía la ventana "+window.get_name())
      if n == 1:
        print "Ampliando todas las ventanas"
        for w in windows:
          w.maximize(0)
        Reproducir('Se amplían todas las ventanas')
    if accion == 'reducir' or accion == 'reduce':
      if n == 2:
        window.unmaximize()
        print "Se reduce la ventana "+window.get_name()
        Reproducir("Se reduce la ventana "+window.get_name())
      if n == 1:
        print "Reduce todas las ventanas"
        for w in windows:
          w.unmaximize(0)
        Reproducir('Se Reducen todas las ventanas')
    if accion == 'titulo':
      if n == 1:
        win =  wnck.Screen.get_windows(wnck.screen_get_default
            ())
        print "La ventana activa es "+window.get_name()
        Reproducir("La ventana activa es "+window.get_name())
      else:
        Reproducir("Las ventanas activas son:")
```

```
            for w in windows:
                Reproducir(" "+w.get_name())

    def SayDate(self, date):
        locale.setlocale(locale.LC_ALL, "")
        if date == 'fecha':
            fecha = time.strftime("%A , %d de %B del %Y", time.
                localtime())
            print fecha
            Reproducir(fecha)
        if date == 'hora':
            hora = time.strftime("son las %I y %M minutos")
            print hora
            Reproducir(hora)

if __name__ == '__main__':
    try:
        CommandAndControl(sys.stdin)
    except KeyboardInterrupt:
        sys.exit(1)
```

B.2. Archivo aplicaciones.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: aplicaciones.py
## version: 0.1
## autor: Jorge Ortega
```

```
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
##########################################

class Aplicaciones:
  name = "Aplicaciones"
```

```
  aplications = {
      'texto': 'gedit',
      'documentos': 'lowriter',
      'documento': 'lowriter',
      'calculo': 'localc',
      'hoja': 'localc',
      'calculadora': 'gnome-calculator',
      'wiki' : 'https://bitbucket.org/gproyectogrado/
        gpyvozcontrol/wiki/Home',
      'proyecto' : 'https://bitbucket.org/gproyectogrado/
        gpyvozcontrol',
      'mensajeria': 'pidgin',
      'terminal': 'gnome-terminal',
      'consola': 'gnome-terminal',
      'presentacion': 'loimpress',
      'navegador': 'http://www.google.com',
      'gmail' : 'http://gmail.com',
  }

  def parse(self, word):
    if word in self.aplications:
      return '%s &' % self.aplications[word]

class Accesibilidad:
  name = "Accesibilidad"

  assesibility = {
      'orca': 'orca',
      'teclado': 'florence',
      'magnificador': 'magnifier',
```

```
  }

  def parse(self, word):
    if word in self.assesibility:
      return '%s' % self.assesibility[word]

class Carpetas:
  name = "Carpetas"

  folders = {
      'documentos': 'Documentos',
      'descargas': 'Descargas',
      'musica': 'Música',
      'videos': 'Vídeos',
      'personal': '',
  }

  def parse(self, word):
    if word in self.folders:
      return 'nautilus /home/joenco/%s' % self.folders[word]
```

B.3. Archivo evince.py

```
#! /usr/bin/python -u
# coding: utf-8
##############################################
##
## archivo: evince.py
## version: 0.1
## autor: Jorge Ortega
```

```
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
##############################################

import os
import re
```

```
from pyPdf import PdfFileReader
from speak import Reproducir

input1 = PdfFileReader(file("/home/joenco/Entrega-1.pdf", "rb"
    ))
paginas = input1.getNumPages()

class Evince:

  def parse(self, word):
    pag = []
    if word == 'anterior':
      pag = self.nro_pag(1, 0)
    elif word == 'siguiente' or word == 'próximo' or word == '
       cambiar':
      pag = self.nro_pag(0, 0)
    elif word == 'primera':
      pag = self.nro_pag(2, 0)
    elif word == 'ultima':
      pag = self.nro_pag(3, 0)
    elif re.findall('[0-9]+', word):
      pag = self.nro_pag(4, int(word))
    elif word == 'actual':
      pag = self.nro_pag(5, 0)
    else:
      pag = self.nro_pag(6, 0)

    return 'evince -s -f -i %s /home/joenco/Entrega-1.pdf' %
       str(pag[0]), pag[1]

  def nro_pag(self, n, np):
```

```
file = open('.pagina', 'r')
pag = int(file.read())
file2 = open('.paginatemp', 'w')
t=0
texto = ""

if n == 1:
  if pag != 1:
    file2.write('%s' %str(pag-1))
    pag = pag - 1
  else:
    texto = 'Estoy en la primera lámina, no puedo ir a la
       anterior'
    file2.write('%s' %str(pag))
    t=1
elif n == 0:
  if pag != paginas:
    file2.write('%s' %str(pag+1))
    pag = pag + 1
  else:
    texto = 'No puedo ir a la siguiente, no hay mas
       láminas'
    file2.write('%s' %str(pag))
    t=1
elif n == 2:
  if pag != 1:
    file2.write('%s' %str(1))
    pag = 1
  else:
    file2.write('%s' %str(1))
    texto = 'Ya estoy en la primera lámina'
```

```
        t=1
    elif n == 3:
      if pag != paginas:
        file2.write('%s' %str(paginas))
        pag = paginas
      else:
        texto = 'Ya estoy en la última lámina'
        t=1
        file2.write('%s' %str(pag))
    elif n == 4:
      if pag != int(np) and int(np) <= paginas:
        file2.write('%s' %str(np))
        pag = int(np)
      elif int(np) > paginas:
        texto = 'No puedo ir a la lámina %s, solo hay %s
           láminas' %(int(np), paginas)
        t=1
        file2.write('%s' %str(pag))
      else:
        texto = 'Ya estoy en la lámina %s' %str(np)
        file2.write('%s' %str(pag))
        t=1
    elif n == 5:
      texto = "Estoy en la lámina %s" %int(pag)
      file2.write('%s' %str(pag))
    else:
      texto = '¡Ups!, no entiendo la orden, repite de nuevo'
      file2.write('%s' %str(pag))
      t = 1

    if t == 0:
```

```
            texto = 'Cambiando a la lámina %s' %str(pag)
        file.close()
        file2.close()
        os.system('cat .paginatemp > .pagina')

        return int(pag), texto
```

B.4. Archivo reproductor.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: reproductor.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
```

```
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

class Rhythmbox:
  name = "Rhythmbox"

  commands = {
      'abrir': 'no-start --play',
      'abre': 'no-start --play',
      'titulo': 'print-playing',
      'siguiente': 'next',
      'próximo': 'next',
      'cambiar': 'next',
      'anterior': 'previous',
      'mostrar': 'notify',
      'pausa': 'pause',
      'cerrar': 'quit',
      'cierra': 'quit',
      'bajar': 'volume-down',
```

```
        'subir': 'volume-up',
    }

    def parse(self, word):
        if word in self.commands:
            return 'rhythmbox-client --%s' % self.commands[word]

class Banshee:
    name = "Banshee"

    commands = {
        'abrir': 'play',
        'abre': 'play',
        'reproducir': 'play',
        'pausa': 'pause',
        'detener': 'stop',
        'siguiente': 'next',
        'anterior': 'previous',
        'silencio': 'pause',
        'cerrar': 'quit',
        'cierra': 'quit',
    }

    def parse(self, word):
        if word in self.commands:
            return 'banshee --no-present --%s %% ' % self.commands[
                word]
```

B.5. Archivo correo.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: correo.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
```

```
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

class Thunderbild:
  name = "Thunderbild"

  commands = {
      'abrir': 'thunderbild',
      'abre': 'thunderbild',
  }

  def parse(self, word):
    if word in self.commands:
      return '%s &' % self.commands[word]

class Evolution:
  name = "Evolution"

  commands = {
      'abrir': 'evolution',
      'abre': 'evolution',
  }

  def parse(self, word):
    if word in self.commands:
      return '%s &' % self.commands[word]
```

B.6. Archivo multimedia.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: multimedia.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
```

```
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#########################################

import os, sys
from reproductor import  Rhythmbox,Banshee
from speak import Reproducir

def Multimedia():
  mediaplayer=""
  if os.system('ps xa | grep -v grep | grep banshee >/dev/null
     ') == 0:
    mediaplayer = Banshee()
  elif os.system('ps xa | grep -v grep | grep rhythmbox >/dev/
     null') == 0:
    mediaplayer = Rhythmbox()
  elif os.system('which banshee >/dev/null') == 0:
    mediaplayer = Banshee()
    os.system('bash -c "nohup banshee >/dev/null 2>&1 < &1 &
       disown %%"')
  elif os.system('which rhythmbox >/dev/null') == 0:
    mediaplayer = Rhythmbox()
  else:
    Reproducir('No se encuentra instalado ningun reproductor.
       ' \
        'Por favor debe instalar Rhythmbox o Banshee.')
    sys.exit(1)
```

```
    return mediaplayer
```

B.7. Archivo speak.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: speak.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
```

```
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

import os

# Función que permite convertir en audio un texto
def Reproducir(texto):

    os.system('pico2wave -l es-ES -w .info.wav \"%s"' %texto)
    os.system('aplay .info.wav > /dev/null')
```

B.8. Archivo numero.py

```
#! /usr/bin/env python
# coding: utf-8
#############################################
##
## archivo: numero.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
```

```
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################
import re

def ConvertirNumero(linea):
  linea = re.sub('diez', '10', linea)
  linea = re.sub('veinte', '20', linea)
  linea = re.sub('treinta', '30', linea)
```

```
linea = re.sub('cuarenta', '40', linea)
linea = re.sub('cincuenta', '50', linea)
linea = re.sub('sesenta', '60', linea)
linea = re.sub('setenta', '70', linea)
linea = re.sub('ochenta', '80', linea)
linea = re.sub('noventa', '90', linea)
linea = re.sub('once', '11', linea)
linea = re.sub('doce', '12', linea)
linea = re.sub('trece', '13', linea)
linea = re.sub('catorce', '14', linea)
linea = re.sub('quince', '15', linea)
linea = re.sub('dieci seis', '16', linea)
linea = re.sub('dieci siete', '17', linea)
linea = re.sub('dieci ocho', '18', linea)
linea = re.sub('dieci nueve', '19', linea)
linea = re.sub('0 y uno', '1', linea)
linea = re.sub('0 y dos', '2', linea)
linea = re.sub('0 y tres', '3', linea)
linea = re.sub('0 y cuatro', '4', linea)
linea = re.sub('0 y cinco', '5', linea)
linea = re.sub('0 y seis', '6', linea)
linea = re.sub('0 y siete', '7', linea)
linea = re.sub('0 y ocho', '8', linea)
linea = re.sub('0 y nueve', '9', linea)
linea = re.sub('uno', '1', linea)
linea = re.sub('dos', '2', linea)
linea = re.sub('tres', '3', linea)
linea = re.sub('cuatro', '4', linea)
linea = re.sub('cinco', '5', linea)
linea = re.sub('seis', '6', linea)
linea = re.sub('siete', '7', linea)
```

```
    linea = re.sub('ocho', '8', linea)
    linea = re.sub('nueve', '9', linea)

    return linea
```

B.9. Archivo clientecorreo.py

```
#! /usr/bin/python -u
# coding: utf-8
##############################################
##
## archivo: clientecorreo.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
```

```
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

import os
import sys
from speak import Reproducir
from correo import Thunderbild, Evolution

def ClienteDeCorreo():
  clientecorreo=""
  if os.system('ps xa | grep -v grep | grep thunderbild >/dev/
     null') == 0:
    clientecorreo = Thunderbild()
  elif os.system('ps xa | grep -v grep | grep evolution >/dev/
     null') == 0:
    clientecorreo = Evolution()
  elif os.system('which thunderbild >/dev/null') == 0:
    clientecorreo = Thunderbild()
    os.system('bash -c "nohup thunderbild >/dev/null 2>&1 < &1
        & disown %%"')
```

```
  elif os.system('which evolution >/dev/null') == 0:
    clientecorreo = Thunderbild()
  else:
    Reproducir('No se encuentra instalado ningún cliente de
       correo. ' \
        'Por favor debe instalar Thunderbild o Evolution.')
    sys.exit(1)

  return clientecorreo
```

B.10. Archivo sistema.py

```
#! /usr/bin/python -u
# coding: utf-8
##########################################
##
## archivo: sistema.py
## version: 0.1
## autor: Jorge Ortega
## Correo: libretecnologia@gmail.com
## tutor: Jacinto Dávila
## Correo: jacinto.davila@gmail.com
## Fecha: Diciembre - 2015
##
## Copyright (C) 2015 Jorge Ortega
##
## Este programa es software libre; puedes redistribuirlo y /
   o
## Modificarlo bajo los términos de la Licencia Pública
   General GNU Affero
```

```
## Publicada por la Fundación para el Software Libre; ya sea
   la versión 3.0
## De la Licencia, o cualquier versión posterior.
##
## Este programa se distribuye con la esperanza de que sea
   útil,
## Pero SIN NINGUNA GARANTÍA; ni siquiera la garantía
   implícita de
## COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR.
   Revisar
## la Licencia Publica General GNU Affero para mayor
   información.
##
## Debería haber recibido una copia de la Licencia Pública
   General GNU Affero junto con este programa.
## Si no es así, consulte <http://www.gnu.org/licenses/>.
#############################################

class Sistema():
  sistema = {
    'subir': 'amixer sset Master 5\%\+',
    'bajar': 'amixer sset Master 5\%\-',
    'apagar': 'halt',
    'reiniciar': 'reboot',
  }

  def parse(self, word):
    if word in self.sistema:
      return self.sistema[word]
```