PROCESAMIENTO DE VOZ EN ESPAÑOL MEXICANO APLICADO A ROBÓTICA MÓVIL UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE.

TESIS

Que para obtener el grado de

MAESTRÍA EN CIENCIAS DE LA INGENIERÍA ELÉCTRICA

presenta

Margarita García Fragoso

Leonardo Romero Muñoz

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Agosto 2008

PROCESAMIENTO DE VOZ EN ESPAÑOL MEXICANO APLICADO A ROBÓTICA MÓVIL UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE

Los miembros del Jurado de Examen de Grado aprueban
la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de Margarita García Fragoso

Dr. José Antonio Camarena Ibarrola
Presidente del Jurado

Dr. Leonardo Romero Muñoz
Director de Tesis

Dr. Félix Calderón Solorio
Vocal

Dr. Juan José Flores Romero
Vocal

Dr. Fernando A. Velasco Avalos
Examinador Externo
Universidad Michoacana de
San Nicolás de Hidalgo

Dr. J. Aurelio Medina Rios
Jefe de la División de Estudios de Posgrado
en Ingeniería Eléctrica

UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO
Agosto del 2008

Resumen

Este trabajo de investigación de tesis de maestría proporciona un panorama global del estado del arte involucrado en la tecnología del reconocimiento del habla y del software de robots móviles. El conocer lo complejo de las herramientas que se utilizan para la construcción de este tipo de sistemas brinda una idea de los avances que se han hecho hasta el momento en el campo del procesamiento del lenguaje natural y justifica el hecho de utilizarlas.

La investigación realizada explica cómo utilizar las herramientas de software libre CMUSphinx3 proporcionado por la Universidad Carnegie Mellon, los modelos acústicos Dimex30 proporcionados por la Universidad Autónoma de México, el sintetizador de voz Festival proproporcionado por la Universidad de Edinburgo y la plataforma de desarrollo abierta Player/Stage proporcionada por la Univeridad del Sur de California.

El reconocimiento automático del habla consiste, en su acepción más general, en la transcripción de voz a texto. A partir de este concepto se describen los pasos necesarios para realizar reconocimiento de voz a partir de CMUSphix3 y los modelos acústicos DIMEx30. Se hace uso del lenguaje de programación PERL y del sintetizador de voz Festival para indicar si el proceso de reconocimiento de voz realizado fue exitoso. Se utiliza la plataforma de desarrollo abierta Player/Stage para crear cinco aplicaciones de movimiento para un robot simulado en un ambiente de dos dimensiones dirigido mediante las órdenes verbales obtenidas en el proceso de reconocimiento de voz realizado por medio de CMUSphinx3. Finalmente, se aprovecha la clasificación de patrones proporcionado por PERL, para dirigir el robot creado en el laboratorio de sistemas computacionales del posgrado de ingeniería eléctrica.

Mediante la investigación y los resultados de este trabajo de tesis se desea fomentar el interés por esta tecnología y en un futuro poder crear un sistema propio que pueda comprender con un mayor porcentaje cualquier mensaje oral.

Abstract

ï»¿This master thesis research provides a global insight on the state of the art on the speech recognition technology and robot movil software. To get to know the complexity of the needed tools in order to build this sort of system gives us an idea of recent progress in the natural language processing field and justifies their usage.

This research explains how to use the following GPL tools: CMSphinx3 by Carnegie Mellon University; DIMEX30 acoustic models by Universidad Nacional Autonoma de Mexico; Festival Text-to-Speech tool by Edinburg University; and Player/Stage Platform by South California University.

In its more general sense, Automatic Speech Recognition can be defined as the transcription of voice into text. From this concept, the steps needed in order to perform speech recognition using CMUSphinx3 and DIMEx30 acoustic models are being described. In order to prove if the speech recognition process was successful PERL and Festival Text-to-Speech tool are being used. Player/Stage platform is being installed in order to create five robot movement aplications simulated in a 2D environment guided by the spoken commands obtained during the speech recognition process given by CMUSphinx3. Finally, the LSC robot is commanded using PERL.

By means of this research and the obtained results, this work is an attempt to increase the interest in this technology in order to be able to create in the future a complete locally developed system capable of understanding with higher precision any oral spanish message.

Índice general

Capítulo 1

Introducción

El uso de la voz es sin duda el método de comunicación más intuitivo y natural para los seres humanos [Cuetara]. La tecnología del uso de la voz en sentido general, está gozando de un interés cada vez más creciente por parte no sólo de la comunidad científica internacional, sino de la sociedad en general y el Reconocimiento Automático del Habla (RAH) presenta uno de los campos más atractivos de investigación de ésta tecnología [Oropeza06].

Gracias a los grandes avances en este campo, la imagen de la computadora dirigiéndose a los hombres con una voz de alta naturalidad e inteligibilidad y sobre todo atendiendo a sus órdenes con conversaciones absolutamente espontáneas se ve cada vez más cerca, este tipo de sistema como tantos otros es un clasificador de patrones estadísticos [Miller], que puede asociar un Modelo Oculto de Markov HMM (por sus siglas en inglés Hidden Markov Model) con cada unidad de sonido, aprende los parámetros de los modelos HMM y luego los utiliza para encontrar la secuencia de unidades de sonido más probable de una determinada señal de voz [Miller]. El proceso de aprendizaje de los parámetros se denomina entrenamiento [Martin]. El proceso de uso de éstos para deducir la secuencia más probable de unidades de una determinada señal, se conoce como decodificación o simplemente reconocimiento [Martin]. En consecuencia un sistema de reconocimiento presenta dos componentes principales: un entrenador y un decodificador.

Hasta hace pocos años, los sistemas de reconocimiento de mediana y gran complejidad estaban disponibles como prototipos de laboratorio [Alexander05]. En la actualidad empresas como IBM y Dragon Systems han abordado el terreno del mercado de gran consumo con productos de elevada calidad para dictado de textos, tanto en habla aislada como continua.

Para obtener resultados satisfactorios utilizando el lenguaje natural en el diseño y desarrollo de los sistemas de reconocedores de voz se requiere de habilidades lingüísticas [Oropeza06]. Los análisis sintáctico y semántico superficiales no son suficientes.

En México son pocos los grupos de trabajo que se dedican con éxito en términos de publicaciones a la investigación en las tecnologías del habla (Instituto Politécnico Nacional, Instituto Tecnológico de Estudios Superiores de Monterrey, Universidad de Colima, Universidad de las Américas y Universidad Nacional Autónoma de México). En el resto del país, si es que el área de investigación se desarrolla, los programas se importan, y los trabajos lingüísticos sobre reconocimiento de habla se copian de la fonética inglesa y en su mejor caso de la española.

Actualmente, el proyecto de investigación más exitoso en México dentro de esta área, es DIME (Diálogos Inteligentes Multimodales en Español). El Proyecto DIME trabaja en el diseño de herramientas útiles para el desarrollo de tecnologías del habla, entre las que se encuentran el reconocimiento de voz en español, la interpretación de intenciones lingüísticas y el modelado de diálogos con aplicaciones prácticas, además, ha etiquetado en diferentes niveles del lenguaje el español mexicano con la finalidad de crear modelos acústicos fidedignos. Bajo el contexto anterior, esta tesis básicamente plantea como objetivo fundamental indicar por qué se eligieron y cómo utilizar las herramientas de de software libre CMSphinx3, DIMEx30 y Festival para crear aplicaciones de movimiento para el robot del laboratorio de sistemas computacionales y la simulación 2D proporcionada por la plataforma de desarrollo abierta Player/Stage.

1.1. Justificación

El reconocimiento de voz es una herramienta fundamental en el área de las ciencias computacionales y cualquier desarrollo o metodología que ayude a simplificar o enriquecer su entendimiento es de utilidad.

1.2. Objetivo General

El objetivo general de este trabajo es retomar la investigación de reconocimiento de voz propuesta por el doctor Leonardo Romero Muñoz mediante la creación de un modelo de lenguaje y de una interfaz comando/respuesta para una aplicación de robótica móvil

apoyada en el uso de la plataforma de desarrollo abierta Player/Stage, las aplicaciones CMUSphinx3 y Festival y los modelos acústicos DIMEx30.

1.3. Metodología

Los principales medios materiales utilizados durante el desarrollo de esta tesis de maestría son:

- Hardware
  1. Computadora HP COMPAQ BUSINESS Desktop DC7100 CMT con sonido audio Intel integrado, procesador Pentium 4-530 a 3.00GHz, 2x256MB de memoria y 40GB de disco duro.
  2. Micrófono General Electric modelo HO98950 con conectividad PLUG 3.5MM.
  3. Robot creado por el ingeniero Antonio Concha Sánchez y el doctor Leonardo Romero Muñoz.
- Software
  1. Aplicaciones
     - CMUSphinx 3.4: implementación para realizar reconocimiento de voz.
     - lm3g2dmp 3.4: herramienta de modelado de lenguaje que facilita la construcción y prueba de bigramas y trigamas.
     - Modelos Acústicos UNAM DIMEx30: recurso creado en la UNAM que permite la creación de sistemas de reconocimiento de habla de propósito general con un amplio vocabulario y diversos locutores.
     - Player/Stage v.2.1.0rc1: proporciona una interfaz de red a una gran variedad de hardware para robots en un ambiente 2D.
     - NIST SCTk: colección de herramientas de software diseñadas para evaluar los sistemas de reconocimiento de voz.
  2. Bibliotecas
     - Build-essential: contiene una lista de los paquetes considerados esenciales para construir paquetes Debian.

- Festival: ofrece un sistema sintetizador de voz completo con varios APIs así como un ambiente propicio para el desarrollo e investigación de técnicas de síntesis de voz.
- Fontconfig: biblioteca de configuración y personalización de tipografías diseñada para localizar tipografías en el sistema y seleccionarlas según los requerimientos especificados por las aplicaciones.
- Libfreetype6-dev: motor de tipografías de alta calidad por software.
- Libglib2.0-dev: biblioteca utilizada en proyectos como GTK+, GIMP, y GNOME.
- Libgtk2.0: conjunto de herramientas multiplataforma para crear interfaces gráficas de usuario.
- Libpango1.0-dev: biblioteca para la composición y renderizado de texto que conforma el núcleo del manejo de texto y tipografías en GTK+-2.0.

3. Compiladores

   - gcc versión 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)
   - perl, v5.8.8

4. Sistema Operativo

   - Linux version 2.6.18-4-686 (Debian 2.6.18.dfsg.1-12)

Debido al hecho de que la aplicación propuesta está pensada principalmente para que los jóvenes con una preparación universitaria la utilicen, en la parte de experimentos, 30 estudiantes fueron requeridos con el propósito de probar el modelo de lenguaje creado. El promedio de edad de las personas seleccionadas fue de 33 años y se obtuvieron resultados mediante el evaluador del NIST (National Institute of Standards and Technology) sclite y la fórmula para generar precisión de palabra. Los resultados obtenidos comparados con los parámetros de éxito de la página tutorial de Sphinx3 demostraron que al comparar la salida del texto hipotético (HYP) del reconocedor de voz con el correcto o texto de referencia (REF), el trabajo de investigación fue adecuado.

1.4. Descripción de capítulos

El presente documento se ha organizado de la siguiente manera: se ha realizado una división del material de estudios en dos partes, la primera de ellas concerniente a los antecedentes teóricos necesarios y la segunda relacionada con las propuestas hechas en este trabajo de investigación.

La primera parte consta de los capítulos 2, 3, 4, 5 y 6, los cuales se organizan como se especifica a continuación. En el capítulo 2 se revisa la clasificación del software de los robots móviles haciendo especial énfasis en las aplicaciones concernientes al desarrollo y producción del habla. En el capítulo 3 se describen las características principales de la plataforma de desarrollo abierta Player/Stage. En el capítulo 4 se abordan los conocimientos mínimos necesarios para comprender de manera básica cómo es la producción, percepción y reconocimiento del habla del español mexicano. En el capítulo 5 se detallan los fundamentos del sintetizador de voz Festival. En el capítulo 6 se abordan los elementos necesarios para realizar reconocimiento de voz.

La segunda parte consta de los capítulos 7 y 8; en el capítulo 7 se propone el diseño y experimentación del modelo de lenguaje necesario para realizar el proceso de reconocimiento de voz utilizado en la interfaz de captura y respuesta de comandos orales para dirigir el modelo de un robot simulado en la plataforma de desarrollo abierta Player/Stage. En el capítulo 8 se emiten las conclusiones del trabajo de investigación.

En cuanto a los apéndices, el A indica las características de la herramienta de dominio pública para el reconocimiento de voz, CMUSphinx3. El B presenta una breve reseña del proceso que debe ser seguido para crear el modelo de lenguaje. Finalmente el apéndice C indica cómo procesa la herramienta CMUSphinx3 las palabras no encontradas en el modelo de lenguaje.

Capítulo 2

Software de los robots móviles

Todos los robots se encuentran conformados por sensores, actuadores y procesadores. Estos componentes de hardware son habilitados por medio de software, el cual enlaza los datos recibidos por los sensores con las respuestas de actuación [Gerkey02].

Una característica común a muchos robots es que tienen recursos computacionales y de almacenamiento limitados [Kramer06]. Por eso, es normal que se utilicen las plataformas de desarrollo o la computadora personal como entorno de desarrollo y los compiladores cruzados generen en la computadora el programa ejecutable que correrá en el procesador del robot [Bermejo03].

En los robots móviles el comportamiento principal es su movimiento. Los programas que se ejecutan en el robot, utilizan comandos que determinan la manera en que éste se mueve y reacciona en el entorno ante los obstáculos percibidos por sus sensores [cañas05].

Desde esta perspectiva, la generación de comportamiento en un robot móvil consiste en desarrollar el software que al ejecutarse en él genere movimiento [Cañas06]. Actualmente, el software de los robots móviles se estructura en tres niveles: *sistema operativo*, *plataforma de desarrollo* y *aplicaciones*[Cañas06].

2.1. Sistemas Operativos

La misión principal del sistema operativo es ofrecer a los programas un acceso básico al hardware del robot, permitiendo su manipulación y uso [Cañas06].

Fundamentalmente, debe permitir recoger lecturas de los sensores y enviar órdenes a los actuadores, incorporando controladores que proporcionen software de soporte de bajo

nivel a los dispositivos físicos [Cañas06].

Históricamente, los robots eran desarrollos únicos [Vaughan03b]. No se producían en serie y los programas de control se construían utilizando directamente los controladores para acceder a los dispositivos sensoriales y de actuación [Cañas06]. El sistema operativo del robot era mínimo, básicamente, una colección de controladores con rutinas para leer datos de los sensores y enviar consignas a los actuadores [Howard03]. El programa de aplicación invocaba directamente las funciones de la librería que ofrecía el fabricante en sus controladores [Cañas06]. Actualmente, gran parte de los robots incluyen sistemas operativos ad hoc [Cañas06]. Por ejemplo, ROBIOS para el robot EyeBot y GNU/Linux, para la mayoría de los centros de investigación [Montemerlo03], tal es el caso del robot utilizado en este trabajo.

2.2. Plataformas de desarrollo

Las plataformas de desarrollo han surgido con la idea de facilitar la construcción incremental de las aplicaciones robóticas [cañas05]. Suelen ofrecer una interfaz de acceso uniforme al hardware heterogéneo de los robots, una arquitectura de software concreta y un conjunto de funcionalidades comunes listas para su reutilización [Utz02].

Hoy en día los fabricantes más avanzados y algunos grupos de investigación incluyen plataformas de desarrollo para simplificar a los usuarios la programación de sus robots [Cañas06]. Por ejemplo, ActivMedia ofrece la plataforma Aria para sus robots Pioneer y PeopleBot [ARIA02]; Evolution Robotics vende su plataforma ERSP [Montemerlo03]; Sony ofrece OPEN-R para sus robots AIBO [Rico05]; TeamBots ofrece una colección de paquetes JAVA para la investigación de multi-agentes para robots móviles [Montemerlo03]; CMU ofrece Player/Stage, un proyecto de software libre diseñado como interfaz de programación [Gerkey03]; Pyro ofrece un ambiente de programación de robot con propósito educativo pero no limitado únicamente a ello [Blank03]; CARMEN es una colección de software libre escrito en C para el control de robots móviles [Montemerlo03]; MissionLab ofrece un grupo de herramientas de software para llevar a cabo planes de estilo militar, utilizando robots individuales o en grupos, reales o simulados [Missionlab06].

Sin embargo, en la actualidad no existen evaluaciones prácticas para determinar concretamente la medida de la facilidad de uso e impacto de los ambientes de desarrollo de los robots [Kramer06]. Pero en el Departamento de Ciencias Computacionales e Ingeniería

de la Universidad de Notre Dame ubicada en Indiana, Estados Unidos de América, se realizó una revisión en febrero de 2007 para evaluar nueve plataformas de desarrollo que apoyan el desarrollo de usos robóticos. Los resultados fueron compilados y usados para comparar los sistemas según sus características, facilidad de uso e impacto en la comunidad educativa y científica [Kramer06]. Finalmente, se llegó a la conclusión de que Player/Stage es actualmente la plataforma de desarrollo más eficaz, es flexible y adaptable y el número de áreas de investigación en las cuales hay publicaciones de ella son indicadores de uso acertado.

A continuación se nombran algunas de las publicaciones encontradas en Internet mediante el buscador Google que citan el uso de Player/Stage. SLAM (Simultaneous Localization and Mapping)de Dennis Wolf en 2005, Learning de Doug Jefferson Provost en 2004, Multi-Robot Coordination de Chris Jones en 2004, Multi-Robot Mapping de Andrew Howard en 2004, Multi-Robot Planning de Andrew Howard en 2004, Robotics Education for All Ages de Maja Mataric en 2004, HRI Task Allocation de Ashley Tews en 2003, Multi-Robot Localization de Andrew Howard en 2003, Multi-Robot Exploration de Andrew Howard en 2002, Multi-Robot Sensing de Boyoon Jung en 2002, Multi-Robot Formation de Jakob Fredslund en 2002, y Multi-Robot Task Allocation de Brian Gerkey en 2002.

Debido al hecho de que en esta plataforma, las aplicaciones orientadas a la tecnología del habla han sido exitosamente desarrolladas, se decidió utilizar Player/Stage en la realización de este trabajo de investigación.

2.2.1. Plataforma de desarrollo abierta Player/Stage

El objetivo de la plataforma de desarrollo abierta *Player/Stage* es desarrollar una infraestructura de software libre que promueva la investigación y proporcione un sistema estándar para utilizar los dispositivos del robot de forma sencilla, proporcionando recursos transparentes y manejando el hardware de forma segura y confiable [Howard03].
Las principales ventajas de utilizar este sistema son:

- Es una plataforma robótica estándar.
- Proporciona un prototipado rápido y prueba de algoritmos.
- Puede ser utilizado con dos simuladores (dos y tres dimensiones) para distintos tipos de experimentos.

- Permite la construcción de plugin de controladores.
- Es fácil de aprender.
- Tiene una licencia GPL.

2.2.2. Servidor Player

Player, originalmente llamado *Golem*, fue creado en el Laboratorio de Investigaciones Robóticas de la Universidad del Sur de California por Brian Gerkey y Kasper Stoy [Gerkey04]. Sin embargo, su estado actual, es el resultado de las contribuciones de muchos desarrolladores, universitarios, compañías y laboratorios de todo el mundo [Vaughan06].

Player es un servidor desarrollado inicialmente en sistemas x86/Linux y todavía es utilizado básicamente en esa plataforma. Sin embargo, con el apoyo de herramientas GNU, player es actualmente instalado, desarrollado y ejecutado en plataformas POSIX, incluyendo además algunas configuraciones de compilación cruzada. A pesar de este avance, Player aún no puede ser ejecutado en Windows [Vaughan06].

La estructura de Player fue diseñada para ocultar los detalles del hardware del robot, proporcionando un ambiente de desarrollo y un sistema de navegación comunes para las aplicaciones robóticas al ofrecer recursos lo más transparente posible [Gerkey03]. Permite el control de los dispositivos del robot, obtiene información de sus sensores y proporciona una interfaz común para hardware variado [Vaughan06]. El robot puede ser controlado mediante bibliotecas de Player (actualmente disponibles para programar en distintos lenguajes: C, C++, JAVA, PYTHON o LISP) o mediante el envío de mensajes utilizando el protocolo de comunicación de Player y el protocolo de transporte cliente/servidor TCP [Colomina07].

La figura 2.1 muestra un ejemplo de un archivo de configuración utilizado por el usuario para iniciar el servidor Player que escucha en un puerto particular TCP (por default 6665). El programa cliente, tal como un controlador de mando o visualización de información de interfaz gráfica de usuario (GUI), es iniciado y establece una conexión socket TCP con el servidor, definiendo los controladores utilizados y especificando cómo unirlos con el hardware [Gerkey05].

Después de la confirmación entre el Cliente y el Servidor, el Cliente envía mensajes informando al servidor que abra los dispositivos. Todas las comunicaciones de Player ocurren a través de dispositivos [Gerkey04]. Los dispositivos se conforman de controladores e

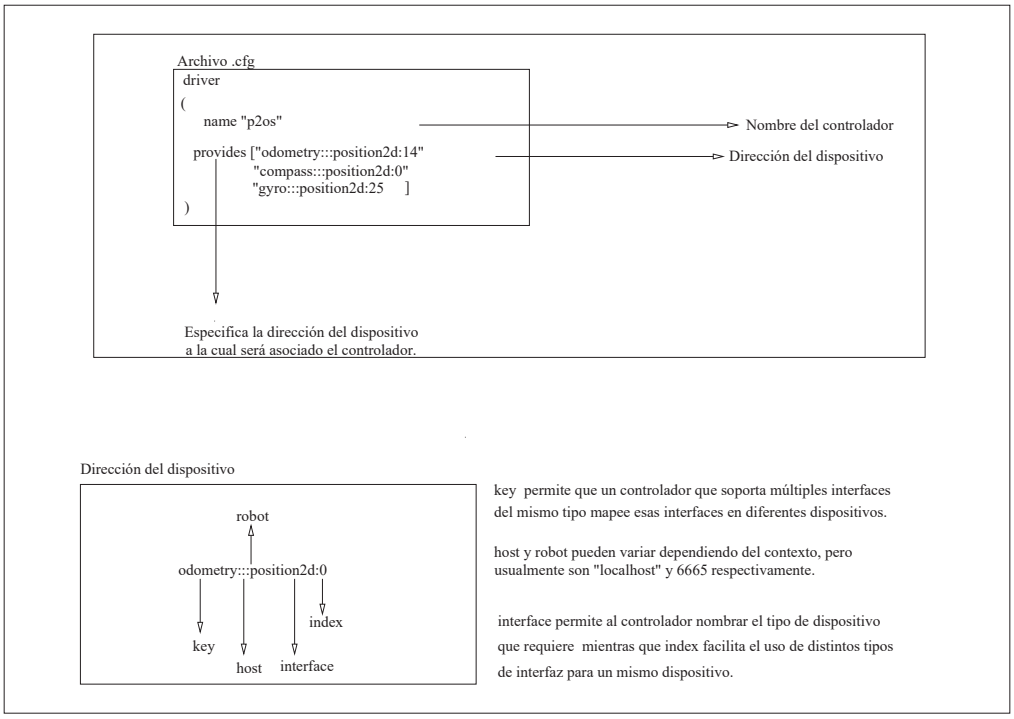

Figura 2.1: Ejemplo de un archivo .cfg para player

interfaces para realizar algún servicio [Gerkey05]. La interfaz define la sintaxis y la semántica de todos los mensajes recibidos de los controladores y, el controlador es el software (por lo general escrito en C++) que se dirige a un sensor robótico, actuador o algoritmo y traduce sus entradas y sus salidas ocultando sus especificaciones técnicas a la interfaz [Gerkey05]. El Servidor alimenta continuamente al Cliente con información de los dispositivos y puede aceptar también comandos o peticiones de configuración del cliente; este intercambio de información continua hasta que el cliente se desconecta.

Actualmente, la frecuencia de actualización de sensores y actuadores es independiente. Ésto proporciona a los clientes la capacidad de hacer pleno uso de los datos generados por los dispositivos que funcionan a alta frecuencia, evitando ser obstaculizados por aquellos que son lentos [Kramer06].

Para su correcto funcionamiento, Player consta de cuatro abstracciones clave representadas en capas separadas [Vaughan06]: *Interfaz Abstracta de Dispositivo de Player* que permite la portabilidad y la reusabilidad del código de Player [Vaughan03a]; *Protocolo*

de Mensajes que inspecciona y controla el comportamiento de Player [Vaughan03a]; *Mecanismo de Transporte* que especifica la forma en que los paquetes son seriados, direccionados y transmitidos [Vaughan06]; y *Mecanismo de Implementación* que especifica el diseño de Player [Vaughan06].

2.2.3. Simulador 2D Stage

Stage, originalmente llamado *Arena*, es un simulador en dos dimensiones para robots móviles creado por Andrew Howard y Richard Vaughan en el Laboratorio de Investigaciones Robóticas de la Universidad del Sur de California [Gerkey04]. Aunque, como en el caso de Player, su estado actual es el resultado de las contribuciones de muchos desarrolladores, universitarios, compañías y laboratorios de todo el mundo [Vaughan06].

Stage ofrece un mundo virtual poblado de robots móviles y otros objetos que los robots pueden detectar y manipular. Usualmente se utiliza para proporcionar dispositivos simulados al servidor de robot Player, pero también puede ser utilizado como una librería de simulación robótica [Gerkey03].

El motor de simulación de Stage es implementado como la biblioteca independiente C *libstage*. *libstageplugin* es la envoltura alrededor de libstage que lo une con la arquitectura del controlador de Player en el momento de su arranque [Gerkey03].

El diseño de Stage está basado en sistemas multi-proceso. Proporciona un compromiso entre una simulación de gran calidad, una simulación mínima útil y las simulaciones basadas en inteligencia artificial, por lo que, es suficientemente realista para permitir a los usuarios utilizar los mismos programas de control entre los robots de Stage y los del mundo real [Howard03], dispone de modelos de la mayoría de los dispositivos robóticos simples y computacionalmente no muy pesados como: *sensores de rango ultrasónicos* e *infrarrojos*, *sensores para encontrar y seguir ciertos colores*, *sensores para encontrar información sobre la identidad de un determinado objeto*, *bases de robot con localización odométrica o global* y *reconocimiento y síntesis de voz* [Vaughan06]. La simulación de estos dispositivos se define en un archivo *World* [Gerkey05].

La figura 2.2 es un ejemplo del archivo World, el cual, describe el mundo simulado por Stage [Gerkey05]. Describe los robots, los sensores, los actuadores, los objetos móviles e inmóviles. El archivo World también puede ser utilizado para controlar varios aspectos del motor de simulación tal como su velocidad y fidelidad [Gerkey03]. Las unidades de

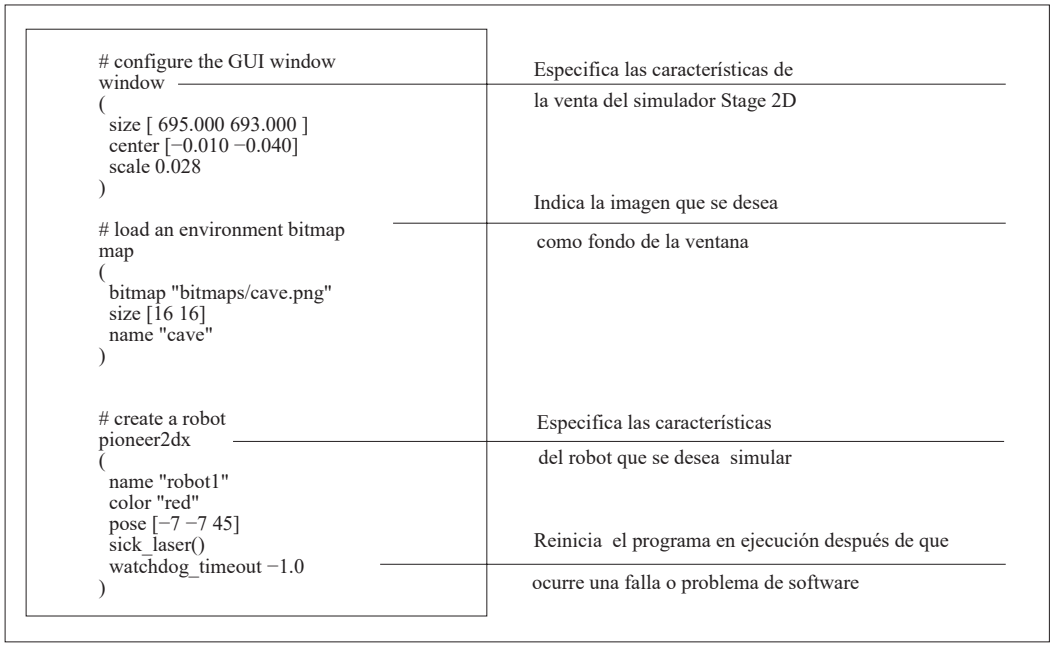

Figura 2.2: Ejemplo de un archivo .world para stage

medida que maneja son el *metro*, el *centímetro*, el *milímetro*, el *segundo*, el *grado* y el *radian* [Gerkey05].

La sintaxis básica de un *archivo World* incluye *comentarios*, *entidades* y *propiedades* [Gerkey03]. Todas las *entidades* se componen de uno o varios *modelos* [Gerkey05]. Cada modelo puede o no tener *propiedades* asociadas a él. Las *propiedades* especifican las características como la forma del objeto o el alcance del sensor [Howard03]. Los modelos se encuentran organizados en una jerarquía con subtipos que heredan *propiedades* del tipo padre [Gerkey03]. Todos los objetos pasivos como las cajas o los mapas de bits son derivados de una *entidad abstracta básica* y son referenciados como objetos [Howard03]. De manera similar, todos los modelos de sensores y actuadores son derivados de un *dispositivo abstracto* de tipo genérico y son referenciados como dispositivos [Gerkey05].

La figura 2.3 muestra la ventana convencional con un menú y un área principal que presenta Stage para realizar la simulación [Gerkey03]. El usuario puede mover, rotar, agrandar o minimizar las entidades del ambiente de Stage mediante el ratón y el teclado [Gerkey03].

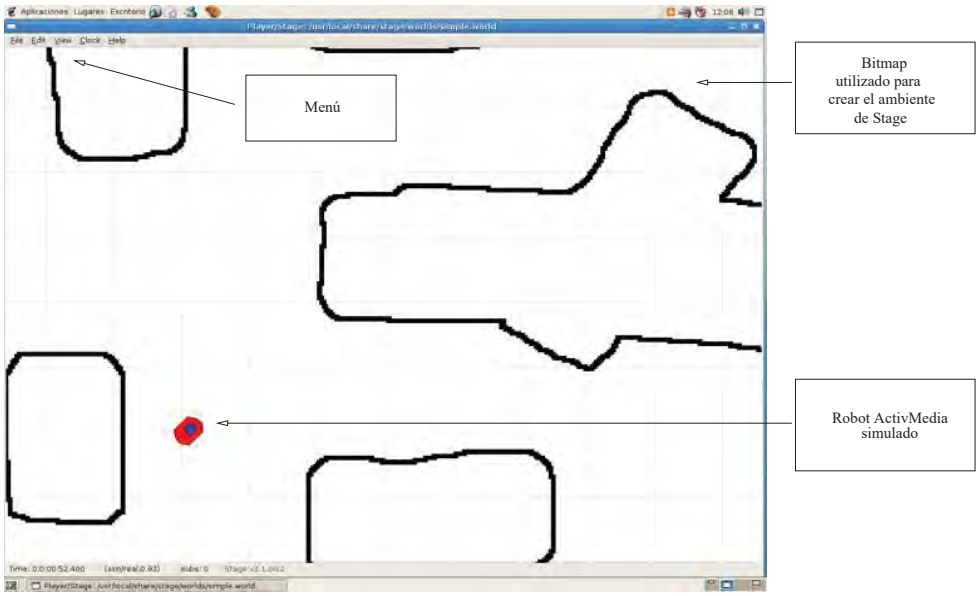

Figura 2.3: Ejemplo de la ventana de Stage

2.3. Aplicaciones

Hoy en día existen varios factores que dificultan la tarea de los programadores de robots [Kramer06]. En cuestión de hardware, existe una gran diversidad de dispositivos sensoriales y de actuación y, por lo tanto de interfaces. El programador debe dominarlos para acceder a ellos desde las aplicaciones.

Mientras que en muchos campos de la informática sí hay bibliotecas que un programador puede emplear para construir su propio programa, en el software de robots no hay un marco homogéneo ni estándares que propicien la reutilización de código [cañas05]. Cada robot concreto, requiere de una aplicación prácticamente construida desde cero [Cañas06].

Por otra parte, el software de robots es cada vez más distribuido [Woo03]. Es usual que las aplicaciones de robots tengan que establecer alguna comunicación con otros procesos ejecutándose en la misma máquina o en una diferente. La distribución ofrece posibilidades ventajosas como ubicar la carga computacional en nodos con mayor capacidad; y en sistemas multirrobot, hace posible la integración sensorial, la centralización y la coordinación [Woo03].

En lo referente al comportamiento inteligente, el conocimiento sobre cómo generarlo es muy limitado [Kordic05]. La división del comportamiento de robots en unidades básicas sigue siendo materia de investigación [Cañas06]. No hay una guía universalmente admitida sobre cómo organizar el código de las aplicaciones de robots para que sean escalables y se puedan reutilizar sus partes [Kramer06]. Cada desarrollador escribe su aplicación combinando ad hoc los bloques de código que existen en su entorno [Kramer06].

En cuanto a las herramientas utilizadas en la programación de robots, los lenguajes que se emplean para programarlos no poseen diferencias significativas respecto a los utilizados en las aplicaciones informáticas tradicionales [Kramer06]. Su objetivo básico es incluir en la propia sintaxis del lenguaje mecanismos como la descomposición de tareas, la monitorización de ejecución o la sincronización que resulten ventajosos para la programación de robots [Cañas06].

Aunque han existido intentos de establecer lenguajes específicos para programar robots, como Task Description Language (TDL) o Reactive Action Packages (RAP) [Firby94], nunca han sido de uso general y, actualmente el lenguaje de programación más extendido para programar robots es C [Kramer06].

Otras aplicaciones muy populares en la actualidad son las interfaces hombre-máquina controladas por voz, los sistemas de respuesta vocal interactiva y la automatización de los sistemas telefónicos [Fernandez05].

El futuro de dichas aplicaciones se encuentra condicionado al desarrollo de la tecnología del habla. Tecnología que involucra un amplio conjunto de conocimientos y procedimientos de distintas ramas del saber como son: fisiología, acústica, lingüística, procesamiento de señales, inteligencia artificial, teoría de la comunicación y de la información y ciencia de la computación [Bermejo03].

La tecnología del habla se estructura en cuatro tecnologías básicas principales [Alexander05]: *reconocimiento de voz* (conversión de un mensaje hablado en texto) [Jiang04], *síntesis de voz* (conversión de texto a voz) [Fernandez05], *reconocimiento de locutores* (proceso para determinar la identidad de un hablante perteneciente a una población de hablantes pre-establecida a través del análisis de su voz) [Alexander05] y *codificación de voz* (transmisión y almacenamiento de la señal de voz en forma digital eficiente y sin pérdida de calidad) [Alexander05].

El reconocimiento del habla será brevemente descrito en la siguiente sección dado que el tema de esta investigación se basa en esa tecnología.

2.3.1. Reconocimiento de voz

El objetivo del reconocimiento automático de voz es, entender computacionalmente el lenguaje hablado mediante la construcción de un sistema que mapee una señal acústica a una cadena de palabras [Cook02]. Esta tarea es fundamentalmente de clasificación de patrones [Martin]. Se toma un patrón de entrada, que en este caso es la señal de voz y, se clasifica dentro del conjunto de realizaciones fonéticas o alofónicas establecido [Oropeza06].

El funcionamiento del sistema comprende dos etapas [Cook02]: una de entrenamiento y una de reconocimiento.

Durante la etapa de entrenamiento, el sistema almacena las propiedades de un conjunto de palabras por ejemplo sus fonemas, alófonos, duración, etc. [Martin] Durante la etapa de reconocimiento, el sistema identifica la pronunciación que con mayor probabilidad se parece a las pronunciaciones almacenadas en la memoria del sistema [Martin].

El esquema de reconocimiento de voz consta de un módulo de adquisición de datos, un módulo de extracción de propiedades o características, un módulo de cuantificación y un módulo de reconocimiento [CMU07].

El *módulo de adquisición de datos* convierte la señal sonora a eléctrica y después a una secuencia de valores numéricos, es decir, hace la conversión analógica a digital [CMU07].

El *módulo de extracción de propiedades o características* obtiene datos de la señal como son energía espectral, tono, formantes, etc., correspondiente a una pronunciación [CMU07]. El proceso consiste en dividir la secuencia de valores, obtenida en el módulo anterior, en segmentos correspondientes a una duración de entre 10 y 35 milisegundos debido a que se ha determinado que la duración de tonos de los sonidos del habla está en ese rango [Cook02]. La salida de este módulo consiste en una secuencia de vectores de características de los segmentos [Martin].

El *módulo de cuantificación* identifica los distintos sonidos que están presentes en la pronunciación [CMU07]. La salida de este módulo es una secuencia de valores, donde cada valor representa el sonido con el que está asociado un vector de características.

Un error de cuantificación ocurre cuando el número de bits es reducido de la señal original, siendo el caso más obvio cuando una señal analógica es convertida a digital en un convertidor AD [CMU07]. De ser una onda sinoidal perfecta, al ser cuantificada se convierte en una señal a pasos, que en vez de ser una onda con curvas perfectas, tiene forma de escalera. Esto provoca que el sonido sea malo debido a la pérdida de resolución.

Añadir un poco de ruido blanco a una señal suaviza su forma escalonada creando un sonido más natural. Sorprendentemente, este proceso también amplía el rango dinámico de tal manera que señales mucho más pequeñas que el más pequeño paso digital pueden ser representadas, lo que hace que los sonidos tengan un decaimiento mucho más natural después de haberse aplicado el ruido.

Además de reducir la distorsión de los componentes de bajo nivel, el ruido permite que el usuario escuche componentes por debajo del nivel del bit menos significativo a través de la adición de una señal no muy grande para evitar una distorsión de bit. El ruido añadido empuja la distorsión sobre el punto de transición a cambio de una cantidad estadísticamente proporcionada a su nivel de amplitud real. El oído y cerebro humano, expertos en la separación de tal señal del ruido de fondo, hacen el resto. Tal como es posible seguir una conversación en un espacio ruidoso, es posible obtener una señal débil del ruido.

Finalmente, el *módulo de reconocimiento* identifica a una pronunciación dada y la clasifica dentro de sus modelos como un sonido conocido, parecido a uno conocido o bien, desconocido [CMU07].

Utilizando el esquema antes descrito, el software de reconocimiento de voz es utilizado en un amplio rango de aplicaciones, desde sistemas de telefonía comercial (dominado por ScanSoft en conjunto con Nuance) hasta productos personales [Jiang04] por ejemplo: en cuanto al software libre se pueden mencionar, *ConsoleVoiceControl* [LinuxOnline07]; *ISIP* (Institute for Signal and Information Processing) [CMU07]; *CMU SPHINX* [CMU07]; y *XVoice* [Creemer07].

En cuanto al software comercial, Babel Technologies ofrece un *Software Development Kit* (SDK) de Linux llamado *Babear* [Acapela07]; *Nuance* [Nuance07]; *Speech Engine* [LumenVox07]; y *Speech Platform* [TechNet07].

Capítulo 3

Desarrollo y experimentación

El objetivo principal del reconocimiento de voz es proporcionar una interacción hombre-máquina apropiada a través de órdenes habladas [Cook02]. Esencialmente trabaja con la discriminación entre conjuntos discretos de unidades lingüísticas; por ejemplo enunciados, palabras o unidades fonéticas [CMU07]. Para ejecutar aplicaciones de reconocimiento de voz, se requiere de varios archivos. Estos se clasifican en tres categorías [Miller]:

1. El modelo acústico.
2. El modelo de lenguaje.
3. El diccionario de pronunciación.

A continuación se describirán cada una de ellas explicando cómo fueron involucradas durante la realización de este trabajo. Posteriormente, se indica el orden en el cuál es necesario utilizarlas dentro de CMUSphinx3 para indicar la decodificación y obtener un archivo que muestre las palabras reconocidas denominado archivo hipotético. EL archivo hipotético es utilizado no sólo para observar el desempeño del reconocedor sino también para obtener el comando que debe obedecer el robot. Para saber si el archivo hipotético contiene alguna de las expresiones propuestas dentro del modelo de lenguaje, se hace uso del lenguaje de programación PERL para encontrar el patrón buscado. Para indicar al usuario el resultado, se hace uso del sintetizador de voz artificial Festival por medio del cual se puede escuchar un mensaje que indica el éxito o fracaso de la búsqueda. Finalmente haciendo uso de un condicional switch desarrollado en PERL, la acción correspondiente es llevada a cabo produciendo el movimiento deseado en el robot real o simulado.

3.1. Modelo Acústico

Los modelos acústicos se pueden ver como filtros que contienen la variabilidad acústica de una lengua [Miller]. Dependiendo de las expectativas de los modelos y dada una señal de entrada se obtiene la hipótesis de lo que el hablante ha dicho. EL modelo acústico incluye [Miller]: el análisis acústico, en el cual se caracteriza a la señal de entrada en una secuencia de vectores acústicos; los modelos acústicos para las unidades que forman las palabras (fonemas modelados dependientes del contexto=; y el diccionario de pronunciación en el cual se define la descomposición de las palabras en unidades más pequeñas que corresponden a las unidades dentro del alfabeto fonético definido. De acuerdo a la sección de documentos de CMUSphinx3.4 y al objetivo de este trabajo se decidió utilizar los siguientes archivos para contar con el modelo acústico necesario para llevar a cabo el reconocimiento de voz:

1. un modelo de definición (mdef), el cual define el conjunto de sonidos, el mapeo de cada estado de modelo oculto de Markov a una senoide, y el mapeo de cada modelo oculto de Markov a una matriz de transición de estado.

2. Un arhivo de medianas gausianas

3. Un archivo de varianzas gausianas

4. Un archivo de los pesos de las mezclas gausianas para todas las senoides en el modelo.

5. Un archivo de matriz de transición de estado que contiene todas las topologías de transición de estado de modelo oculto de Markov y sus probabilidades de transición en el modelo.

Los archivos antes descritos forman parte del proyecto DIME del IIMAS de la UNAM y fueron utilizados en la realización de este trabajo con el consentimiento de esa institución. La identificación de una palabra es difícil debido a que la entrada es ruidosa o ambigua. El profesionalismo con el que fueron producidos los modelos acústicos utilizados, fomentó el buen reconocimiento de las expresiones producidas por los usuarios durante los experimentos realizados. Para capturar las restricciones necesarias en la combinación de palabras para generar las frases posibles se hizo uso de un modelo de lenguaje.

3.2. Modelo de lenguaje

De acuerdo a Dam Jurafski, el modelo de lenguaje contiene las propiedades lingüísticas del lenguaje y brinda la probabilidad a priori de una secuencia de palabras. Adivinar la siguiente palabra o lo que se llama predicción de la palabra, es una subtarea esencial en el reconocimiento de voz [Cook02]. La importancia de dichos modelos es que permiten recudir el espacio de búsqueda sobre el que se mueven los reconocedores y también pueden utilizarse para corregir a posteriori la salida de los mismos [Miller].

La tabla muestra una parte del modelo de lenguaje creado a partir de las frases utilizadas para dirigir el robot de la simulación proporcionada por la plataforma de desarrollo abierta Player/Stage y el robot del laboratorio de sistemas computacionales.

Dado que un archivo de texto grande puede ser muy lento al ser cargado a memoria, es necesario un archivo binario que aguilice el proceso. El archivo binario del modelo de lenguaje (dmp) es más o menos una imagen de disco de la estructura de datos del modelo de lenguaje construido en memoria. Puede ser creado utilizando la herramienta lm3g2dmp que es parte de la distribución de Sphinx [CMU07]. Una vez instalada la herramienta, se utilizan los comandos presentados en la figura para crearlo.

Los objetivos de este proceso son mapear los trigramas al vocabulario establecido y especificar la estrategia para el suavizado, es decir, la técnica en la cual se cambian las probabilidades para eventos con valor cero o muy pequeño.

Debido a las limitaciones computacionales y a la naturaleza abierta del lenguaje donde hay una infinidad de palabras posibles, en el modelo de lenguaje se asume una independencia tal que, cada palabra sólo depende de las últimas n palabras anteriores, convirtiéndose en un buen modelo de Markov [Brown]

Este trabajo utiliza un modelo de lenguaje trigrama matemáticamente definido como P(palabra3 — palabra1, palabra2) o la probabilidad condicional de que la palabra3 siga inmediatamente a la secuencia de la palabra1 palabra2 en el lenguaje.

Por ejemplo, en la frase *la madrastra era una auténtica bruja*, los trigramas serían: *la madrastra era*, *madrastra era una*, *era una auténtica* y u*na auténtica bruja*. Para secuencias de caracteres los trigramas que podrían generarse a partir de la frase *buenos días*, serían *bue*, *uen*, *eno*, *nos*, *s d*, *d* , etc [CMU07]

La probabilidad del modelo de lenguaje de una oración completa es el producto de las probabilidades individuales. Concretamente, un modelo de ngrama predice un valor

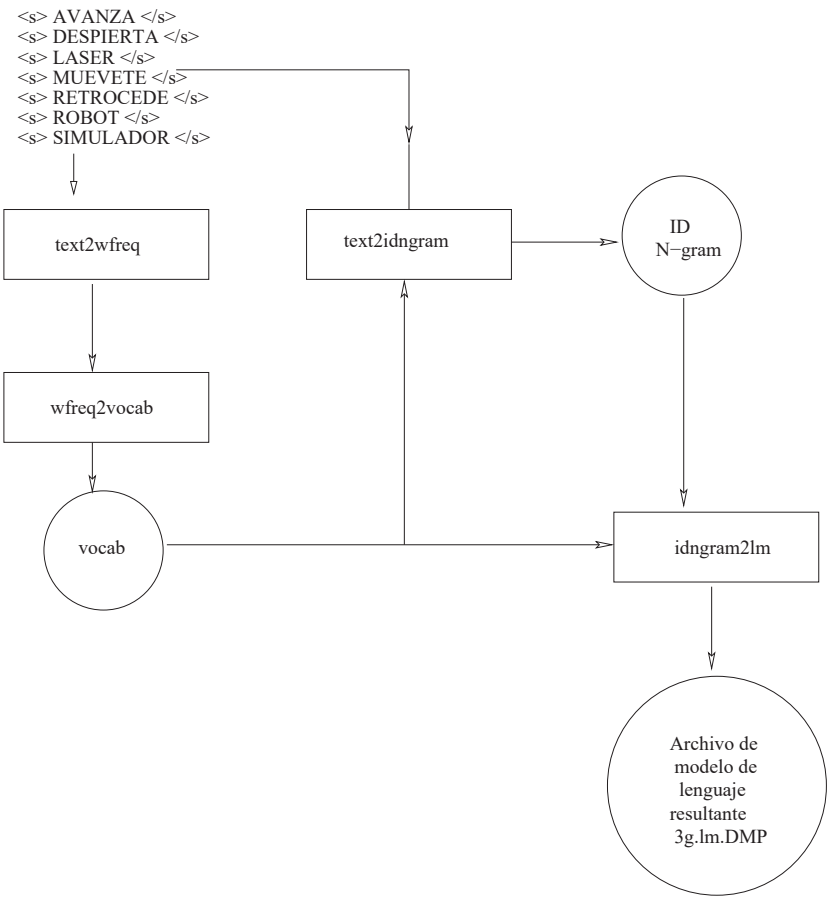

Figura 3.1: Diagrama representativo de la creación de un archivo binario de modelo de lenguaje

a partir de los anteriores. Por ejemplo, la probabilidad de la oraciíon "HOY ES LUNES.^{es}: P(HOY — ¡s ¿)* P(ES — ¡s ¿, HOY)* P(LUNES — HOY, ES)* P(¡s ¿— ES, LUNES)Sin embargo, esta aproximación presenta un problema. La probabilidad de muchas frases perfectamente aceptables será cero simplemente porque no fueron incluidas en el modelo de lenguaje. Puesto que la probabilidad de una frase se calcula como el producto de las probabilidades de sus ngramas las posibilidades ceros se propagarán resultando en una mala estimación de la probabilidad (probabilidad cero) de aquellas frases que contengan ngramas no encontrados en el corpus de entrenamiento. Para evitar esto, es necesario utilizar mejores estimadores. Los métodos por los que se obtienen estos estimadores suelen recibir el nombre

de métodos de descuento y el proceso mediante el cual se realiza el descuento suele denominarse suavizado. El algoritmo de suavizado utilizado para crear el archivo binario de modelo de lenguaje es el algoritmo Witten-Bell Discounting basado en un evento de frecuencia cero considerado como un evento que aún no ha sucedido.

Sin embargo, el tamaño del vocabulario es un factor importante que se debe tomar en cuenta para efectuar el reconocimiento. A medida que crece el vocabulario, aumenta la dificultad del reconocimiento de palabras debido a que se pueden parecer unas con otras y por lo tanto confundir. Además, el espacio de búsqueda es mayor y encontrar la palabra con mayor probabilidad resulta complicado [Miller]. Finalmente, las palabras utilizadas por el modelo de lenguaje deben estar contenidas en un diccionario de pronunciación.

3.3. Diccionario de pronunciación

El diccionario de pronunciación contiene la pronunciación de todas las palabras de interés para el decodificador. Como la mayoría de los sistemas de reconocimiento modernos, Sphinx3 utiliza unidades fonéticas para construir la pronunciación de las palabras. Actualmente, la pronunciación del diccionario se encuentra hecho a mano. Los diccionarios de pronunciación se construyen a partir de grandes corpus [CMU07]. Puede ser el mismo corpus utilizado en el entrenamiento de los modelos acústicos o puede ser otro aún mayor. El vocabulario del diccionario corresponde al conjunto de palabras asociadas a una pronunciación que se requieren reconocer [Macias].

Las palabras se encuentran divididas en fonemas, los cuales son la unidad básica del habla y en conjunto determinan los sonidos con los que se pueden construir las palabras de cualquier lenguaje [Cuetara]. Acústicamente, un fonema es un sonido que se distingue por un patrón característico y las diferentes pronunciaciones que puede llegar a tener se conocen como alófonos [Cuetara]. Su clasificación depende de la forma de articulación de los sonidos [Cuetara]. Si los órganos cierran total y momentáneamente la salida del aire los sonidos son oclusivos; cuando la salida de aire provoca fricción al atravesar el paso o estrechamiento formado por los órganos de la cavidad bocal los sonidos son fricativos; si hay un primer momento de cierre seguido de otro de fricción o roce los sonidos son africados; cuando el aire sale tanto por las fosas nasales como por la boca y además es de cierre (como los oclusivos) el sonido es nasal; un sonido vibrante se caracteriza porque la lengua toca una o repetidas veces los alvéolos; finalmente si el aire se escapa por los lados de la lengua

es un sonido lateral. El punto de articulación es el lugar donde toman contacto los órganos que intervienen en la producción del sonido [Cuetara]. Si para producir un sonido entran en contacto los labios se crean sonidos labiales; si se produce un contacto del labio inferior con los incisivos superiores entonces son labiodentales; si se utilizan los dientes superiores y la lengua son dentales; si intervienen la lengua y los alvéolos los sonidos son alveolares; si se producen en la zona del paladar duro los sonidos son palatales; finalmente si interviene la lengua y el velo del paladar el sonido es velar. La sonoridad se obtiene de las cuerdas vocales, si las cuerdas vocales no vibran los sonidos se llaman sordos, de lo contrario son sonoros.

La Facultad de Letras Hispánicas en conjunto con el Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas de la Universidad Nacional Autónoma de México identificó 22 fonemas (17 sonidos consonantes y 5 vocales) en el español mexicano para la creación de sus modelos acústicos. La imagen muestra la clasificación propuesta y la forma en la fue empleada para crear el diccionario de pronunciación necesario durante el proceso de decodificación.

También es necesario incluir símbolos que representen silencio como lo muestra la imagen en un diccionario complementario denominado diccionario de relleno.

El vocabulario activo durante la decodificación de una expresión es el conjunto de palabras que se encuentren presentes en el diccionario de pronunciación y en el modelo de lenguaje [Miller]. EL decodificador es incapaz de reconocer una palabra fuera del vocabulario activo y debe ser inicializado con un diccionario de pronunciación que defina todas las palabras de interés para la aplicación y la pronunciación fonética de cada palabra [Miller]. A continuación se describirá el proceso que debe ser llevado a cabo para realizar la decodificación.

3.4. Decodificación

Debido a la simplicidad y disponibilidad de uso, la construcción del reconocedor de voz se hizo a través de la herramienta CMU Sphinx3.4 y de los modelos acústicos DIMEx30 de etiquetación manual de la UNAM. Como muestra la imagen la decodificación se realizó mediante un ejecutable simple, al que se le pasaron un grupo de entradas: los modelos acústicos entrenados (DIMEx30), el modelo de lenguaje binario, un diccionario de pronunciación y un diccionario de relleno.

CMUSphinx3 es un motor de descifrado desarrollado en la Universidad de Carnegie Mellon en la ciudad de Pittsburg Pensilvania, uno de los más destacados centros de investigación superior de los Estados Unidos en el área de informática y robótica [CMU07]. CMUSphinx puede ser utilizado para construir aplicaciones de vocabulario pequeño, mediano o grande. Las palabras reconocidas pueden servir de entrada a otros sistemas que los requieran para realizar otra acción.

El motor de reconocimiento es el núcleo del decodificador de voz y opera en segmentos de longitud finita de voz o expresiones, una expresión a la vez [Miller]. El decodificador trata de detectar zonas de voz en las que existe un desacople importante entre los modelos acústicos y la voz pronunciada, o zonas en las que aparecen varias alternativas con gran confusión acústica entre ellas.

Algunas de las causas que dificultaron el reconocimiento automático del habla durante los experimentos efectuados son: las variaciones de producción, es decir, falta de cuidado al pronunciar algunas palabras, las variaciones fonéticas, la duración de una palabra e incluso de los sonidos puede cambiar; y los ruidos e interferencias. El ruido de fondo puede influir dramáticamente en el nivel de reconocimiento. Por ejemplo, se observó que las muestras tomadas en el Posgrado de Ingeniería Eléctrica fueron inferiores a las tomadas en el edificio de idiomas de la Universidad Michoacana. Se llegó a la conclusión de que este resultado se debe principalmente al ruido producido por el locutor, así como su estado de ánimo y estrés.

Cuando se han decodifica la expresión producida por el locutor es necesario conocer si contiene alguna de las órdenes establecidas en el modelo de lenguaje para dirigir el robot. Para ello se creo una aplicación en PERL la cual será descrita a continuación.

3.5. Aplicación Comando/Respuesta

3.6. Experimentación

El rendimiento de un decodificador de voz depende normalmente de un equilibrio entre la precisión en el reconocimiento y los entornos creados para minimizar el uso de recursos y maximizar la velocidad de aquellos necesarios para acomodar un tamaño y complejidad concretos de los modelos acústicos, modelos de lenguaje y diccionarios.

Las cuestiones más importantes sobre el uso eficiente de un decodificador son

aquellas relacionadas con la velocidad de decodificación, uso de memoria y consumo de potencia. Dado que en este trabajo de investigación se utilizó una computadora fija el consumo de potencia no fue una cuestión crucial.

Durante el proceso de experimentación, el reconocedor se probó realizando experimentos utilizando un micrófono para PC GE modelo HO98950 para introducir la señal al sistema. Sólo fue necesario contar con el diccionario de pronunciación y el modelo de lenguaje.

Se prestó cuidado en cuanto al diccionario de pronunciación ya que el sistema sólo puede reconocer palabras representadas explícitamente en el diccionario. Si se utilizan palabras fuera de él, el reconocedor encuentra la mejor aproximación que muchas veces resulta en una frase incoherente.

Para la prueba se seleccionaron 18 personas todas ellas procedentes del Posgrado de Ingeniería Eléctrica con un promedio de edad de 33 años.

No se utilizó ningún dispositivo especial ni lugar especial; tampoco se especificaron características de lectura a los hablantes. La intención era realizar una prueba en condiciones normales.

Para el proceso de evaluación de las muestras tomadas se utilizó el programa sclite de NIST (Nacional Institute of Standards and Technology) una herramienta para evaluar la salida de los sistemas de reconocimiento de voz y la fórmula de precisión de palabra $pp = (100 * (1 - (\frac{insertadas+eliminadas+sustituidas}{total})))$.

El programa compara la salida del texto hipotético (HYP) del reconocedor de voz con el correcto o texto de referencia (REF). Después de comparar REF con HYP (un proceso llamado alineación), se obtienen estadísticas durante el proceso de evaluación y una variedad de reportes pueden ser producidos para mostrar el funcionamiento del sistema de reconocimiento.

Finalmente se utilizó una función creada en perl para resumir los resultados obtenidos de cada una de las personas participantes.

3.6.1. Primer experimento: ocho comandos para ordenar al robot

En el primer experimento se seleccionaron 18 personas que leyeron los 8 comandos propuestos para dirigir al robot.

Palabras	correctas	sustituidas	eliminadas	insertadas	precisión de palabra
Avanza	57.8	36.8	0	10.5	52.6
Despierta	68.4	21.0	5.2	5.2	68.4
Dirígete a la derecha	80.2	5.2	9.2	6.5	78.9
Dirígete a la izquierda	82.8	5.2	6.5	5.2	82.8
Gira a la derecha	44.7	30.2	19.7	5.2	44.7
Gira a la izquierda	39.4	26.3	28.9	0	44.7
Muévete	74.5	9.6	10.5	0	79.8
Retrocede	78.9	15.7	0	21.0	63.1
Resultados Globales	70.5	16.4	13.0	4.2	66.3

Cuadro 3.1: Resultados del primer experimento

Los resultados obtenidos demuestran que es factible utilizar comandos cortos para dirigir el robot mediante comandos hablados.

3.6.2. Segundo experimento: once frases obtenidas de la WEB

En el segundo experimento se seleccionaron 18 personas que leyeron las 11 frases obtenidas de la WEB.

Un problema observado durante este experimento es que las personas deben hablar fuerte, claro y sin reírse y el volumen de audio debe ser adecuado. Se debe tomar en cuenta el tipo de micrófono, la calidad del canal de transmisión, el eco, el ruido, la distancia a la que la persona sostiene el micrófono y su dicción.

3.6.3. Reporte del tercer experimento

En el tercer experimento se probó la interfaz de comando/respuesta. La interfaz fue creada en el lenguaje Perl y permite traducir las palabras reconocidas en comandos ejecutables capaces de generar movimiento en el robot simulado en la plataforma de desarrollo abierta Player/Stage. Se introdujeron comandos registrados y no registrados obteniéndose en cada uno de ellos las respuestas esperadas. Es decir, los comandos mencionados en el primer experimento generaron el movimiento adecuado. Para el resto de la palabras o frases posibles el robot indica a través del sintetizador Festival que el comando propuesto no es válido.

La interfaz fue probada en repetidas ocasiones para evaluar su eficiencia. Adjunto a la tesis escrita se presenta un breve video que demuestra los resultados obtenidos en este experimento,

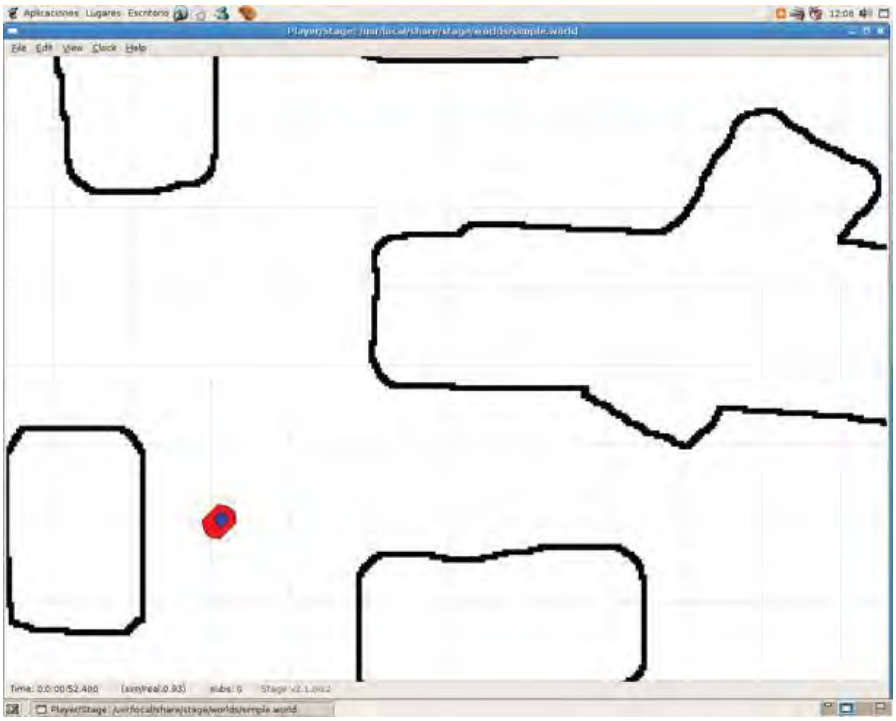

Figura 3.2: Robot simulado en la plataforma abierta Player/Stage

Palabras	correctas	sustituidas	eliminadas	insertadas	precisión de palabra
Solicita información	83.0	16.9	0	0.8	82.2
Acompañados es más fácil	74.8	24.1	0.9	1.6	73.2
Como un acordeón	75.2	21.5	3.2	0	75.2
En la colmena hay una abeja que no quiere trabajar	43.8	41.2	14.8	0	43.8
Ensayos acerca de múltiples temas	58.7	40.6	0.6	0.6	58.0
Especialistas abordaran equidad y desarrollo	43.5	54.0	2.4	0	43.5
Formación académica y asesoría tecnológica	67.7	31.6	0.6	0	67.7
La pobreza es abatida	72.5	27.4	0	0	72.5
Los abanicos nunca pasan de moda	57.4	17.4	25.1	0.3	57.0
Los secretos del abuelo	67.7	30.6	1.6	62.9	4.8
Se acabó una etapa para la selección de mi país	37.0	27.4	35.48	2.4	34.6
Resultados Globales	62.5	28.7	8.7	2.7	59.7

Cuadro 3.2: Resultados del segundo experimento

Capítulo 4

Conclusiones

Durante el proceso de construcción de un sistema reconocedor de voz el tener las aplicaciones para generarlo no es suficiente para lograr resultados; se debe conocer cómo utilizarlas de forma correcta y adicionalmente se debe contar con el hardware adecuado para probar el sistema creado.

Después de haber aprendido a usar el paquete CMUSphinx3 durante una estancia en la UNAM, se crearon un diccionario de pronunciación, un modelo de lenguaje de 3 gramas, una función para obtener las estadísticas globales de los experimentos realizados, una interfaz comando/respuesta para manejar el robot simulado en la plataforma abierta Player/Stage a partir de voz y 5 funciones de movimiento para el robot simulado.

Posteriormente se llevaron a cabo las pruebas de reconocimiento, obteniéndose, según la página tutorial de Sphinx, resultados satisfactorios: durante el primer experimento se obtuvo un porcentaje de 70 % en palabras correctamente reconocidas y en el segundo 62 %.

Los sistemas de reconocimiento de voz no toman en cuenta el contexto cultural, no conocen el significado de lo que el hablante dice y sólo se limitan a clasificar las frases desde un punto de vista acústico.

Además, el reconocedor de voz esta limitado a un vocabulario. Esto restringe el reconocimiento; sin embargo es posible cambiar el modelo de lenguaje y el diccionario de pronunciación por unos más amplios para futuros experimentos.

Desarrollar un software para construir este tipo de sistemas requiere de una enorme cantidad de conocimientos matemáticos y de programación.

El grupo de personas que crearon paquetes como Sphinx han hecho una labor de

años de investigación en el área y es por eso que esos paquetes son considerados de lo mejor que existe en el mercado.

La tesis mostró el proceso de construcción del reconocedor y cómo utilizar uno de los avances que hasta el momento presenta el campo del procesamiento del lenguaje natural.

4.1. Trabajos futuros

Como trabajos futuros es necesaria la creación de un modelo acústico propio que en conjunto con la Facultad de Letras Hispánicas de la Universidad Michoacana proporcione un diseño que identifique el acento y las expresiones de los estudiantes de la Universidad Michoacana de San Nicolás de Hidalgo.

Además, varias universidades del país se encuentran trabajando en la creación de sistemas multimodales inteligentes y gramáticas computacionales del español que el Posgrado en Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo puede también desarrollar en conjunto con el área de inteligencia artificial.

Finalmente, se puede pretender desarrollar un reconocedor de voz propio que proponga mejoras como diccionarios automáticos o múltiples modelos de lenguaje.

Bibliografía

[Acapela07] Acapela, G. Acapela group speech solutions give you the say!, dec. 2007.
URL http://www.acapela-group.com/

[Alexander05] Alexander, D. y Rodríguez, F. Estado del arte en el reconocimiento automático de la voz, apr. 2005.
URL http://es.geocities.com/deibywolf/Estado del arte.pdf

[ARIA02] ARIA, jun. 2002.
URL www-users.cs.umn.edu/ stergios/classes/csci5551/ Aria-Reference.pdf

[Bermejo03] Bermejo, S., ed. *Introducción a los robots basados en comportamiento*. Universidad Politécnica de Cataluña, Barcelona, 2003.

[Black] Black, A. y Lenzo, K. Festvox.
URL http://festvox.org/festival/

[Blank03] Blank, D., Kumar, D., Meeden, L., y Yanco, H. Pyro: A python-based versatile programming environment for teaching robotics. *Journal on Educational Reosurces in Computing*, 3(4):1–15, sep. 2003.

[Brown] Brown, M., Kellner, A., y Raggett, D. Stochastic language models (n-gram) specification.
URL http://www.w3.org/TR/ngram-spec/

[cañas05] cañas, J. y Matellan, V. Integrating behaviors for mobile robots: an ethological approach. Inf. Téc. TR-GSYC-2004-7, Universidad Rey Juan Carlos, jul. 2005.

[Cañas06] Cañas, J., Matellán, V., y Montúfar, R. Programación de robots móviles. *RIAI: Revista Iberoamericana de Automática e Informática Industrial*, 3(2):99–110, apr. 2006.

[CMU07] CarnegieMellon, U. The cmu sphinx group open source speech recognition engines, dec. 2007.
URL `http://cmusphinx.sourceforge.net/html/cmusphinx.php/`

[Colomina07] Colomina, O. y Cazorla, M., apr. 2007.
URL `http://cursos.rvg.ua.es/file.php/14/Practicas/intro-pse1.pdf`

[Cook02] Cook, S. Speech recognition howto, apr. 2002.
URL `www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Speech-Recognition-HOWTO.pdf`

[Creemer07] Creemer, D., Doris, T., y Craft, B. Xvoice, dec. 2007.
URL `http://xvoice.sourceforge.net/`

[Cuetara] Cuetara, J. Fonética de la ciudad de méxico. aportaciones desde las tecnologías del habla.
URL `www.filos.unam.mx/LICENCIATURA/VOX/Lecturas.htm`

[Fernandez05] Fernandez, J. Linux para todos - todos necesitamos ayuda, apr. 2005.
URL `www.linux-magazine.es/issue/06/Educacion.pdf`

[Firby94] Firby, J. Task networks for controlling cotinuous processes. *Proceedings of the 2nd International Conference on AI Planning Systems AIPS94*, 5(2):49–54, jul. 1994.

[Gerkey02] Gerkey, B. y Mataric, M. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, oct. 2002.

[Gerkey03] Gerkey, B., Vaughan, R., y Howard, A., dec. 2003.
URL `http://playerstage.sourceforge.net/doc/Stage-manual-1.3.3.pdf`

[Gerkey04] Gerkey, B., Vaughan, R., y Howard, A., jun. 2004.
URL `http://playerstage.sourceforge.net/doc/Player-manual-1.5.pdf`

[Gerkey05] Gerkey, B., Vaughan, R., y Howard, A., sep. 2005.
URL `http://playerstage.sourceforge.net/doc/Player-cvs/player/`

[Goldsmith] Goldsmith, J. Ngram models and the sparsity problem.
URL `humanities.uchicago.edu/faculty/goldsmith/Industrial/Ngrams.ppt`

[Howard03] Howard, A., Gerkey, B., y Vaughan, R. The player/stage project: Tools for multi-robot and distributed sensor systems. *Proceedings of the 2003 IEEE/RSJ International Conference on Advanced Robotics*, 3(3):317–323, jul. 2003.

[Jiang04] Jiang, H. Confidence measures for speech recognition: A survey. *Journal of Computer Science and Technology.*, 20(6):885–894, dec. 2004.

[Jurafsky] Jurafsky, D. y Martin, J. Speech synthesis.
URL `http://www.cs.tut.fi/kurssit/SGN-4010/puhesynteesi_en.pdf`

[Kordic05] Kordic, V., Lazinica, A., y Merdan, M. *In Cutting Edge Robotics*. Pro Literature Verlag / ARS, 2ª ed$^{\underline{\text{ón}}}$., 2005.

[Kramer06] Kramer, J. y Scheutz, M. Development environments for autonomous mobile robots: A survey. *International Journal of Intelligent Automation and Soft Computing, Special Issue on Global Look at Robotics Education*, 22(2):101–132, feb. 2006.

[LinuxOnline07] LinuxOnline. Application: Cvoicecontrol, dec. 2007.
URL `http://www.linux.org/apps/AppId 2379.html`

[Lozano01] Lozano, M., Iborra, I., y Gallardo, D. Entorno java para simulación y control de robots móviles. *Actas de la IX Conferencia de la Asociación Española Para la Inteligencia Artificial CAEPIA 2001 Gijón*, 35(9):1249–1258, mar. 2001.

[LumenVox07] LumenVox. Lumenvox speech understood, dec. 2007.
URL `http://www.lumenvox.com/products/speech engine/`

[Macias] Macias, J. Arquitecturas y métodos en sistemas de reconocimiento automático de habla de gran vocabulario.
URL `http://lorien.die.upm.es/macias/tesis.pdf`

[Martin] Martin, J. y Jurafsky, D. Speech and language processing.
URL `http://www.cs.colorado.edu/ martin/SLP/Updates/9.pdf`

[Miller] Miller, J. Reconocimiento de voz con sistemas basados en hmm: Sphinx3.
URL `mit.ocw.universia.net/.../6563BA98-DD44-4DB1-B28B-9B1CCD0EF7C1/`

[Missionlab06] Missionlab, jul. 2006.
URL `http://www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab/`

[Montemerlo03] Montemerlo, M., Roy, N., y Thrun, S. Perspectives on standardization in mobile navigation carmen toolkit. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3(22):2436–2441, 2003.

[Nuance07] Nuance. The leading supplier of speech recognition, dec. 2007.
URL `http://www.nuance.com/`

[Oropeza06] Oropeza, L. y Suárez, S. Ialgoritmos y métodos para el reconocimiento de voz en español mediante sílabas. *Computación y sistemas*, 9(3):270–286, dec. 2006.

[Renals] Renals, S. y King, S. The festival speech synthesis system.
URL `www.cstr.ed.ac.uk/projects/festival/`

[Rico05] Rico, F. Open-r un enfoque práctico, jan. 2005.
URL `www.uji.es/ badia/e69/docs/tutorial-openr.pdf`

[Taylor] Taylor, P. y Caley, R. Festival at cmu.
URL `http://www.speech.cs.cmu.edu/festival/index.html`

[TechNet07] TechNet, M. Speech platform, dec. 2007.
URL `http://technet.microsoft.com/en-us/library/bb684798.aspx`

[Utz02] Utz, H., Sablatnög, S., Enderle, S., y Kraetzschmar, G. Miro - middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18(4):493–497, 2002.

[Vaughan03a] Vaughan, R., Gerkey, B., y Howard, A. On device abstractions for portable, reusable robot code. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3(3):2121–2427, oct. 2003.

[Vaughan03b] Vaughan, R., Gerkey, B., Stoy, K., Howard, A., Mataric, M., y Sukhatme, G. Most valuable player: A robot device server for distributed control. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3(3):1226–1231, oct. 2003.

[Vaughan06] Vaughan, R. y Gerkey, B., jun. 2006.
URL `http://www.ai.sri.com/pubs/files/1221.pdf`

[Woo03] Woo, E., MacDonald, B., y Trépanier, F. Distributed mobile robot application infraestructure. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2(2):1475–1480, oct. 2003.

Apéndice A

CMUSphinx

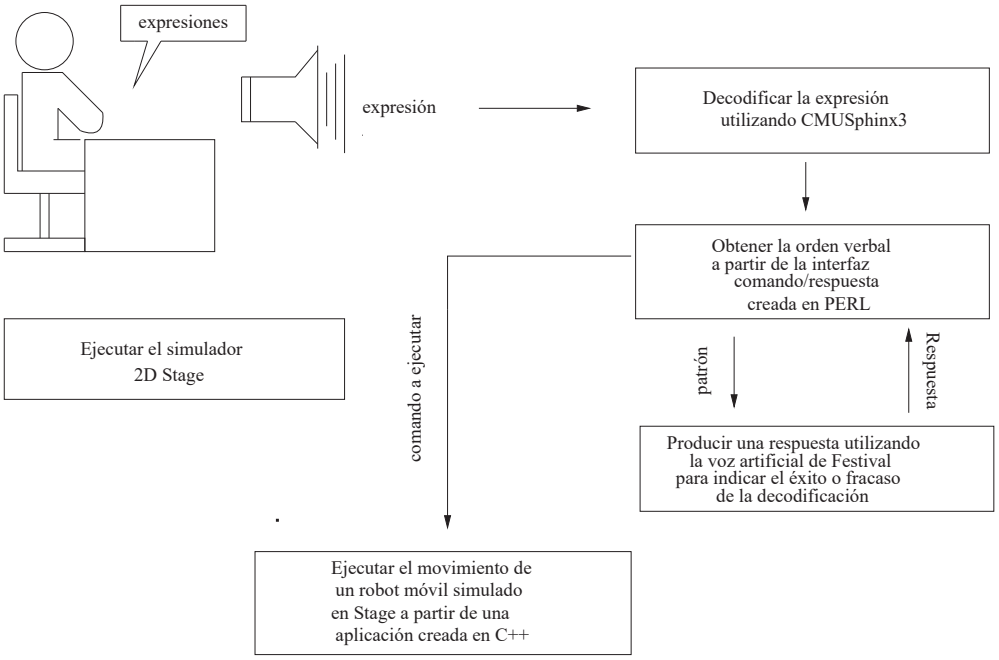

Figura A.1: Diagrama representativo del reconocimiento de voz

CMUSphinx es un motor de descifrado desarrollado en la Universidad de Carnegie Mellon en la ciudad de Pittsburg Pensilvania, uno de los más destacados centros de investigación superior de los Estados Unidos en el área de informática y robótica [CMU07].

CMUSphinx consta de un conjunto de bibliotecas que incluyen funciones de capa de reconocimiento de voz así como auxiliares de captura de audio de bajo nivel [CMU07]. Las bibliotecas están escritas en C y han sido compiladas en varias plataformas UNIX (Linux, DEC Alpha, Sun Sparc, HPs) y Pentium/PentiumPro, PCs ejecutadas con WindowsXp, WindowsNT o Windows95.

CMUSphinx puede ser utilizado para construir aplicaciones de vocabulario pequeñas, medianas o grandes. Sus características principales son [CMU07]: descifrado de voz continua; locutor independiente; capacidad para proporcionar el mejor o varios reconocimientos alternativos; modelos continuos y semicontinuos de densidad acústica; bigramas, trigramas o modelos de lenguaje de gramática finita; alineamiento forzado y modos de reconocimiento alófono.

A.1. Entrenador CMUSphinx

El entrenador de CMUSphinx consta de un conjunto de programas, cada uno responsable de una tarea definida, y un conjunto de scripts que organizan el orden en el que los programas son llamados [Miller].

El entrenador aprende las características de los modelos de las unidades de sonido utilizando un conjunto de señales de voz muestra denominado base de datos de entrenamiento [CMU07].

El entrenador también requiere saber de qué unidades de sonido debe aprender las características y la secuencia en la que ocurren en cada señal de voz en la base de datos de entrenamiento [Miller]. Esta información es proporcionada al entrenador a través de un archivo de trascripción, en el que la secuencia de las palabras y los sonidos no sonoros son escritos exactamente como ocurren en la señal de voz, seguidos de una etiqueta que puede ser utilizada para asociar esa secuencia con la señal de voz correspondiente [CMU07]. Posteriormente, el entrenador busca en el diccionario de palabras derivar la secuencia de unidades de sonido asociados con cada señal.

En cuanto a los modelos de lenguaje o gramáticas, CMUSphinx acepta de estado finito (FSG) y N-gramas [CMU07]. Se pueden cargar múltiples modelos de lenguaje, N-gramas o FSG en el decodificador durante la inicialización o en tiempo de ejecución [Miller]. La aplicación puede cambiar modelos de lenguaje entre expresiones. Sin embargo, sólo un modelo de lenguaje puede estar activo para una expresión dada [CMU07].

El vocabulario activo durante la decodificación de una expresión es el conjunto de palabras que se encuentren presentes en el diccionario de pronunciación y en el modelo de lenguaje activo [Miller]. El decodificador es incapaz de reconocer cualquier palabra fuera del vocabulario activo.

A.2. Decodificador CMUSphinx

El motor de reconocimiento es el núcleo del decodificador de voz y opera en segmentos de longitud finita de voz o expresiones, una expresión a la vez [Miller]. Una expresión es un pedazo de información de voz procesado entre las llamadas de las funciones del API Sphinx uttproc_begin_utt y la siguiente uttproc_end_utt. El decodificador debe ser inicializado con un diccionario de pronunciación que define todas las palabras de interés para la aplicación y la pronunciación fonética de cada palabra [Miller].

CMUSphinx utiliza modelos ocultos de Markov continuos y semicontinuos que emplean funciones de densidad de probabilidad continuas para representar el espacio de vectores que contienen las características de la señal [Martin]. La principal diferencia entre estos dos tipos de modelos es que, en el caso de los modelos semicontinuos, las funciones de densidad de probabilidad se comparten entre todos los estados de todos los modelos [Cook02]. Por otro lado, en el caso continuo cada estado de cada modelo tiene su propia función de densidad de probabilidad.

Los modelos semicontinuos presentan más ventajas porque el número de funciones de densidad de probabilidad no depende del número de modelos y, por lo tanto, el coste computacional se reduce en cierta medida [Cook02].

Los parámetros que el decodificador utiliza, tratan de detectar zonas de voz en las que existe un desacople importante entre los modelos acústicos y la voz pronunciada, o zonas en las que aparecen varias alternativas con gran confusión acústica entre ellas.

A.3. Evaluación de la salida de CMUSphinx

El programa sclite de NIST (National Institute of Standards and Technology) es una herramienta para evaluar la salida de los sistemas de reconocimiento de voz. Sclite es parte del equipo de herramientas de NIST SCTK. El programa compara la salida del texto hipotético (HYP) del reconocedor de voz con el correcto o texto de referencia (REF). Des-

pués de comparar REF con HYP (un proceso llamado alineación), se obtienen estadísticas durante el proceso de evaluación y una variedad de reportes pueden ser producidos para mostrar el desempeño del sistema de reconocimiento.

Este método se basa en el alineamiento de una señal que ya esta segmentada con otra que no lo está. Lo que se hace es alinear la señal que se desea sintetizar con una señal sintetizada con los mismos fonemas. Como la base de datos utilizada para la síntesis está etiquetada, se conocen las fronteras de los fonemas sintetizados. Estas fronteras se mapean a la vez en la señal grabada obteniendo así la segmentación de la señal deseada.

La alineación de una señal de voz con su correspondiente trascripción fonética es un proceso esencial en la investigación del habla. De ahí que la alineación fonética y el reconocimiento automático de voz (ASR) sean tareas muy relacionadas. En reconocimiento de voz, dada una grabación, es importante identificar lo que se ha dicho sin importar donde inicia cada segmento individual (palabra, sílaba o fonema).

Apéndice B

Creación de un modelo acústico

El modelo acústico es la representación de un sonido basado en datos empíricos [CMU07]. Para este propósito se utilizaron los recursos DIMEx30 del Posgrado en Ciencias Computacionales del Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS) de la Universidad Nacional Autónoma de México (UNAM).

La creación de modelos acústicos es el proceso más largo en la construcción de un reconocedor; éste proceso puede tomar mucho tiempo dependiendo de la cantidad de datos de entrenamiento con que se cuente [Miller].

El proceso de entrenamiento lo llevó a cabo la UNAM en un periodo de cuatro años con el uso de las herramientas SphinxTraining para la creación de los modelos acústicos.

El corpus DIMEx30 es un corpus de habla y fonética para el español de México construido dentro del contexto del proyecto DIME (Diálogos Inteligentes Multimodales en Español) base de datos DIMEx100 T22 de granularidad básica.

El corpus DIMEx100 tiene por objetivo hacer posible la construcción de modelos acústicos y diccionarios de pronunciación para la creación de sistemas computacionales para el reconocimiento del español hablado en México. Los niveles de transcripción correspondientes a los fonemas del Español de México son como sigue:

- Nivel T54 transcripción segmental fina con 54 unidades alofónicas.
- Nivel T44 transcripción segmental media con 44 unidades alofónicas.
- Nivel T22 transcripción segmental básica con las 22 unidades fonéticas.

Los datos necesarios para el proceso de entrenamiento son [Miller]:

- *Diccionario fonético* creado automáticamente de las etiquetas de palabras del corpus DIMEx100 al nivel básico T22.
- *Trascripción de los datos de entrenamiento*
- *Archivos de audio.*
- *Lista de los modelos a entrenar.*

Las transcripciones y los archivos de audio tomados del corpus fueron formateados de acuerdo a las especificaciones de la herramienta utilizada (SphinxTraining).

El procedimiento de creación de los modelos acústicos se realiza en dos fases [CMU07]: *fase de preparación del ambiente y datos* y *fase de entrenamiento.*

El ambiente es creado automáticamente, mediante un script de Sphinx (setup SphinTrain.pl) que genera una serie de directorios [Miller]: *directorios de archivos de configuración* como el *directorio bin* que contiene los archivos ejecutables de la herramienta, el *directorio etc* que contiene los diccionarios, la trascripción del corpus y el archivo de control y, el *directorio scripts_pl* que contiene los módulos de Sphinxtrain; *directorios para depositar los datos necesarios para la creación de los modelos* como el *directorio trees* que contiene los árboles de decisión y el *archivo wav* que contiene los archivos de sonido; y los *directorios que contienen los archivos de salida y los modelos*, tal es el caso de los *directorios bwaccumdir* que almacena los archivos para computar medias o varianzas, el *directorio feat* en donde se depositan los archivos de características, el *directorio logdir* que guarda todos los archivos de error, el *directorio model_architecturre* que contiene los archivos de definición y topología de los modelos y, el *directorio model_parameters* que guarda todos los modelos acústicos.

Una vez creada la estructura de directorios, el siguiente paso es procesar los archivos wav para extraer los vectores de características [Miller].

La extracción de características se realiza desde SphinxTrain mediante el script make_feats.pl en el cual también se especifica la aplicación de dither a todos los archivos de audio [CMU07].

El script computará para cada enunciado una secuencia de coeficientes cepstrales de frecuencia Mel (MFCC) de 13 dimensiones [CMU07]. Los coeficientes MFCC son conocidos actualmente como la mejor parametrización de rasgos simples para el funcionamiento

de un óptimo reconocimiento de voz en sistemas basados en modelos HMM, bajo las mejores condiciones acústicas [CMU07].

Para realizar el entrenamiento, Sphinx requiere los siguientes modelos [Miller]:

- Los archivos de audio convertidos a MFCC o algún otro formato que acepte Sphinx.
- El archivo de trascripción del corpus correspondiente.
- La lista con las unidades acústicas para los modelos que se quieren entrenar.
- El archivo de control que contiene la lista de nombres del conjunto de archivos de características (por ejemplo MFCC) sin extensión.
- El diccionario de pronunciación.
- El diccionario con sonidos de relleno (etiquetas que representan silencio y sonidos que no formen una palabra).

Dentro del directorio etc, se encuentra un archivo de configuración, donde es posible especificar, entre otras cosas [CMU07]: el tipo de los archivos a procesar; el número de iteraciones del algoritmo Baum-Welch; el número de estados del modelo oculto de Markov; el tipo de modelo a generar ya sea discreto, continuo o semicontinuo; el número total de distribuciones de estado compartidas del grupo final de HMM entrenados (los modelos acústicos); la proporción entre la diferencia en probabilidad de la iteración actual y la anterior de Baum-Welch, y la probabilidad total de la iteración anterior; y el número de gaussianas por estado.

El entrenamiento se realiza mediante el script run_all.pl que contiene a su vez otros scripts que ejecutan cada un de los nueve módulos que tienen Sphinxtrain para generar los modelos acústicos [Miller]: módulo de verificación de los archivos de entrenamiento; módulo donde se realiza la cuantificación de vectores; módulo de entrenamiento de los modelos independientes del contexto; módulo donde se atan los modelos y se crean los llamados trifonemas; módulo donde se crean modelos dependientes de contexto; módulo donde se construyen árboles de decisión; módulo donde se construyen los árboles de decisión para cada estado del HMM; módulo donde se podan árboles anteriores; módulo donde se reentrenan los modelos dependientes del contexto hasta determinado número de gaussianas; y módulo donde se realiza el borrado de interpolaciones.

Una vez completado el entrenamiento, los modelos acústicos se localizan en un directorio llamado SPHINX3/model_parameters/RM.cd_continuous_8gau. En este directorio hay cuatro archivos que constituyen sus modelos acústicos [Miller]: means, variantes, transition-matrixes y mixture-weights (medias, varianzas, matrices_transición y pesos_mezcla). Estos son archivos binarios. Sus versiones ASCII también se encuentran en el mismo directorio y los utiliza el decodificador posteriormente. Además de estas, se produce un archivo para estos modelos llamado RM.1000.mdef en el directorio model_architecture. El sistema utiliza este archivo para asociar el grupo de parámetros HMM adecuados con el modelo HMM para cada unidad de sonido que se este modelando.

Apéndice C

Reevaluación de las probabilidades de los N-gramas

Un modelo de lenguaje permite asignar una probabilidad a cada frase posible del idioma seleccionado. Sin embargo, esta aproximación presenta el siguiente problema. La probabilidad de muchas frases perfectamente aceptables será cero simplemente porque no han sido vistas anteriormente.

Para juzgar si una frase es correcta o no, es posible dividir la frase en una serie de componentes. Si los componentes son correctos y están combinados de una forma razonable, la frase es aceptada.

Para una computadora, la manera más fácil de partir una frase en componentes es la de fraccionarla en subcadenas. Una subcadena de n palabras recibe el nombre de n-grama. Si la subcadena está formada por tres palabras, es decir, n es 3, es un trigrama, si n es 2 es un bigrama y si n es 1 es un unigrama o palabra.

Una computadora puede asignar una determinada probabilidad a una frase dependiendo de si los n-gramas son o no razonables. Si está formada por un gran número de ngramas razonables, puede que la frase sea correcta. No necesariamente, pero es probable.

En principio, parece interesante que los ngramas sean cuanto mayor es n, mucho mayor es el número de parámetros que deben ser estimados. Por ejemplo, si el vocabulario del lenguaje que se desea modelar es de tan sólo 500 palabras, utilizando bigramas se estimarían 250000 parámetros. Para el caso de trigramas 12, 5 millones. Es decir, cuanto más grande sea el ngrama utilizado, menos práctico resulta. Aunque se cuente con un gran corpus, el

número de parámetros que se deben estimar superará con creces al número de ejemplos disponibles. Por está razón normalmente se utilizan modelos de bigramas o trigramas, que en la práctica, y en contra de lo que podría parecer desde un punto de vista lingüístico, resultan ser suficientemente buenos predictores.

Puesto que la probabilidad de una frase se calcula como el producto de las probabilidades de sus ngramas las posibilidades ceros se propagarán y darán una mala estimación de la probabilidad (probabilidad cero) de aquellas frases que contengan ngramas no vistos previamente en el corpus de entrenamiento. Para evitar esto, es necesario utilizar mejores estimadores. Estimadores que tengan en cuenta que aquellos eventos que no han sido vistos en el corpus de entrenamiento podrían darse en la práctica. La mayoría de estos estimadores se basan en la reducción de una u otra forma de la probabilidad asignada a los eventos vistos previamente. Así esta masa de probabilidad descontada de los eventos vistos se puede repartir entre los eventos no vistos.

Los métodos por los que se obtienen estos estimadores suelen recibir el nombre de métodos de descuento y el proceso mediante el cual se realiza el descuento suele denominarse suavizado.

Conviene tener presente que la necesidad de utilizar modelos de lenguaje suavizados es el resultado de los siguientes condicionales:

- Cualquier combinación de palabras debe ser posible, esto es, no tiene que haber ninguna secuencia de palabras que tenga una probabilidad exactamente igual a cero.
- La cantidad de datos de entrenamiento disponible para la estimación de dichos modelos es siempre insuficiente, incluso si el corpus de entrenamiento consta de varios cientos de millones de palabras.
- Considerar a los eventos no vistos (en los datos de entrenamiento) cuando se utiliza validación cruzada.

Los algoritmos de suavizado más comunes incluyen backoff o deleted interpolation como forma de redistribuir la masa de probabilidad obtenida de los algoritmos de descuento, ya sea Witten-Bell discounting o Good-Turing discounting.

El algoritmo Witten-Bell Discounting es un concepto recurrente en procesamiento estadístico. Está basado en un evento de frecuencia cero considerado como un evento que aún no ha sucedido. Cuando ocurre, será modelado mediante la probabilidad de ver un

N-grama por primera vez. El cálculo de la probabilidad es muy simple de producir porque los N-gramas que se ven por primera vez representan el número de tipos de N-gramas en los datos de entrenamiento.

El algoritmo de Good-Turing Discounting vuelve a estimar la cantidad de masa de probabilidad para asignarla a los N-gramas con valor cero o muy bajo, observando el número de N-gramas con valores altos.

Los N-gramas se obtienen de los textos de entrenamiento que comparten las mismas características del lenguaje que las señales de entrada esperadas.

Los modelos de lenguaje basados en corpus como los N-gramas son evaluados separando el corpus en un conjunto de entrenamiento y un conjunto de prueba, entrenando el modelo en el conjunto de entrenamiento y evaluando sobre el conjunto de prueba. Estos conjuntos pueden ir variando de acuerdo a la cantidad de datos que se tengan.