UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA - PROCESAMIENTO DIGITAL DE SEÑALES

Detección de palabras fonéticamente similares orientada a reconocimiento automático de voz

TESIS

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN INGENIERÍA

PRESENTA:

CARLOS DANIEL HERNÁNDEZ MENA

DIRECTOR DE TESIS:
DR. JOSÉ ABEL HERRERA CAMACHO, FACULTAD DE INGENIERÍA

COMITÉ TUTOR:
DR. GERARDO SIERRA MARTÍNEZ, INSTITUTO DE INGENIERÍA
DR. FELIPE ORDUÑA BUSTAMANTE, CCADET
DR. ALFONSO MEDINA URREA, EL COLEGIO DE MÉXICO

CIUDAD DE MÉXICO, SEPTIEMBRE, 2018

JURADO ASIGNADO:

Presidente:	DR. FELIPE ORDUÑA BUSTAMANTE
Secretario:	DR. PABLO ROBERTO PÉREZ ALCÁZAR
1 er. Vocal:	DR. JOSÉ ABEL HERRERA CAMACHO
2 do. Vocal:	DR. GERARDO SIERRA MARTÍNEZ
3 er. Vocal:	DR. ALFONSO MEDINA URREA

Lugar o lugares donde se realizó la tesis: LABORATORIO DE TECNOLOGÍAS DEL LENGUAJE, FACULTAD DE INGENIERÍA, UNAM.

TUTOR DE TESIS:

DR. JOSÉ ABEL HERRERA CAMACHO

FIRMA

(Segunda hoja)

A mi madre:
La única persona en este mundo
que nunca ha dejado de creer en mí...
y que jamás lo hará.

Toda mi vida y todos mis sueños
te los dedico a ti mamá
Daniel.

Agradecimientos

Quiero agradecer primeramente a mi madre Gloria por su infinito apoyo, amor y cuidados que a lo largo de mi vida me ha regalado sin miramientos. Si algún día llega a servir de algo mi efímera existencia en esta tierra, eso únicamente se deberá a ti mamá.

Agradezco a mi padre, a mis abuelos y a mi querida tía Míriam por todos sus consejos y apoyo moral en todas las fases de mi vida.

Agradezco la guía y la paciencia de los miembros de mi comité tutoral: el Dr. Abel, el Dr. Gerardo, el Dr. Alfonso y el Dr. Felipe. Agradezco también a Iván Meza por su apoyo y asesorías en momentos difíciles.

Agradezco a mis amigos Aldo y Jeannete que han sido para mi un ejemplo a seguir en el mundo académico y a Juan Ramírez por su ayuda con el artículo JCR.

Agradezco a Nancy Norely, Alejandra Chavarría y Tara Hansen que sin obligación me ayudaron muchas veces con su conocimiento lingüístico. Agradezco también a Jaime Reyes por sus asesorías en los primeros semestres.

Agradezco al Mtro. Javier Cuétara por todo lo que sus clases aportaron a mi doctorado.

Agradezco a Elena Vera y a Valeriano Assem por su apoyo con mi programa de servicio social. Agradezco también a mis alumnos de servicio social por todo su arduo trabajo y en especial a Frederick por su excelente labor con la página del CIEMPIESS.

Agradezco a CEP-UNAM, CONACYT y al proyecto UNAM PAPIIT/DGAPA IT102314 por su apoyo financiero en mis estudios.

Finalmente agradezco a Dios por permitirme encontrar en todo momento a personas dispuestas a ayudarme. Yo siempre me he considerado como un becario suyo en esta vida y sé que la mejor forma de agradecerle será ayudando a otros a cumplir sus sueños, así como él ha permitido que se cumplan los míos.

Resumen/Abstract

Resumen

La presente tesis tiene como objetivo estudiar dos fenómenos poco investigados en el campo del reconocimiento automático de voz. Uno es el efecto de las palabras fonéticamente similares y el otro es el efecto de las vocales tónicas.

Por una parte, las palabras fonéticamente similares contribuyen a decrementar el rendimiento de este tipo de sistemas. Por tanto, se plantea una nueva definición de las mismas, se introduce una colección de herramientas y técnicas necesarias para su detección en corpus orales y se presenta un conjunto de soluciones para mejorar el rendimiento en sistemas de reconocimiento de voz con una gran densidad de estas palabras.

Por otro lado, para el estudio del segundo fenómeno, se realizan experimentos sobre el impacto de las vocales tónicas en sistemas de reconocimiento automático de voz, encontrando que: 1) es favorable tomarlas en cuenta en el diseño del reconocedor, porque estas mejoran su precisión y, 2) las vocales tónicas guardan una estrecha relación con las palabras fonéticamente similares.

Para este trabajo se crea un corpus oral para uso en reconocimiento automático voz de 17 horas de duración en el español del centro de México, junto con un conjunto de herramientas destinadas a la creación de transcripciones fonéticas precisas.

Finalmente, para lograr una mayor comprensión de los dos fenómenos antes descritos, se adaptan al español cuatro de los más destacados sistemas de reconocimiento automático de voz que cumplen con ser libres y de código abierto. Algunos de los experimentos realizados con estos sistemas son inéditos en el español hablado en México, y permiten en un futuro próximo, la creación de un sistema de reconocimiento de voz de un carácter más nacional.

Resumen/Abstract

Abstract

The present dissertation aims to study two little investigated phenomenons in the field of automatic speech recognition. One of those is the effect of phonetically similar words and the other one is the effect of tonic vowels.

In the first place, phonetically similar words contribute to decrease performance on these kind of systems. Therefore, it is proposed a new definition of them, a collection of software tools is introduced to detect them in oral databases, and it is presented a set of solutions destined to increase performance on speech recognition systems with a big density of these words.

On the other hand, in order to investigate the impact of tonic vowels, a set of experiments is performed, so it is found that: 1) tonic vowels contribute to increase performance on speech recognition systems and, 2) tonic vowels are closely related to the phonetically similar words.

For this work, it is created an oral corpus of 17 hours duration in Spanish of Central Mexico. It is also developed a set of software tools destined to perform accurate phonetic transcriptions.

Finally, in order to improve our understanding of the two phenomenons described, four of the most popular speech recognition engines are selected to be adapted to Mexican Spanish. This selection is based on systems that are free and open source. Some of the experiments with these four systems are unpublished and they contribute, in the future, to the creation of one speech recognition system made in the country.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Planteamiento del problema

La popularización masiva de los sistemas de reconocimiento automático de voz (sistemas ASR) se dio a partir de la década de los noventa. Esto significó que más personas pudieron tener acceso a sistemas de código abierto como Sphinx (Lee et al., 1990), desarrollado por la Universidad Carnegie Mellon, pese a que ya había en el mercado, diversos sistemas previos de carácter privado. Los sistemas desarrollados durante esta década, y que podemos considerar como sistemas estándar, están basados en modelos ocultos de Markov (HMM por sus siglas en inglés). Estos son capaces de reconocer habla continua, pueden ser independientes del hablante y pueden hacer tanto reconocimiento basado en gramáticas, como utilizando modelos de lenguaje.

Sin embargo, las investigaciones continúan a nivel mundial para mejorar el rendimiento de estos sistemas en condiciones demandantes, es decir, en condiciones de alto ruido, sin hablantes entrenados y con hablantes de acentos distintos, entre otras muchas variables.

Actualmente, son escasos los experimentos de reconocimiento de voz continua hechos en México, y más aún, usando el español hablado al interior del país. Así, esta tesis se propone realizar experimentos con varios sistemas de reconocimiento utilizando una base de datos (corpus) abierta, mayor a las que se han creado en México hasta el momento, y cuyo diseño es fruto del presente trabajo.

Por lo anterior, resulta preocupante que los recursos lingüísticos para lenguajes diferentes al inglés suelan ser escasos la mayoría de las veces, como es el caso del español de México. Esto es un problema porque limita la experimentación con sistemas ASR en nuestro idioma por parte de investigadores tanto mexicanos como extranjeros.

Con todo, en los sistemas ASR actualmente desarrollados, hemos podido identificar dos aspectos que merecen un estudio amplio. El uno es el reconocimiento de palabras

fonéticamente similares (PFS), y el otro es el tratamiento de las vocales tónicas, considerando el fenómeno conocido en lingüística como "desplazamiento acentual"[1.1].

Para el primer aspecto, es preciso estar al tanto de lo que en el campo de la percepción del habla (speech perception) se denomina como "vecinos léxicos" (lexical neighbors). Los vecinos léxicos son palabras que compiten dentro de la memoria humana por ser identificadas, es decir, compiten por el "acceso léxico". Un ejemplo exagerado pero exacto de esto, es cuando una persona lucha en su memoria por recordar una palabra que le es poco frecuente. En su mente surgen (se activan) palabras candidatas que van siendo descartadas una por una hasta que finalmente, surge vencedora una de ellas, es decir, esa palabra pasa, de estar guardada en la memoria, a formar parte del vocabulario de la persona, por lo tanto, se dice que la palabra obtiene el acceso léxico.

Algunas de las características que guardan los vecinos léxicos, es que varían entre sí en un solo fonema (Buchwald et al., 2008) y su proceso de reconocimiento se efectúa, evidentemente, de una manera dinámica (Salverda et al., 2007). Ahora bien, al conjunto de todos los vecinos léxicos que se activan dentro de la memoria de una persona durante el proceso de reconocimiento de una palabra dada, se le denomina "cohorte" (cohort) (Mousty et al., 1996).

No obstante, aunque en la literatura se pueden encontrar diversos modelos de percepción del habla como el SHORTLIST (Norris, 1994), el modelo TRACE (Mcclelland & Elman, 1986) o el modelo NAM (Goldinger et al., 1989), (Luce, 1986), que tratan de explicar el proceso de cómo una cohorte de vecinos léxicos activados compiten por el acceso léxico, el problema que subyace a todos ellos es que entre más vecinos léxicos se activen, más lento será el proceso de reconocimiento por parte de la persona.

Con esto en mente, se hace evidente que todos estos estudios sobre percepción del habla por humanos, guardan un interesante paralelismo con los sistemas de reconocimiento de voz automáticos. A saber, en un sistema ASR las palabras del diccionario de pronunciación también "compiten" por ser "elegidas", es decir que también compiten por un tipo de acceso léxico. Luego, debido a que la búsqueda en un diccionario de pronunciación se hace de forma alfabética, también es posible reconocer claramente el fenómeno de vecinos léxicos, que no son más que palabras muy similares del diccionario de pronunciación, que empiezan con la misma cadena de caracteres y que por lo tanto, son iguales hasta un cierto punto.

En consecuencia, era de esperarse que también en sistemas ASR ocurra que entre más palabras similares existan en el vocabulario, más bajo será el desempeño. Siendo esto reportado anteriormente en la literatura (Kim & Un, 1988).

No obstante, pese a todo este interesante paralelismo entre percepción humana y sistemas ASR, es difícil estudiar el fenómeno de las PFS, sobre todo cuando en la literatura no se ponen de acuerdo con su definición. El problema con las definiciones

[1.1] Ejemplos de desplazamiento acentual son: médico, medico y medicó; ejército, ejercito y ejercitó; título, titulo y tituló.

convencionales de palabra fonéticamente similar, es que suelen describir a aquella palabra que varía con respecto a otra, en sólo un fonema (Buchwald et al., 2008), como es el caso de los vecinos léxicos o de los "pares mínimos" en lingüística (Gil Fernández, 2007; Llisterri, 2012). Esto significa que más de una definición puede describir a palabras fonéticamente similares, pero la mayoría de estas definiciones no servirá para abordar el problema de las mismas, desde un punto de vista útil para el reconocimiento automático de voz. Sin mencionar que en ninguna de estas definiciones se habla del desplazamiento acentual.

Otro ejemplo que ilustra la importancia de tal definición, se remonta al hecho de que en las primeras investigaciones hechas sobre reconocimiento de voz, el reto era reconocer palabras aisladas en vocabularios muy limitados, pero a partir de la introducción de los modelos de lenguaje, aparecen en la escena de los sistemas ASR otro tipo de palabras conocidas como "pares doblemente confusos" o "pares doblemente confundibles" (double confusable pairs) que son palabras que cumplen con dos condiciones: la de tener la misma probabilidad de aparición y la de ser fonéticamente similares (Goldwater & Jurafsky, 2010). El problema con esta definición es que aunque define lo que es un par doblemente confundible, no define el significado de fonéticamente similar, el cual se deja como algo sobreentendido.

Es así cómo con las ideas de reconocimiento de palabras en humanos, surge en el presente trabajo la idea de atacar el problema del exceso de palabras fonéticamente similares en sistemas ASR. Lo cual conlleva a proponer una definición más precisa de las mismas, así como algoritmos para su detección y tratamiento.

Finalmente, para el segundo aspecto, se conoce que en fonética no sólo son importantes el punto y el modo de articulación (rasgos articulatorios), sino además, la posición de la vocal tónica (desplazamiento acentual) en la palabra. Esto último es relevante a la hora de hacer una transcripción fonética que indique división silábica, ya que a menudo se requiere distinguir a diptongos de hiatos. Por tanto, en este trabajo se plantea que la posición de la vocal tónica influye y es relevante por sí sola, en el reconocimiento automático de palabras, así como es relevante en el reconocimiento por parte de humanos. Por otra parte, también se propone que las palabras con desplazamiento acentual guardan una estrecha relación con las PFS, y que hecho, debería existir una definición de PFS que las incluyera.

El presente trabajo desarrolla estos dos aspectos de una manera amplia, experimentando con sistemas ASR.

1.2 Objetivo

Los objetivos centrales de este trabajo son dos. El primero es el de proponer soluciones metodológicas que resuelvan el problema de trabajar con un sistema de reconocimiento automático de voz continua, que tenga una gran cantidad de palabras

fonéticamente similares, y que por lo tanto, vea reducido su rendimiento. El segundo es el de estudiar el efecto de las vocales tónicas en el reconocimiento automático de voz, así como su relación con las palabras fonéticamente similares.

Para lograr estos objetivos se han considerado las siguientes metas:

- Diseñar y construir un corpus en el español hablado en México, que sirva para la validación de experimentos en esta área.
- Revisar, y diseñar en su caso, un conjunto de herramientas de código abierto y técnicas, que permitan la creación de sistemas de reconocimiento de voz en español del centro de México, y su adaptación a necesidades específicas.
- Proponer una definición del término "palabras fonéticamente similares", desde el punto de vista de un sistema automático de reconocimiento de voz, con el fin de localizar y medir el efecto de estas palabras en dichos sistemas.
- Realizar pruebas de reconocimiento continuo de voz con un corpus en el español hablado en el centro de México, utilizando los sistemas de reconocimiento más exitosos hasta hoy (que estén en versión libre).
- Realizar experimentos para comprender la relevancia que tienen las vocales tónicas en el reconocimiento automático de voz, así como su relación con las palabras fonéticamente similares.

1.3 Metodología

Es necesario identificar el conjunto de rasgos de la voz que resulten útiles para este trabajo. En la literatura se pueden encontrar numerosos ejemplos de cómo algunos rasgos de la voz son estudiados para diversos propósitos. Algunos de estos son, la duración de las palabras (Deng et al., 1989), la edad de los hablantes (Dalston, 1975), su sexo (Adda-Decker & Lamel, 2005), los rasgos articulatorios (Scharenborg, 2010) o incluso, los sonidos que hay entre palabra y palabra (Nakatani & Dukes, 1977).

Ahora bien, para estudiar cualquiera de estas características de la voz, se necesita contar con un corpus oral con el correspondiente rasgo de la voz marcado (etiquetado). Al no estar disponible ningún corpus oral adecuado, se decidió que debía diseñarse uno, poniendo especial atención en los rasgos articulatorios del español de México, a través del conocimiento de sus alófonos. Por tanto, se descartó el enfocarse en rasgos como la velocidad de habla. En cuanto al sexo y a la edad, se sabe bien que pueden influir en la precisión del reconocimiento, pero no son necesariamente importantes para el estudio de las palabras similares, ni de las vocales tónicas.

Asimismo, debido a los estudios sobre palabras fonéticamente similares, se consideró oportuno realizar un análisis de las diferentes construcciones conceptuales alrededor del

término "palabras fonéticamente similares", considerando su impacto en los sistemas ASR.

Con base en este análisis sería posible realizar algoritmos y metodologías adecuadas para la detección, conteo y tratamiento de palabras fonéticamente en sistemas ASR en español de México.

Adicionalmente, se consideró que era importante realizar experimentaciones sobre el uso de vocales tónicas en diferentes sistemas ASR. Con esto, se creyó pertinente manejar cuatro de los sistemas de reconocimiento libres más destacados en la actualidad que, conjuntamente con una base de datos en español, dieran un panorama inédito del reconocimiento de voz en nuestro idioma.

La adecuación de los cuatros sistemas de reconocimiento antes mencionados, implica un arduo trabajo en el diseño de herramientas propias para el español hablado en México, pero también perfila en un corto plazo, el diseño total de un sistema de reconocimiento de voz autóctono.

1.4 Actividades

Para lograr los objetivos propuestos, se llevaron a cabo las siguientes actividades:

- **Diseño del corpus CIEMPIESS**. Se realizó una investigación bibliográfica para conocer distintas bases de datos de sistemas ASR. De este modo se definieron las características requeridas para diseñar un corpus propio. Luego, se contactó con Radio-IUS[1.2] para obtener grabaciones con los permisos correspondientes. Posteriormente se formó un equipo de trabajo de 20 ayudantes de servicio social para realizar las transcripciones y el etiquetado a nivel palabra, y otro equipo que incluía al autor para revisarlas. El autor seleccionó el alfabeto fonético apropiado con base en una investigación plasmada en la presente tesis.
- **Sistemas ASR**. Se realizó un estudio teórico-práctico de cuatro de los sistemas ASR más populares en el mundo que son de software libre y de código abierto. En un trabajo de programación, se adecuaron estos sistemas para su uso con el español del centro de México. Dentro de las herramientas utilizadas, destaca la creación de diccionarios de pronunciación en español de México.
- **Palabras fonéticamente similares**. Se realizó un estudio teórico sobre las diferentes concepciones de palabras fonéticamente similares y se incorporó una definición en el contexto de los sistemas ASR, que el autor propone como definición funcional.

[1.2] Radio-IUS es una estación de radio de la Facultad de Derecho de la UNAM. Ver: http://www.derecho.unam.mx/cultura-juridica/radio.php

- **Experimentos**. Se realizaron tres tipos de experimentos. Unos estuvieron dedicados a la validación de diversas herramientas de software y algoritmos creados en esta tesis, así como la verificación del correcto funcionamiento del corpus CIEMPIESS en diferentes sistemas ASR. Otros experimentos tuvieron el objetivo de detectar y medir la cantidad de PFS en vocabularios de sistemas ASR. Finalmente, el tercer grupo de experimentos tuvo el objetivo de medir el impacto de palabras con desplazamiento acentual, por medio del sistema de reconocimiento de voz Sphinx3.

1.5 Estructura de la tesis

Este trabajo está dividido en siete capítulos, una sección de referencias y una sección de apéndices, conteniendo lo siguiente:

- **Capítulo 1. Introducción**. Aquí se describe el objetivo, el planteamiento del problema, la metodología y la estructura de la tesis.
- **Capítulo 2. Marco teórico**. Se tratan los diversos paradigmas de reconocimiento de voz, extracción de características de la señal de voz y los sistemas de reconocimiento automático.
- **Capítulo 3. Fonética y fonología**. Se habla sobre la importancia que tienen la fonética y la fonología en el campo del reconocimiento de voz. Se dan conceptos básicos sobre lo que son los alfabetos fonéticos computacionales, se explican las reglas de transcripción del alfabeto Mexbet y se muestra cómo implementarlas en un código de programación. También se muestran algoritmos para hacer división silábica y acentuación de palabras.
- **Capítulo 4. Recursos lingüísticos**. Se exponen diversas técnicas para la creación de corpus orales para uso en sistemas ASR. Se describe el diseño de dos corpus diseñados en esta tesis (CHM150 y CIEMPIESS).
- **Capítulo 5. Definición y algoritmo PFS**. Se presenta una definición del término "palabras fonéticamente similares" desde el punto de vista del reconocimiento automático de la voz, y se diseña un algoritmo para detectar PFS.
- **Capítulo 6. Experimentos**. Se presentan experimentos para validar las diferentes herramientas de software creadas como parte del trabajo de esta tesis. Se comparan cuatro sistemas ASR en cuanto a su desempeño para el corpus CIEMPIESS. Se presentan experimentos para mostrar el impacto que tienen las palabras acentuadas en el reconocimiento y se prueba la eficacia del algoritmo PFS para agrupar palabras fonéticamente similares en español, contenidas en un archivo texto.

- **Capítulo 7. Conclusiones**.
- **Apéndices**. En esta sección se presentan tablas útiles y una serie de manuales que explican detalladamente cómo utilizar todas las herramientas presentadas para hacer reconocimiento de voz en español del centro de México, con los sistemas Sphinx y HTK, en 11 apéndices.
- **Referencias**.

Capítulo 2

Marco teórico

En este capítulo se exponen brevemente los fundamentos teóricos del reconocimiento automático de voz y luego se presenta una descripción detallada de los procesos que lo conforman, al final se presentan varios sistemas de reconocimiento de voz de código abierto que han tenido relevancia tanto en la investigación como en las aplicaciones de ingeniería.

Como complemento de los temas tratados aquí, se destacan los capítulos de marco teórico de las tesis (Pérez-Pavón, 2006) y (Reyes-Cortés, 2010), en donde se describen los fundamentos del reconocimiento de voz de una manera sencilla pero sin falta de rigor.

De igual manera, se destacan textos clásicos en esta área, que presentan el problema del reconocimiento automático de voz con profundo rigor matemático, una mayor precisión en el lenguaje y ejemplos para una mayor compresión de los conceptos: (Huang et al., 2001; Jelinek, 1997; Jurafsky & Martin, 2000; Rabiner & Juang, 1993)

2.1 El problema del reconocimiento de voz

El área del reconocimiento de voz pretende convertir el habla en texto, es decir, mapear una locución o señal acústica a una cadena de símbolos escritos. Para lograrlo, se vale de un conjunto de algoritmos computacionales que dan vida a los llamados "sistemas de reconocimiento automático de voz".

El tipo de dificultades que desde el principio han enfrentado este tipo de sistemas es la inmensa variabilidad de las realizaciones de voz por parte de los diferentes tipos de hablantes (hombres, mujeres, niños, ancianos, etc), sin mencionar que esa misma variabilidad aparece aún en el habla de un mismo hablante determinado, dependiendo de si está enojado, triste, enfermo, etc.

Es por esta variabilidad que la solución al problema ha escapado a enfoques de-

terminísticos, siendo necesario atacarlo de una manera probabilística y con la ayuda del aprendizaje automático o "machine learning" en inglés. No obstante, es posible dar una descripción matemática sencilla del problema general, definiendo formalmente todos sus elementos.

Sea O una locución de entrada. La O hace referencia a "observación acústica". En la práctica, una locución es primero dividida en pequeños segmentos (o "frames" en inglés) que son luego tratados por medio de un proceso llamado "extracción de atributos".

Una observación de entrada O puede descomponerse en frames O_i consecutivos en el tiempo. En la práctica, estos frames suelen tener una duración constante, fijada por lo regular en el intervalo de 5 a 50 milisegundos:

$$O = o_1, o_2, o_3, ..., o_t \tag{2.1}$$

Se asume que O contiene o representa una secuencia de símbolos de un alfabeto Ω. De manera análoga, es posible representar un mensaje escrito W como si estuviera formado por una cadena de palabras pertenecientes a un diccionario D, y que a su vez, los mensajes pertenecen a un lenguaje L:

$$W = w_1, w_2, w_3, ..., w_t \quad W \in L \tag{2.2}$$

La ecuación 2.3 describe el problema del reconocimiento de voz de manera óptima, aunque lo hace también de una manera ideal. El cálculo de $P(W|O)$ implica que para cualquier mensaje escrito W se requiere obtener la probabilidad condicional de todas las locuciones O posibles dentro del alfabeto Ω, lo cual es imposible. Sin embargo, gracias al teorema de Bayes es posible hacer una manipulación de la ecuación 2.3 en términos más prácticos.

$$\hat{W} = \underset{W \in L}{\arg} \ \text{máx} \ P(W|O) \tag{2.3}$$

El teorema de Bayes establece que:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{2.4}$$

Por lo tanto, si se adecua conforme a las variables de la ecuación 2.3 se tiene que:

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)} \tag{2.5}$$

Ahora bien, sustituyendo la ecuación 2.5 en la ecuación 2.3 se tiene:

$$\hat{W} = \underset{W \in L}{\arg} \ \text{máx} \ \frac{P(O|W)P(W)}{P(O)} \tag{2.6}$$

Las probabilidades en la ecuación 2.6 son más fáciles de calcular, excepto por la probabilidad $P(O)$. La probabilidad $P(O)$ implica calcular la probabilidad de todas las observaciones O posibles en el alfabeto Ω, que es justo lo que se trataba de evitar en la ecuación 2.3. Sin embargo, debido a que $P(O)$ es constante para todas las secuencias O y valiéndose del hecho de que es un denominador común a todas las secuencias de O dado W, la ecuación 2.3 puede reducirse a:

$$\hat{W} = \underset{W \in L}{\arg} \ \text{máx} \ P(O|W)P(W) \tag{2.7}$$

En donde, $P(O|W)$ es la probabilidad de la observación y se calcula mediante el modelo acústico, y $P(W)$ (probabilidad a priori) se calcula mediante el modelo de lenguaje tal como lo muestra la ecuación 2.8:

$$\hat{W} = \underset{W \in L}{\arg} \ \text{máx} \ (\ \underbrace{P(O|W)}_{Modelo\ Acústico} \cdot \overbrace{P(W)}^{Modelo\ de\ Lenguaje} \ ) \tag{2.8}$$

2.1.1 Características de la señal de voz

La señal de voz tiene la característica de ser una señal que varía lentamente (con respecto a otras señales, como el audio musical). Para periodos de tiempo corto (entre 5 y 50 ms) es, en términos prácticos, estacionaria(Rabiner & Juang, 1993).

La señal de voz es una onda sonora producida por el aparato fonador humano, que presenta un tono fundamental producido por la vibración de las cuerdas vocales. La media de esta frecuencia o tono fundamental se sitúa en torno a los 125 hertz para los hombres y 250 hertz para las mujeres. Los valores extremos de tono fundamental en los hombres se encuentran entre los 80 y 200 hertz, y para los mujeres están entre los 150 y 400 hertz, pudiendo ser superiores en los niños (González-Sigüenza, 2008).

Normalmente los sistemas telefónicos recogen la voz en un rango de entre los 300 y 3,400 hertz para tratar de abarcar todos los rangos de voz posibles, en un ancho de banda suficiente para que la voz sea inteligible. Es por eso que la frecuencia de muestreo en los sistemas telefónicos está estandarizada a 8,000 muestras por segundo (Cabeza-Galan, 2000). Sin embargo, en este trabajo, los experimentos hechos con reconocedores de código abierto se trabajan a 16 mil muestras por segundo, lo cual permite detectar con mayor facilidad, rasgos importantes de la voz como los formantes.

Ahora bien, para entender qué son los formantes, es preciso recordar que las cuerdas vocales son la fuente principal de vibración en el aparato fonador, el cual está formado

por varias cavidades de resonancia que amplifican ciertas frecuencias y atenúan otras, por medio del fenómeno de interferencia de las ondas. Por tanto, estas cavidades que funcionan como cajas de resonancia se convierten en fuentes de sonido aperiódico o ruido, cuando las cuerdas vocales no vibran y el aire pasa por algunas de esas zonas constreñidas (Celdrán, 1998).

Esto significa que un formante es una de las resonancias ocurrida en alguna cavidad del aparato fonador, o para decirlo de otro modo, un formante es una concentración de energía alrededor de una frecuencia en particular de la señal de voz. Por lo tanto, cada formante corresponde con una frecuencia de resonancia en el tracto vocal (que va desde la laringe hasta los labios).

Por convención, los formantes se enumeran de abajo hacia arriba desde la frecuencia más baja, siendo F0 el tono fundamental producido por las cuerdas vocales, F1 el formante siguiente de frecuencia más baja y así sucesivamente (Wood, 2005).

2.2 Parametrización de la señal de voz

El proceso de parametrización de la señal de voz consiste en obtener un conjunto de vectores con información que resulte más sencilla de discriminar después, por medio de algún algoritmo de reconocimiento de patrones (ej. HMM, SVM, Redes Neuronales, etc). De este modo, existe una gran variedad de posibilidades para parametrizar una señal de voz, siendo los métodos basados en análisis espectral, los que son considerados como los más importantes en el módulo de entrada o"front-end" de un sistema de reconocimiento de voz.

A continuación se presentan las parametrizaciones más utilizadas en sistemas de reconocimiento de voz.

2.2.1 Coeficientes de predicción lineal

Los coeficientes de predicción lineal o LPC, por sus siglas en inglés, son un tipo de parametrización muy utilizada. Modelan el tracto vocal por medio de la representación de la envolvente espectral de una señal digital de voz. Los máximos de los LPC representan los formantes de voz; además, los LPC tienen un método rápido de cálculo, por lo que son adecuados para sistemas en tiempo real.

Dada una señal de voz $s(n)$ con n muestras, se puede aproximar por medio de una combinación lineal entre p muestras precedentes y a coeficientes LPC (Markel & Gray, 1976), como se muestra en la ecuación 2.9:

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \cdots + a_p s(n-p) \tag{2.9}$$

2.2.2 Coeficientes cepstrum en frecuencias mel

Los "coeficientes cepstrum en frecuencias mel" o MFCCs por sus siglas en inglés, son coeficientes para la parametrización de la señal de voz basados en la percepción auditiva en los seres humanos. Se derivan de la transformada de Fourier (FT) o de la transformada coseno discreta (DCT). La diferencia básica entre la FT o la DCT y los MFCC es que en los MFCC las bandas de frecuencia están situadas logarítmicamente (según la escala mel), que modela la respuesta auditiva humana más apropiadamente que las bandas espaciadas linealmente de FT o DCT. Esto permite un procesado de datos más eficiente, por ejemplo, en compresión de audio.

Los MFCCs se calculan comúnmente de la siguiente forma:

- Se calcula la transformada de Fourier de todos los frames de la señal.
- Se mapea la potencia espectral obtenida en cada frame a la escala mel, usando una función ventana triangular.
- Se calcula el logaritmo de las potencias espectrales en escala mel, obtenidas en el paso anterior
- Se calcula la transformada coseno discreta de la lista de potencias espectrales logarítmicas en escala mel, calculada en el paso anterior, como si fuera una señal.
- Los MFCCs son las amplitudes del espectro resultante.

2.2.3 Modelos de mezclas gaussianas

Una "gaussian mixture model" (GMM) es una función paramétrica de densidad de probabilidad representada como la suma ponderada de densidades gaussianas. Las GMMs usualmente se utilizan como modelos paramétricos de funciones de probabilidad de mediciones continuas en sistemas biométricos, tales como componentes espectrales relacionadas con el tracto vocal en un sistema de reconocimiento de voz.

Los parámetros de las GMM se estiman a partir de los datos de entrenamiento utilizando el algoritmo de maximización de la media (expectation-maximization) por medio de una estimación máximo a posteriori de un modelo bien entrenado previamente (Reynolds, 2009).

2.3 Paradigmas de reconocimiento de voz

En esta sección se muestran algunas de las técnicas clásicas para hacer reconocimiento de voz, las cuales involucran técnicas de reconocimiento de patrones. Sin embargo, se

presta especial atención a la técnica de los modelos ocultos de Markov, ya que todos los sistemas de reconocimiento de voz presentados en este trabajo están basados en esta.

2.3.1 Modelos ocultos de Markov

Un modelo oculto de Markov (HMM), es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos. El objetivo es determinar los parámetros desconocidos (u ocultos, de ahí el nombre) de dicha cadena a partir de los parámetros observables. Los parámetros extraídos se pueden emplear para llevar a cabo sucesivos análisis, por ejemplo en aplicaciones de reconocimiento de patrones.

Existen tres problemas canónicos asociados a los modelos ocultos de Markov:

- Dados los parámetros del modelo, obtener la secuencia más probable de estados ocultos que pudiera haber generado una secuencia de salida dada. Este problema se resuelve con el algoritmo de Viterbi.
- Dados los parámetros del modelo, calcular la probabilidad de una secuencia de salida en particular. Este problema se resuelve con el algoritmo de avance-retroceso.
- Dada una secuencia de salida o un conjunto de tales secuencias, determinar el conjunto de estados de transición y probabilidades de salida más probables. En otras palabras, entrénense a los parámetros del HMM dada una secuencia de datos. Este problema se resuelve con el algoritmo de Baum-Welch.

Otra manera de considerar a un HMM, es como una máquina de estados finitos que cambia de estado una vez por unidad de tiempo. Cada tiempo t que está en un estado j, un vector o_t es generado con base en la función de densidad de probabilidad $b_j(o_t)$. Luego, las transiciones del estado i al estado j son también probabilísticas y están gobernadas por la probabilidad discreta a_{ij} (Young et al., 2006). Esto queda representado gráficamente en la figura 2.1.

Es así como la secuencia observada de vectores de voz o_t que corresponde con la locución de entrada, es generada por medio del modelo de Markov. Una manera más clara de explicar esto, sería decir que a un modelo de Markov, como el de la figura 2.1, entra una secuencia de vectores de atributos (por ejemplo, vectores MFCC) y el modelo entrega a la salida, una secuencia de vectores observados o_t. Ahora bien, lo que el modelo de Markov intenta hacer es generar la misma secuencia de entrada diciendo qué probabilidad hay de que esa secuencia haya sido generada con ese modelo de Markov en particular. Es por esto que Markov es considerado un modelo generativo.

En el proceso de entrenamiento, los parámetros del modelo de Markov se modifican para maximizar la probabilidad de que esa secuencia de salida haya podido ser generada

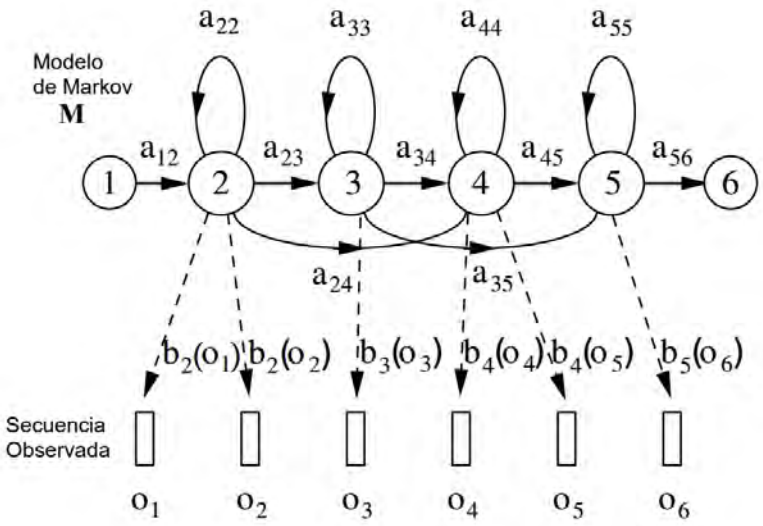

Figura 2.1: Representación gráfica de un HMM.[2.1]

con ese modelo de Markov particular. En contraparte en el proceso de reconocimiento, se busca el modelo de Markov que con mayor probabilidad haya dado lugar a esa secuencia de vectores observados a la entrada.

2.3.2 Redes neuronales

Las redes neuronales son un paradigma de reconocimiento de patrones basado en la manera en que funciona el sistema nervioso humano, el cual consiste en un sistema de neuronas interconectadas que trabajan juntas para producir un estímulo de salida (Hertz et al., 1991).

Una red neuronal está hecha a partir de unidades más pequeñas que individualmente se llaman neuronas. Cada una de estas neuronas recibe un conjunto de señales de entrada a través de sus interconexiones y con ello emite una señal de salida. Esta señal de salida es producida por una de las siguientes tres funciones (Bishop, 1995; Hagan & Demuth, 1996):

- Función de excitación, casi siempre consiste en la suma de todas sus entradas, y luego esa suma se multiplica por el peso de su interconexión. Si el peso es mayor o igual a cero, la conexión se llama excitatoria; de otro modo, se le conoce como inhibitoria.
- Función de activación, que depende de la función anterior, y es opcional ya que

[2.1] Imagen extraída de (Rabiner & Juang, 1993)

en caso de no implementarse, puede suplirse con la función de excitación.

- Función de transferencia, que se aplica al valor de salida de la función de activación y se utiliza para limitar la salida de la neurona, para que entregue valores dentro de intervalos bien definidos como 0 y 1, ó -1 y 1, etc.

2.3.3 Máquinas de vectores de soporte

Las máquinas de vectores de soporte, o SVM por sus siglas en inglés (support vector machines), son una familia de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik en los laboratorios AT&T.

Los SVM se relacionan directamente con problemas de clasificación y de regresión. Requieren de un conjunto de entrenamiento consistente en ejemplos de las clases que se quieren clasificar. Si el conjunto de esas clases es infinito, el problema de clasificación se convierte entonces en un problema de regresión. Después de que el SVM analiza el conjunto de entrenamiento y crea un modelo que lo explica, le es posible utilizarlo después para clasificar nuevas entradas desconocidas (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Flores, 2004).

Ahora bien, de manera más precisa y a la vez abstracta, un SVM crea un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta, para así poder resolver problemas de clasificación o de regresión. De esta manera, si se tiene una adecuada separación entre clases, se tendrá una clasificación correcta (Gales et al., 2009; Martín-Iglesias et al., 2005; Solera-Ureña et al., 2007).

2.3.4 Deep learning

El "aprendizaje profundo", mejor conocido universalmente como "deep learning", es un conjunto de técnicas de aprendizaje de máquina que intenta aprender representaciones de datos. Un ejemplo de esto, aterrizado al reconocimiento de voz, sería pedirle a un sistema de deep learning que haga reconocimiento de voz contando únicamente con los archivos de audio en formato RAW o WAV y con las transcripciones en archivos de texto plano, sin explicarle ningún tipo de alfabeto fonético, ni de parametrización de la señal de voz, ni ninguna otra información adicional (Mohamed et al., 2009).

De este modo, el sistema tendría que analizar los archivos de audio y las transcripciones de manera exhaustiva hasta encontrar una representación "óptima" de los datos analizados. El problema es que esta representación no sería un modelo de Markov o un conjunto de fonemas, palabras o MFCCs ni nada parecido, sino una intricada maraña de estados ligados y probabilidades, muy difícil de dilucidar por parte de un ser humano (Seide et al., 2011; Yu & Deng, 2015).

Es preciso decir que el deep learning toma su nombre de las redes neuronales pro-

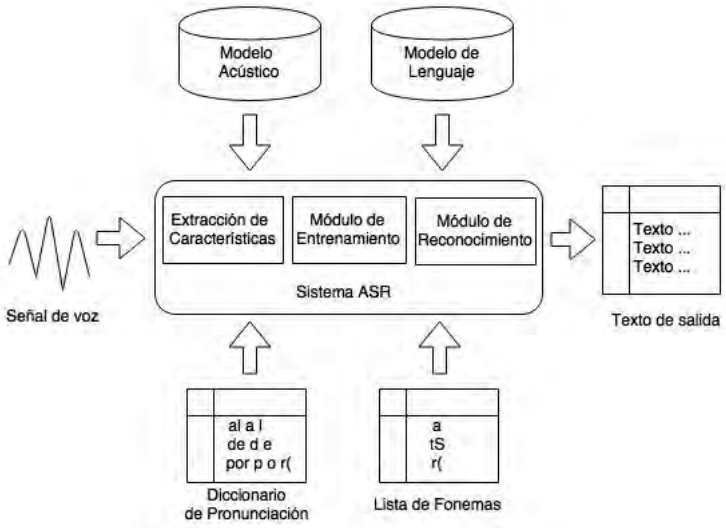

Figura 2.2: Diagrama a bloques de un sistema ASR convencional.

fundas o "deep neural netwoks", ya que estas están basadas en varias capas. Luego entonces, el calificativo de "deep" ("profundo" en inglés) en las "deep neural netwoks", proviene de que estas tienen muchas más capas, y están mucho más conectadas entre sí, con respecto a las redes neuronales tradicionales.

2.4 Requerimientos de entrada de un sistema ASR

A pesar de que existe una gran variedad de paradigmas de reconocimiento y de tipos distintos de parametrizaciones para la señal de voz, lo cierto es que aún hoy en día, los sistemas de reconocimiento estándar se basan en el paradigma de los modelos ocultos de Markov, con una parametrización basada en MFCCs. Por esta razón, la presente tesis se enfoca más en este paradigma, y todas las pruebas de reconocimiento presentadas en secciones posteriores están hechas sobre sistemas de esta clase.

No obstante, sin importar el paradigma de reconocimiento o la parametrización, existen ciertos puntos en común que cualquier sistema ASR requiere como elementos de entrada. En la figura 2.2 se muestra un diagrama a bloques de un sistema ASR convencional.

Los elementos de entrada que aparecen en la figura 2.2 son:

- La señal de voz de entrada, como un corpus oral de entrenamiento y prueba.
- El modelo acústico.
- El modelo de lenguaje.

- El diccionario de pronunciación.
- El alfabeto fonético.

En esta sección se explica brevemente la relevancia que tienen cada uno de estos elementos y su razón de ser en el sistema.

2.4.1 Los corpus orales

Un corpus lingüístico es una colección relativamente rica en materiales lingüísticos como pueden ser: texto, grabaciones de voz, video o combinaciones de lo anterior. En general, un corpus lingüístico se crea para estudiar uno o varios lenguajes en particular. En la sección 4.1 se aborda ampliamente este tema, pero lo que por ahora es importante destacar es que un corpus diseñado para uso en reconocimiento de voz debe proveer al sistema de los siguientes elementos:

- Grabaciones de voz.
- Transcripciones escritas de las grabaciones de voz.
- Alineaciones temporales de las grabaciones de voz (opcional).

Las grabaciones pueden o no, tener ruido, música de fondo, o pueden ser de una calidad alta o baja, según el tipo de reconocedor que se desee entrenar o el fenómeno que se desee estudiar. En general, siempre debe tratarse de recolectar el mismo tipo de audio que se quiere reconocer.

Las grabaciones deben estar, por supuesto, en un formato digital. Los formatos más utilizados para uso en reconocimiento de voz son el formato WAV de Microsoft, SPHERE de NIST[2.2] o FLAC[2.3], que es un tipo de algoritmo de compresión como el MP3, pero sin pérdidas de calidad.

Las grabaciones en el corpus de entrenamiento deben estar asociadas con sus respectivas transcripciones ortográficas, que por lo regular se encuentran en uno o varios archivos de texto plano. Estas transcripciones ortográficas deben registrar con precisión las locuciones provenientes de cada una de las grabaciones en el corpus.

La alineación temporal es el proceso que indica en qué instante de tiempo inicia o termina algún fenómeno dentro de la grabación de voz. Por ejemplo, se puede indicar en qué milisegundo empieza y en qué milisegundo termina cada palabra en la grabación, así como también se podría indicar lo mismo, pero por cada fonema. El resultado de este

[2.2] National Institute of Standards and Technology: http://www.nist.gov/

[2.3] Para más información sobre el formato FLAC, ver: https://xiph.org/flac/

proceso se ve registrado en archivos que normalmente se conocen como “etiquetas”. De hecho, vulgarmente se le conoce al proceso de alineación temporal como “etiquetado”.

Sin embargo, estas etiquetas temporales no son cien por ciento necesarias para un sistema ASR. Debido a que el proceso de etiquetado es bastante arduo y costoso, lo más común es que un corpus para ASR no incluya etiquetas de ningún tipo, pero sí las transcripciones ortográficas. Si esto ocurre, el sistema es capaz de deducir por si mismo un alineamiento temporal de los datos aunque necesite mayor cantidad de estos.

Por regla general, entre más información provenga del corpus de entrenamiento, menor cantidad de audio se necesita para obtener mayor precisión en el reconocimiento.

2.4.2 El modelo acústico

El proceso de “aprendizaje” en un sistema ASR es conocido como entrenamiento. El entrenamiento consiste en que el sistema toma un audio del corpus de entrada y lo “analiza” junto con su transcripción para “saber qué dice”. Luego toma el siguiente audio y hace lo mismo, y así hasta “analizar” todo el corpus.

Ahora bien, gracias a los algoritmos de aprendizaje máquina, propios del tipo de paradigma de reconocimiento elegido para el sistema en cuestión, se observa que poco a poco, durante el proceso de aprendizaje, comienzan a emerger patrones distinguibles, provenientes de los audios analizados. Y es así como al final, estos patrones son almacenados dentro del modelo acústico.

Por lo tanto, un modelo acústico es un conjunto de archivos computacionales que contienen una representación matemática del fenómeno acústico que se desea reconocer. El modelo acústico es el resultado del aprendizaje obtenido por el sistema tras analizar una por una, todas las grabaciones, transcripciones y etiquetas proporcionadas por el corpus de entrada, y su “forma”, depende completamente del paradigma de reconocimiento del sistema. Inclusive su formato computacional, depende totalmente del sistema particular que se esté utilizando.

La figura 2.3 muestra, de manera esquemática, un modelo acústico basado en HMM. En este caso, cada rectángulo representa un HMM para un fonema en específico. Un HMM se compone de la información sobre la forma que tiene el vector de atributos de la señal de audio de entrada, obtenido mediante el proceso de parametrización de la voz (puede tener 39 coeficientes o 22, puede tener la información sobre la energía o no tenerla, etc.), así como de la definición de los estados del propio HMM. En esta figura, la definición de estados está representada por medio de un vector de medias y un vector de varianzas. Esto es así porque las densidades de probabilidad de salida $b_j(o_t)$ de cada estado en el HMM (ver figura 2.1) son gaussianas. Por último, se observa también la matriz de transición del HMM.

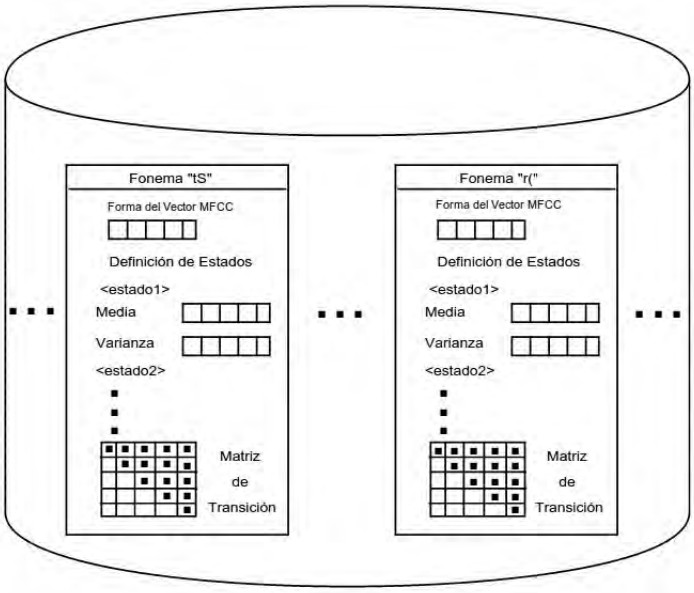

Figura 2.3: Diagrama esquemático de un modelo acústico.

2.4.3 El alfabeto fonético

Un alfabeto fonético es un conjunto de símbolos que registra los sonidos del habla de una o varias lenguas. La diferencia entre un alfabeto fonético y uno ortográfico o convencional, radica en que el segundo está lleno de convenciones provenientes de la tradición.

Un alfabeto fonético, en cambio, está diseñado de una manera científica y trata de registrar con precisión los fenómenos que ocurren en la realización de una lengua. Un ejemplo de esto es la letra o grafía "h" que en el español de México no tiene sonido alguno, pero que sin embargo se escribe en palabras como "hola", "hilo", etc. Por tanto, un alfabeto fonético no tomaría en cuenta de ninguna manera esta grafía, y de hecho, no lo hace.

Un sistema ASR configurado para el reconocimiento de fonemas (como todos los presentados en esta tesis) requiere un alfabeto fonético, el cual es introducido al sistema por medio de un archivo que contiene simplemente una lista de los símbolos que componen ese alfabeto fonético particular, como aparece ilustrado en el bloque llamado "lista de fonemas" en la figura 2.2.

En el presente trabajo, el alfabeto fonético utilizado se llama Mexbet, que fue creado en el año 2004 (Cuetara-Priede, 2004). Mexbet fue diseñado expresamente para el español hablado en el centro de México y esa fue la razón de haber sido elegido para los experimentos de esta tesis.

En la sección 3.6 se presenta el alfabeto Mexbet con todo lujo de detalles y en los apéndices F, H y I se presentan sus reglas de transcripción; así como, sus símbolos y equivalencias con otros alfabetos fonéticos.

2.4.4 El diccionario de pronunciación

El diccionario de pronunciación es un elemento muy importante en los sistemas ASR que están configurados para el reconocimiento de fonemas. Esto último se resalta debido a que no todos los tipos de sistemas ASR requieren uno.

Por ejemplo, los sistemas configurados para reconocer palabras o frases completas no requieren un diccionario de pronunciación como tal, sino que sus modelos acústicos estarán diseñados de tal manera que las secuencias observadas o_t de salida (figura 2.1) se mapeen directamente a las palabras o frases, que resulten del reconocimiento.

En el caso de los sistemas ASR configurados para el reconocimiento de fonemas (como es el caso de todos los utilizados en esta tesis), se requiere de un diccionario de pronunciación para indicar al sistema cuál es la pronunciación fonética de todas las palabras de su vocabulario. Esta transcripción fonética resulta ser una secuencia de símbolos pertenecientes a un alfabeto fonético, que representa los sonidos que conforman dicha palabra.

Un ejemplo de esto se observa en la figura 2.4, que muestra claramente como el diccionario de pronunciación es un archivo de texto plano, el cual contiene una lista de palabras ordenadas alfabéticamente. Cada palabra es seguida de su pronunciación fonética, y cada símbolo de esta, se encuentra separado por espacios en blanco. En esta imagen en particular, los símbolos que se observan pertenecen al alfabeto Mexbet.

Una variante posible de un diccionario de pronunciación podría ocurrir en un sistema ASR configurado para el reconocimiento de sílabas. En un sistema así, el diccionario de pronunciación contendría en vez de una secuencia de fonemas, una secuencia de sílabas que podrían estar o no representadas por medio de símbolos de un alfabeto fonético.

2.4.5 El modelo de lenguaje

Como se muestra en la ecuación 2.8 de la sección 2.1, la probabilidad a priori $P(W)$ debe ser proporcionada por medio del modelo de lenguaje. Luego entonces, el modelo de lenguaje es un mecanismo que sirve para calcular la probabilidad de una palabra w_k ocurrida en una locución, dada la secuencia $W_{k-1} = w_1, \ldots, w_{k-1}$ de palabras previamente pronunciadas. La ecuación 2.10 muestra un método para calcular precisamente esta probabilidad, donde $P(\boldsymbol{W})$ representa un modelo de lenguaje.

Figura 2.4: Ejemplo de un diccionario de pronunciación.

$$P(\boldsymbol{W}) = \prod_{k=1}^{K} P(w_k | w_{k-1}, \ldots, w_1) \tag{2.10}$$

Elegir el modelo de lenguaje es una tarea sumamente importante, ya que esto afectará significativamente el desempeño del sistema. Algunos tipos de modelos de lenguaje utilizados para reconocimiento de voz son (Reyes-Cortés, 2010):

- **Modelos de lenguaje uniformes**: Cada palabra en cualquier oración tiene igual probabilidad.
- **Modelos de lenguaje estocásticos**: La probabilidad de una palabra dada, se calcula con base en la secuencia de palabras que precedieron a esa palabra. Los modelos basados en n-gramas caen dentro de esta categoría.
- **Modelos de lenguaje de estados finitos**: La probabilidad se calcula con base en las reglas de una gramática. Cabe hacer notar que las gramáticas se pueden representar mediante el grafo de una máquina de estados o mediante una red de palabras (word network en inglés).

En particular, la mayoría de los sistemas ASR presentados en esta tesis utilizan un modelo de lenguaje basado en trigramas, en formato ARPA[2.4], y la figura 2.5 muestra

[2.4] Ver "ARPA Model Languages" en la página de la Universidad Carnegie Mellon: http://cmusphinx.sourceforge.net/wiki/sphinx4:standardgrammarformats

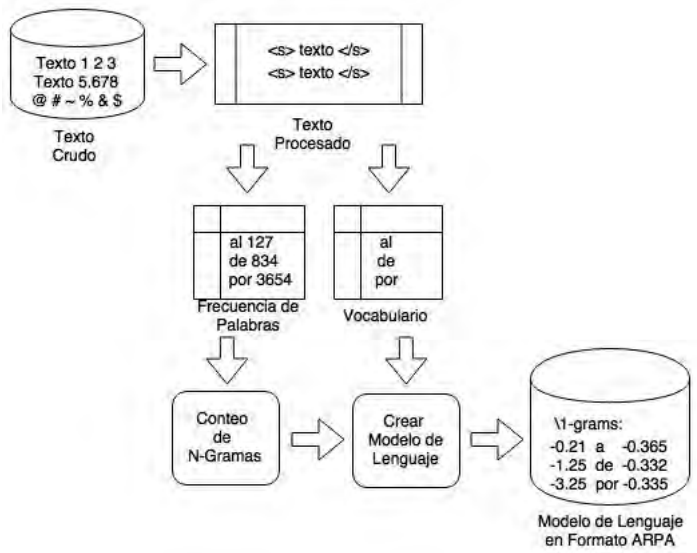

Figura 2.5: Proceso de creación del modelo de lenguaje.

el proceso de creación de este tipo de modelos, mediante los siguientes pasos:

- **Procesado del texto crudo**: El texto inicialmente elegido para la creación del modelo de lenguaje debe ser transformado a un archivo de texto plano, en dónde se lea una sola oración por línea y sin signos de puntuación.
- **Frecuencia de palabras**: Se debe obtener un archivo con la frecuencia de aparición de las palabras en el texto procesado.
- **Vocabulario**: Se deben elegir las palabras que serán parte del vocabulario del sistema de reconocimiento, ya sea de forma manual, o de forma automática. Una manera de hacerlo de forma automática es eligiendo, por ejemplo, las palabras que tengan una frecuencia de aparición mayor a cierto número m.
- **Conteo de n-gramas**: Se obtiene la frecuencia de aparición de los unigramas, luego de los bigramas, luego de los trigramas y así sucesivamente hasta llegar a los n-gramas que se desee.
- **Creación del modelo de lenguaje**: Después de calcular todo lo anterior, se escribe el resultado en un archivo de texto plano, para que sea leído por el sistema ASR. Un formato muy utilizado para n-gramas es el formato ARPA.

A grandes rasgos, el formato ARPA se divide en secciones $\backslash 1-grams$, $\backslash 2-grams$, $\backslash 3-grams$, etcétera. En cada sección hay una lista de n-gramas ordenada alfabéticamente. A la izquierda de cada n-grama se encuentra el logaritmo (normalmente base

Figura 2.6: Módulo de extracción de atributos de un sistema ASR.

10) de la probabilidad de aparición de ese n-grama. A la derecha puede haber o no, una "probabilidad de retroceso" también logarítmica.

Una "backoff probability" o "probabilidad de retroceso" significa que si la ocurrencia de un n-grama no sobrepasa cierto umbral k, se hará entonces un "retroceso" y se calculará su probabilidad con base en la probabilidad del $(n-1)-grama$ correspondiente.

Para una mayor comprensión, el lector puede acudir al apéndice C.

2.5 Componentes básicos de un sistema ASR

En un sistema de reconocimiento de voz se pueden distinguir tres módulos fundamentales, los cuales son: el de extracción de atributos, el de entrenamiento y el de reconocimiento. En esta sección se proporciona una explicación detallada del funcionamiento de los mismos, así como del papel que desempeñan dentro del sistema.

2.5.1 El módulo de extracción de atributos

Tal y como se discute en la sección 2.2, el proceso de parametrización de la señal de voz consiste en obtener sus tramas, para luego extraer información de estas. A este proceso se le conoce como "extracción de atributos".

Un ejemplo clásico, es el proceso de obtención de los vectores de coeficientes MFCC. Esta parametrización es utilizada por todos los sistemas presentados en esta tesis.

La figura 2.6, basada en (Seltzer, 1999), describe detalladamente todos los pasos intermedios para crear vectores de atributos MFCC. Se distinguen los pasos siguientes:

- **Filtro de pre-énfasis**: El filtro de pre-énfasis es un filtro FIR que se aplica a la señal de entrada y que ayuda a resaltar las altas frecuencias.
- **División en segmentos**: La señal de voz es dividida en pequeños segmentos que pueden ir desde los 5 hasta los 50 milisegundos.
- **Ventaneo**: En general, se aplica una ventana cosenoidal para suavizar el espectro en los bordes del segmento.
- **Espectro de potencia**: Se calcula el espectro de potencia mediante una DFT, aplicada a todo el segmento ventaneado.
- **Espectro mel**: Se calcula el "espectro mel" multiplicando el espectro de potencia por cada filtro triangular del banco de filtros de mel.
- **Coeficientes mel cepstrum**: Se aplica una transformada coseno discreta (DCT) al logaritmo natural del espectro Mel para obtener los coeficientes mel cepstrum[2.5].

El proceso de extracción de atributos se aplica, tanto en la ronda de entrenamiento para el corpus de entrada, como en la ronda de reconocimiento, a los archivos de audio que se quiere reconocer. Además, este proceso ha llegado a ser tan eficiente, computacionalmente hablando, que se puede aplicar también a sistemas que hagan reconocimiento en vivo (live decoding).

2.5.2 El módulo de entrenamiento

En un sistema ASR, el módulo de entrenamiento es el encargado de implementar el paradigma de reconocimiento elegido para el sistema, así como de administrar los audios y las transcripciones del corpus de entrada, para generar al final un modelo acústico.

En la figura 2.7, se muestra el diagrama a bloques del módulo de entrenamiento, en el cual se advierte un conjunto de elementos de entrada, constituido por:

- **Corpus de entrenamiento**: Contiene grabaciones de voz, cada una con una transcripción asociada.
- **Lista de fonemas**: La lista de fonemas debe provenir de un alfabeto fonético; no obstante, dependiendo de las condiciones de diseño del sistema, podría no ser necesario que todos los elementos del alfabeto fonético en cuestión estén en la lista de fonemas. Sin embargo, lo más conveniente es que todos los elementos de la lista de fonemas aparezcan al menos una vez en el diccionario de pronunciación.

[2.5] Un dato curioso es que la palabra "cepstrum" es un anagrama de la palabra inglesa "spectrum" o "espectro" en español.

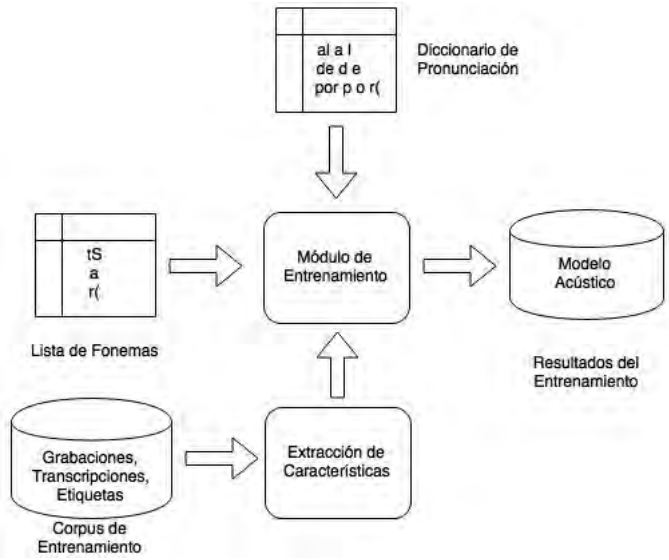

Figura 2.7: Requerimientos para la creación de un modelo acústico.

- **Diccionario de pronunciación**.

2.5.3 El módulo de reconocimiento

El módulo más importante es el de reconocimiento. Tal y como ocurre con el módulo de entrenamiento, el módulo de reconocimiento también necesita que los audios que se quiere reconocer, pasen por el módulo de extracción de atributos. Luego, comparando esos vectores de atributos con el modelo acústico, se logra encontrar la equivalencia que guardan con los elementos de la lista de fonemas.

De esta manera es como el reconocedor entrega un tren de fonemas que, con ayuda del modelo de lenguaje y del diccionario de pronunciación, es mapeado a palabras, que luego forman las frases reconocidas. La figura 2.8, muestra precisamente el diagrama a bloques de este proceso.

Ahora bien, en algunos sistemas ASR, el módulo de reconocimiento puede trabajar en dos modos: reconocimiento por lotes, también conocido como “reconocimiento en modo batch” o “batch decoding”; y reconocimiento en vivo, también conocido como “reconocimiento en modo live” o “live decoding”.

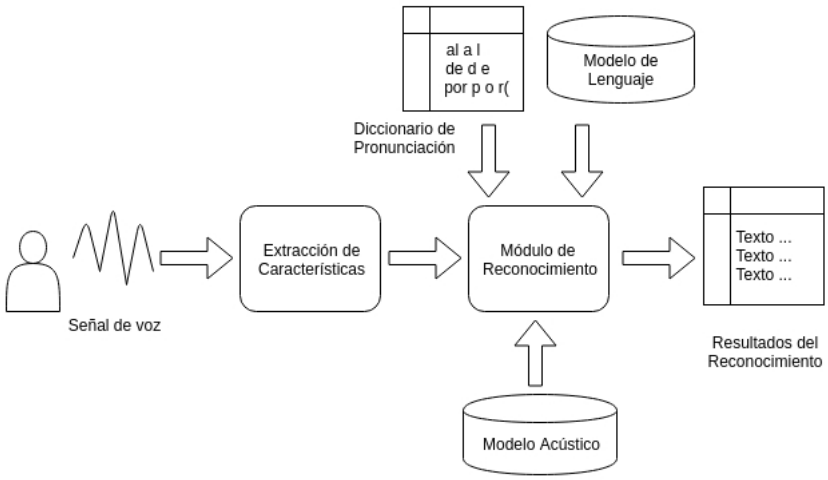

Figura 2.8: Módulo de reconocimiento de un sistema ASR.

2.6 Software libre para reconocimiento de voz

En esta sección se presentan cuatro de los sistemas de reconocimiento de voz más conocidos y utilizados de la última década. Todos están basados en el paradigma de los modelos ocultos de Markov y utilizan los MFCCs como parametrización de la señal de entrada.

Finalmente, otra de las características que guardan en común estos sistemas es que son de código abierto y cuentan con una licencia que permite hacerles modificaciones para todo tipo de fines.

2.6.1 CMU-Sphinx

El proyecto CMU Sphinx o simplemente Sphinx, es el nombre que se utiliza para denominar a un conjunto de reconocedores de voz desarrollado en la Universidad de Carnegie Mellon (CMU).

La primera versión de Sphinx, creada por el Dr. Kai Fu Lee a finales de los 80's (y que ahora es utilizada solamente como referencia histórica), consistía en un reconocedor de voz continua, independiente del hablante y basado en el uso de modelos ocultos de Markov discretos, así como de un modelo de lenguaje estático basado en n-gramas (Lee et al., 1990).

A partir de la década de los 90's comenzaron a aparecer nuevas versiones. En la actualidad existe todo un conjunto de versiones de Sphinx para diferentes propósitos, así como herramientas especializadas que permiten a los usuarios desarrollar una amplia gama de aplicaciones. A continuación se hace un listado de las versiones de Sphinx y

sus herramientas.

- Sphinx2: Es un reconocedor rápido, escrito en lenguaje C y desarrollado por Xuedong Huang. Se enfoca principalmente a aplicaciones de tiempo real para diálogo con el usuario. Incorpora algunas funcionalidades útiles como la de detección de inicio y fin de palabra, y modelo de lenguaje dinámico (Huang et al., 1993). Ha sido implementado en diversos productos comerciales. Una de las desventajas de esta versión es que sacrifica un poco la precisión en el reconocimiento a cambio de obtener mayor velocidad.

- Sphinx3: Adopta la representación continua de modelos de Markov y fue pensado para ser un reconocedor de alta precisión y no de tiempo real (Seltzer, 1999), sin embargo, debido a modificaciones recientes en sus algoritmos, se ha logrado que Sphinx3 sea un sistema "casi" de tiempo real. También está escrito en lenguaje C.

- Sphinx4: Es una versión totalmente escrita en Java que tiene el objetivo de proveer a los usuarios un marco de trabajo mucho más flexible para la investigación en el campo del reconocimiento de la voz. La empresa Sun Microsystems da soporte al desarrollo de Sphinx4 y contribuye con su experiencia en desarrollo de software comercial (Walker et al., 2004). Algunos de los nuevos logros que obtuvo Sphinx4 son:

  - Implementar adaptación del hablante (speaker adaptation).
  - Mejorar la manera en que los usuarios configuran el sistema.
  - Crear una interfaz gráfica de usuario para el diseño basada en grafos.

- PocketSphinx: Esta es una versión de Sphinx que puede ser utilizada en dispositivos móviles (por ejemplo, en procesadores basados en ARM) (Huggins-Daines et al., 2006). Actualmente PocketSphinx se encuentra ampliamente soportado por los miembros del proyecto Sphinx y presenta la característica de manejar aritmética de punto fijo.

- SphinxTrain: Es una herramienta creada para Sphinx3 que consiste en una suite de programas destinados a la obtención del modelo acústico.

- SphinxBase: Es un conjunto de librerías creadas para Sphinx3 que son requeridas para diversos procesos.

Sphinx es una buena opción como herramienta de reconocimiento de voz. De hecho, es una de las más completas que existe, sin mencionar que es de código abierto y maneja un tipo de licencia que no obliga a los usuarios distribuir sus aportaciones propias, aunque sean para fines comerciales (Chan et al., 2007).

Otro de los aspectos benéficos de Sphinx es que es una herramienta tan popularizada, que existe toda una variedad de herramientas y de recursos facilitados tanto por usuarios en todo el mundo, como por empresas.

Todas las versiones de Sphinx están basadas en el uso de modelos ocultos de Markov y un aspecto interesante al respecto para los investigadores, es que Sphinx maneja el cálculo de las GMM y el proceso de búsqueda por separado, lo que en esencia permite manipular unas partes de Sphinx sin afectar otras.

El modularismo de Sphinx también se extiende a sus algoritmos de entrenamiento, en los cuales es fácil elegir una configuración determinada de procesos y cambiarla.

2.6.2 HTK

Es un software portable, creado en el año de 1993 en la Universidad de Cambridge, Inglaterra. HTK fue creado para elaborar y manipular modelos ocultos de Markov. Inicialmente, fue diseñado para ser usado en el campo del reconocimiento de la voz, aunque hoy en día, ha sido utilizado en otras numerosas aplicaciones como por ejemplo: síntesis del habla, alineado fonético, reconocimiento de caracteres y reconocimiento de secuencias de ADN (Young et al., 2006).

HTK consiste en un conjunto de librerías y herramientas disponibles en lenguaje C. Todas estas herramientas proveen un conjunto de sofisticadas facilidades para el análisis de voz, entrenamiento de modelos ocultos de Markov y análisis de resultados, entre otras más.

En esta tesis se presentan dos sistemas de reconocimiento de voz diseñados por el autor y que están basados en HTK. Los cuales son: El HTK2SPHINX-CONVERTER (Hernández-Mena & Herrera-Camacho, 2015a) (ver apéndice D), que implementa reconocimiento de voz basado en gramáticas, y el HTK-BENCHMARK (ver apéndice E), que lo hace mediante el uso de un modelo de lenguaje basado en 3-gramas, tal y como lo hace el CMU-Sphinx3.

2.6.3 Julius

Julius[2.6] es una herramienta para el reconocimiento de voz continua, de código abierto, de alto desempeño, para vocabularios amplios (del orden de miles de palabras) y diseñado para la investigación por parte de la Universidad de Kyoto, Japón. Está basado en trigramas y modelos ocultos de Markov dependientes del contexto. Julius es capaz de hacer el reconocimiento en tiempo real en la mayoría de las computadoras personales actuales (Lane, 2001).

[2.6]El sitio web de Julius es: http://julius.osdn.jp/en_index.php

Julius fue desarrollado como parte de un sistema gratuito de origen japonés en 1997, y a partir del año 2000 fue retomado por el "continuous speech recognition consortium (CSRC)" de Japón hasta el 2003. Actualmente, el proyecto Julius está a cargo del Instuto de Tecnología de Nagoya, Japón.

El sistema Julius contempla únicamente el módulo de reconocimiento, siendo necesario crear sus modelos acústicos por medio del sistema HTK. Julius es capaz de hacer reconocimiento en modo batch o en modo live.

En esta tesis se diseña el sistema llamado HTK2SPHINX-CONVERTER (ver apéndice D) que implementa también el reconocimiento de voz en vivo basado en Julius.

2.6.4 Kaldi

Kaldi es un software de reconocimiento de voz muy reciente. Fue presentado por primera vez en un taller de la IEEE llamado "workshop on automatic speech recognition and understanding", en el año 2011(Povey et al., 2011), y desde entonces ha tenido un enorme auge, tanto así, que ha desplazado a otros reconocedores clásicos como HTK y CMU-Sphinx.

Kaldi es un sistema de reconocimiento de voz escrito en C++, inspirado en el sistema HTK. Tiene como principal característica el tener código de programación flexible, moderno, fácil de modificar y extender. Las características adicionales del sistema Kaldi son:

- Implementa de manera nativa máquinas de estados finitos.
- Incluye librerías para acelerar el cálculo numérico.
- Sus algoritmos han sido escritos de la manera más genérica posible, en pro de la flexibilidad de los usuarios.
- Cuenta con una licencia de código abierto "Apache 2.0", que es una de las menos restrictivas.
- Cuenta con repositorios unificados para descargar actualizaciones de software, y en cuanto a los datos, trabaja con muchos de los corpus encontrados en el Linguistic Data Consortium (LDC).

El proyecto Kaldi posee buena documentación en línea.

En este trabajo se realizaron experimentos de reconocimiento con este sistema. Se utilizó su clasificador HMM, aunque Kaldi ya cuenta con un clasificador basado en deep learning.

Capítulo 3

Fonética y fonología

La fonética y la fonología son dos ramas de la lingüística que están estrechamente ligadas con las tecnologías del lenguaje.

En artículos antiguos sobre fonética, como son los casos de (Fischer-Jorgensen, 1952) y (Twaddell, 1952), se percibe cómo los análisis de fonemas y alófonos de una cierta lengua podrían servir hoy en día a las tecnologías del lenguaje. Es decir, que muchos de los problemas que algunas tecnologías del habla necesitaban resolver, han sido resueltos por la fonética y la fonología desde hace largo tiempo.

Ahora bien, en el campo particular del reconocimiento de voz, la lingüística nos ayuda a darle expresividad a los vectores MFCC, a los modelos de Markov y a las redes neuronales. Por lo tanto, es con el conocimiento lingüístico (y con otros más) que podemos hacer abstracciones más profundas para tratar de explicar la realidad que subyace en la señal de voz.

Por lo anterior, en este capítulo se aborda una explicación concisa y breve sobre los valiosos conocimientos que la fonética y la fonología aportan al campo del reconocimiento automático de voz. Al final del capítulo se presentan algoritmos para elaborar transcripciones fonéticas y fonológicas, en el alfabeto Mexbet.

3.1 El aparato fonador

El aparato fonador está dividido en tres partes fundamentales: las cavidades infraglóticas, la cavidad glótica o laríngea y las cavidades supraglóticas (Martínez Celdrán, 1984). La figura 3.1 muestra al aparato fonador completo, junto con el aparato respiratorio. El sufijo "glótica", de cada una de estas partes indica de alguna manera, que la glotis es el centro del aparato fonador, aunque en realidad no es el elemento más importante de este. De hecho, puede decirse que los tres elementos mencionados son necesarios para la producción de la voz.

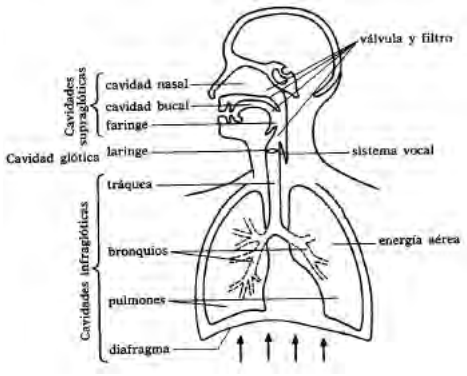

Figura 3.1: Aparato fonador y aparato respiratorio.[3.1]

3.1.1 Las cavidades infraglóticas

Las cavidades infraglóticas están compuestas básicamente del diafragma, los pulmones, los bronquios y la tráquea. Lo importante en esta zona es la producción del flujo de aire que permite la fonación, por medio de la respiración. Esta se compone de dos tiempos o fases: el primero es la inhalación o aspiración y el segundo es la espiración o exhalación (Martínez Celdrán, 1984).

En la mayoría de las lenguas del mundo se utiliza la corriente de exhalación para producir la fonación, aunque también es posible producir sonido con aire inhalado. Sin embargo, las lenguas que aprovechan en alguna medida los sonidos aspirados, producen la mayor parte de sus elementos fónicos con aire espirado (Martínez Celdrán, 1984).

El diafragma es un tabique muscular que se sitúa entre la cavidad torácica y los órganos del vientre. Tiene forma de cúpula y, al comprimirse las fibras del músculo, el diafragma se aplana y la caja torácica se ensancha. Al relajarse las fibras se provoca un estrechamiento de la cavidad torácica, lo que produce un impulso de aire (Martínez Celdrán, 1984).

Los pulmones son únicamente bolsas que sirven para almacenar el aire. En la aspiración se llenan y en la exhalación lo expulsan. Sin embargo, en la respiración normal nunca quedan completamente vacíos o llenos. Los bronquios y la tráquea son solamente tubos de salida y entrada de aire, por lo tanto, su función es la conducción del mismo (Martínez Celdrán, 1984).

[3.1] Imagen extraída de (Martínez Celdrán, 1984)

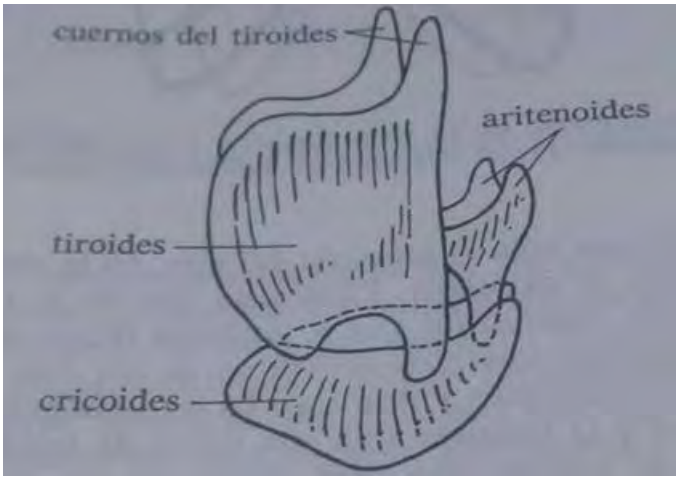

Figura 3.2: Posición de los cartílagos laríngeos.[3.2]

3.1.2 La cavidad glótica o laríngea

A la cavidad glótica se le llama también cavidad laríngea, porque es la laringe su elemento constitutivo. La laringe es un tubo móvil cartilaginoso situado al final de la tráquea que puede ascender y descender, siendo su posición habitual la parte inferior (Martínez Celdrán, 1984).

La laringe se compone de cuatro cartílagos fundamentales que van de la parte inferior de la laringe a la parte superior de la misma, en este orden: el cricoides, el tiroides, el aritenoides y los cuernos del tiroides. El tiroides está situado por encima del cricoides, en la parte anterior a la laringe, y se conoce con el nombre de "manzana de Adán". En su parte interior central, sujeta los extremos anteriores de las cuerdas vocales. Los distintos cartílagos de la laringe poseen una gran movilidad, gracias a la cual, las cuerdas vocales pueden tener todo tipo de movimiento (Martínez Celdrán, 1984). La figura 3.2 muestra una imagen de la posición de los cartílagos laríngeos.

Cuando se produce la fonación, el cartílago tiroides es atraído hacia adelante, de modo que las cuerdas vocales se alargan y se tensan, mientras los aritenoides giran ligeramente, estrechando la glotis (Martínez Celdrán, 1984). Las cuerdas vocales son dos músculos que están situados en la parte parte inferior de las bandas ventriculares como se aprecia en la figura 3.3.

La glotis, es por tanto, el espacio triangular que dejan las cuerdas vocales. El nombre de cuerdas vocales puede confundir porque no son en realidad cuerdas, sino músculos que además, no vibran como lo hacen las cuerdas de una guitarra. Sucede que al querer emitir voz, la tensión de los aritenoides hace que los repliegues vocales se junten, impidiendo el paso del aire. Sin embargo, el empuje de ese aire que sube de los pulmones

[3.2] Imagen extraída de (Martínez Celdrán, 1984)

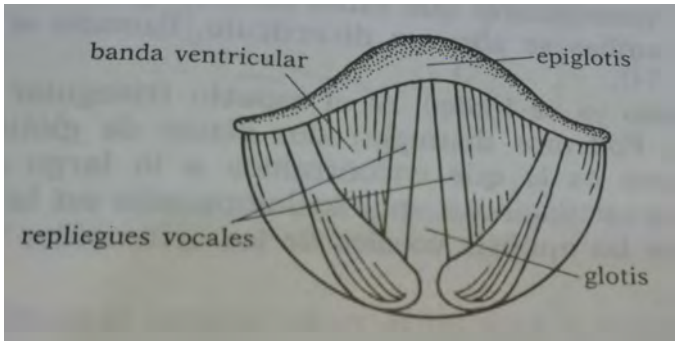

Figura 3.3: Las cuerdas vocales.[3.3]

es suficiente para abrir los repliegues momentáneamente puesto que su propia tensión hace que vuelvan a cerrarse. Este movimiento de abertura y cierre es lo que ocasiona la vibración. Esta vibración produce lo que se conoce como un tono laríngeo, compuesto de una frecuencia fundamental y múltiples armónicos con una intensidad muy débil (Martínez Celdrán, 1984).

3.1.3 Las cavidades supraglóticas

Cuando el aire, ya en vibración, sale de la laringe, se encuentra en primer lugar con la cavidad faríngea, situada entre la pared posterior llamada faríngea y la raíz de la lengua en la parte anterior. Por arriba, está la úvula y la entrada a dos cavidades diferentes: la nasal y la bucal (Martínez Celdrán, 1984).

En la boca, desde el interior al exterior, está la úvula, que es el apéndice final del paladar blando o velo del paladar, después sigue lo que es el paladar duro, los alvéolos y los dientes incisivos superiores. Estos últimos tres elementos son órganos pasivos en el proceso de fonación, ya que no tienen ninguna movilidad. Por otro lado, la lengua, en colaboración con los labios y el maxilar inferior, son órganos muy activos junto con el velo del paladar. La figura 3.4 muestra a detalle, las diferentes partes que conforman la boca y que tienen que ver directamente con el proceso de fonación (Martínez Celdrán, 1984).

La figura 3.4 también muestra las partes de la lengua, las cuales se dividen en: el ápice, que no es otra cosa que la punta de la lengua, y el dorso, que entra en contacto con la región velar y palatal para formar cierto tipo de fonemas oclusivos y aproximantes (Martínez Celdrán, 1984).

[3.3] Imagen extraída de (Martínez Celdrán, 1984)

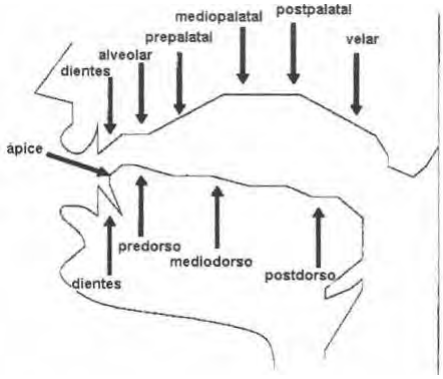

Figura 3.4: Regiones linguales.[3.4]

3.2 Fundamentos de fonética

La fonética es la rama de la lingüística que estudia los sonidos del habla desde el punto de vista de su realización acústica y se interesa por la duración, la amplitud, la frecuencia y por otros parámetros medibles de la señal de voz; así como de la posición de los órganos que producen los sonidos del habla (rasgos articulatorios) (Odden, 1996).

Una diferencia importante entre fonología y fonética es que la primera trata a los sonidos del habla como unidades prototípicas e ideales llamadas fonemas, y la segunda trata de saber cómo es que esos fonemas varían de persona a persona. Las variantes de fonemas prototípicos son conocidas con el nombre de alófonos.

El estudio de la fonética puede adoptar tres diferentes perspectivas:

- **Fonética Articulatoria**: Estudia los sonidos desde el punto de vista de su producción, por parte del emisor del mensaje.
- **Fonética Acústica**: Estudia las modificaciones establecidas en el medio elástico en que se transmiten los sonidos articulados por el emisor.
- **Fonética Perceptiva**: Estudia los sonidos desde la óptica del receptor del mensaje.

Ahora bien, uno de los instrumentos más importantes para la fonética es el espectrograma. Un espectrograma se considera una representación en tres dimensiones. Por lo regular se pone el tiempo en su eje horizontal, la frecuencia en su eje vertical y por medio de un código de colores o escala de grises, se representa la energía del contenido frecuencial de la señal en cada punto de la gráfica. Si se usa una escala de grises, resultaría que los colores más obscuros indicarían una mayor energía. La figura 3.5 muestra

[3.4] Imagen extraída de (Celdrán & Planas, 2007)

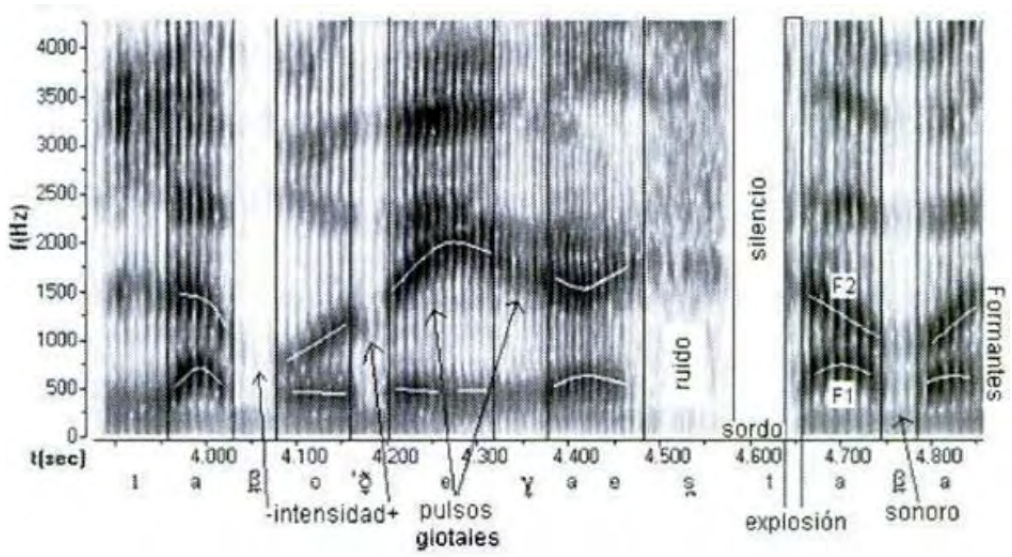

Figura 3.5: Espectrograma con información fonética.[3.5]

un espectrograma con indicaciones del tipo de información de carácter fonético que se puede obtener de este (Celdrán & Planas, 2007).

El espectrograma es útil para obtener información fonética relevante como:

- Los formantes.
- Transiciones debidas a las influencias consonánticas.
- Explosión de la oclusiva.
- El tiempo de inicio de la sonoridad o VOT (voice onset time).
- Fases de silencio.
- Barra de sonoridad o su ausencia.
- Pulsos glotales.
- Ruido de las fricativas.
- Duración de los eventos del habla.

Otro concepto importante en fonética son los segmentos. Los segmentos fonéticos son las unidades lingüísticas mínimas del habla que se aproximan a las letras del alfabeto. En un espectrograma los segmentos muestran el problema de la "no invariancia"

[3.5]Imagen extraída de (Celdrán & Planas, 2007)

de la voz, esto es, que una misma característica acústica puede definir varios segmentos del habla del mismo modo que un segmento puede a su vez, ser definido por diferentes contextos fonéticos. De hecho, la segmentación aparece al intentar extraer sonidos o fonemas discretos a partir de la cadena hablada (Gotor, 2001).

Los segmentos fonéticos pueden ser clasificados como obstruyentes, sonantes y vocálicos (Celdrán & Planas, 2007; Martínez Celdrán, 1984). Sin embargo, existen otras clasificaciones de estos segmentos dependiendo de su modo y lugar de articulación.

3.2.1 Segmentos fonéticos obstruyentes

Los segmentos fonéticos obstruyentes son aquellos que oponen algún obstáculo a la salida del aire, aunque sea mínimo. Puede haber tanto obstruyentes sordos cuando no hay vibración de las cuerdas vocales, como sonoros cuando si la hay. En esta categoría además, entran las oclusivas (se incluyen en ellas las africadas), fricativas y aproximantes de tipo espirante (Celdrán & Planas, 2007), cuyas definiciones son:

- **Oclusivas**: Son las que impiden totalmente la salida del aire.
- **Fricativas**: Oponen una gran resistencia a la salida del aire por su estrechamiento en el punto de articulación, pero no la impiden por completo.
- **Aproximantes espirantes**: O no poseen el mismo grado de estrechamiento que las fricativas o no ofrecen gran resistencia a la salida del aire. Por otra parte, su alternancia en la lengua con las oclusivas o su posible realización fricativa en pronunciaciones enfáticas ofrece evidencias claras para su inclusión en la categoría de obstruyentes.

3.2.2 Segmentos fonéticos sonantes

Los segmentos fonéticos sonantes son los sonidos que aún presentando algún obstáculo en el tracto vocal, dejan paso libre a la salida del aire y, por regla general, se realizan con vibración de las cuerdas vocales, esto es, son sonidos sonoros (Celdrán & Planas, 2007) y pueden clasificarse en:

- **Nasales**: En estas, el aire queda retenido en algún punto de la boca, pero el aire sale de forma totalmente libre por la nariz.
- **Laterales**: Obstruyen a un lado de la boca o su centro, pero por los lados abiertos el aire fluye sin problemas.
- **Róticos**: Estos sonidos se desarrollan de forma vibrante, aproximante o fricativa, todos ellos con un cierto grado de obstáculo, pero con un elemento de tipo vocálico que permite el escape del aire ampliamente.

- **Semivocales**: Poseen un grado mayor de abertura que las aproximantes espirantes, pero menor que las vocales más cerradas, por eso son clasificadas también dentro de esta categoría.

3.2.3 Segmentos fonéticos vocálicos

Los segmentos fonéticos vocálicos son los segmentos más abiertos. En su producción no hay ningún tipo de obstáculo importante a la salida del aire y pueden aparecer de forma aislada, sin el apoyo de ningún otro segmento; por lo tanto, se constituyen en el sostén de las otras clases de sonidos, principalmente de las obstruyentes y en menor medida de las sonantes. Esto es sobre todo verdad en el español, razón por la cual el único segmento que puede aparecer en el núcleo de la sílaba y constituir una sola sílaba por sí misma es la vocal; ningún otro segmento posee esta capacidad en esta lengua (Celdrán & Planas, 2007).

3.2.4 Suprasegmentos

Se denomina suprasegmento a un tipo de información que está en la señal de voz, y que no está en directa relación con la identificación de las cualidades de un segmento (vocal o consonante), sino que más bien destaca un elemento por sobre otro (como el acento) o bien produce un efecto más global en un enunciado (como la melodía de la frase). Una misma frase puede ser dicha como afirmación o como pregunta. En ese caso, los segmentos serían los mismos. La diferencia entre una modalidad y otra estaría dada por los aspectos suprasegmentales de ambas emisiones (Celdrán & Planas, 2007). Por ejemplo:

```
¿Vendrá mañana?

Vendrá mañana.
```

De manera inversa, dos frases muy distintas en cuanto a contenido pueden tener básicamente la misma estructura melódica. Por ejemplo, estos dos versos del famoso "soneto a Cristo crucificado":

```
el cielo que me tienes prometido,

ni me mueve el infierno tan temido
```

Aquello que diferencia a las dos primeras frases y aquello que es semejante en las dos segundas, es lo suprasegmental. El estudio de los temas suprasegmentales se llama prosodia. Las vocales juegan un papel fundamental en relación con esta información

suprasegmental. Los factores que permiten hacer análisis prosódico son tres (Celdrán & Planas, 2007). Desde el punto de vista acústico se denominan:

- Intensidad
- Duración
- Frecuencia fundamental, o f0.

3.3 Fundamentos de fonología

La fonología es la rama de la lingüística que estudia los sonidos del habla (fonemas) y los considera como un sistema abstracto (Odden, 1996), (Salcedo, 2010). Esto significa que en fonología, los fonemas se representan sólo de forma ideal o prototípica. De hecho, la rama de la lingüística que trata de representar los sonidos del habla de la forma más precisa posible es la fonética (Odden, 1996). Por tanto, mientras que la fonética estudia la naturaleza acústica y fisiológica de los sonidos del habla (alófonos), la fonología describe el modo en que los sonidos funcionan en una lengua en particular o en las lenguas en general, en un nivel abstracto (fonemas).

En fonología, uno de los elementos más importantes de estudio, además del fonema, es la sílaba. De hecho, al igual que en fonética, los fonemas pueden comportarse de formas distintas, dependiendo de su lugar en la sílaba. También es común que las transcripciones fonológicas se hagan marcando explícitamente la división silábica y el acento. A continuación se exponen las diferentes partes de la sílaba.

3.3.1 Características de la sílaba española

Los fonemas españoles se pueden clasificar basándose en su capacidad de poder formar sílabas o no, en silábicos (vocales) y no silábicos (consonantes). En la lengua española las consonantes nunca pueden formar un núcleo silábico, mientras que las vocales pueden ser núcleos silábicos y márgenes silábicos, como en el caso de los diptongos y triptongos (Quilis, 1993).

Ahora bien, una sílaba, según el número de fonemas que la constituyen, puede ser: monofonemática, cuando sólo tiene un fonema, como en las dos primeras sílabas de la palabra “o . í . do”; o polifonemática cuando tiene más de un fonema, como en las sílabas de la palabra “ca . sa” (Quilis, 1993). A continuación se muestran las estructuras silábicas posibles en el idioma español. La letra “C” significa consonante y la letra “V” significa vocal:

1. CV

2. CVC
3. V
4. VC
5. CCV
6. CCVC
7. VCC
8. CVCC
9. CCVCC

Es importante hacer notar que en la lengua española hay una tendencia clara hacia la sílaba abierta y en especial a la estructura CV, que es la más recurrente de todas (Quilis, 1993).

3.3.2 Núcleo de sílaba

Las vocales "a", "e" y "o" son siempre núcleo de sílaba. Por ejemplo, la palabra "océano" tiene cuatro sílabas. Las vocales "i" y "u" son más complicadas, es decir, son siempre núcleo de sílaba si no van inmediatamente junto a otra vocal. Por ejemplo, la palabra "incinerado" tiene cinco sílabas (Quilis, 1993).

Cuando van junto a otra u otras vocales, son núcleos de sílaba si son tónicas, en cuyo caso siempre llevan tilde (Quilis, 1993). Por ejemplo:

- **Fiáis. Tiene sólo un núcleo**: La vocal "a" es siempre núcleo de sílaba; las dos "íes" van inmediatamente junto a una vocal y no se pronuncian tónicas, luego no son núcleos de sílaba. En consecuencia, sólo hay un núcleo de sílaba y la palabra es monosilábica.
- **Huíais. La "a" es siempre núcleo**: La última "i" va inmediatamente junto a una vocal y no es tónica, por lo que no es núcleo; la "u" va inmediatamente junto a una vocal y no es tónica: no es núcleo; la primera "i" es tónica y va inmediatamente junto a vocales: luego es núcleo de sílaba y siempre con tilde, aún en contra de la regla general. La palabra tiene dos sílabas y se acentúa gráficamente.

3.3.3 Coda silábica

Coda es un término lingüístico que se refiere a la consonante final de una sílaba. En lingüística, la coda es la consonante o el grupo consonántico (agrupación de consonantes) en posición postnuclear dentro de una sílaba, es decir, después de la vocal nuclear. La coda, conjuntamente con el núcleo, se denomina rima y no es totalmente necesaria en una sílaba (Quilis, 1993). Ejemplos de codas son:

- "n" en "pan"
- "l" en "sal"
- "s" en "es"

Rima es un término lingüístico que denomina, dentro de una sílaba, la vocal nuclear y la coda o codas, si procede. Ejemplos de rimas:

- "al" en sal
- "il", "a" y "o" en "Bilbao"

3.3.4 Sílabas abiertas y cerradas

En lingüística, una sílaba abierta o libre es aquella que termina en vocal, es decir, que termina en núcleo silábico (Quilis, 1993). Por ejemplo, todas estas palabras están compuestas de sílabas abiertas solamente:

```
do.mi.na.do
e.na.je.na.do
a.ma.ri.llo
```

En contraparte, una sílaba cerrada o trabada es aquella que termina en una o más consonantes (Quilis, 1993). Por ejemplo, todas estas palabras están compuestas de sílabas cerradas únicamente:

```
trans.por.tes
per.ver.sión.
in.ver.tir
```

En el caso de la sílaba "tes", en la palabra "transportes", se dice que es una sílaba trabada por "s", así como "ver", en "perversión" es una sílaba trabada por "r".

3.4 Los rasgos articulatorios

El tono de la voz lo determinan las cuerdas vocales. Se utilizan la faringe, los labios y la lengua para modular el aire de tal manera que se puedan formar los fonemas, que luego se constituyen en palabras (Martínez Celdrán, 1984). Todos estos órganos son conocidos como órganos articulatorios. Por tanto, los rasgos son aquellas características que nos ayudan a distinguir los diferentes sonidos del habla, y en particular, los rasgos articulatorios son aquellos que nos ayudan a hacerlo con base en lo que se conoce como punto de articulación (Stüker et al., 2003).

Con respecto al punto de articulación, Martínez Celdrán nos dice:

"Tradicionalmente, se ha dado gran importancia al concepto de punto de articulación, puesto que se pensaba que el lugar exacto de unión o aproximación de los órganos, en el momento de la pronunciación, era básico. No obstante, en el transcurso de las investigaciones experimentales del siglo XX se ha visto que no es tan principal como se creía, pues lo importante es la forma o el volumen del resonador, por lo que el punto es sólo un elemento más de la articulación" (Martínez Celdrán, 1984).

Este extracto proviene de un libro publicado por primera vez en el año de 1984. Así, el estudio de los rasgos articulatorios para mejorar el reconocimiento automático de voz data de finales de los 80's (Kirchhoff, 2000). Por lo tanto, aquí puede verse cómo ya desde esa época surge la necesidad de obtener mayor información de la señal de voz, para poder reconocerla con mayor precisión (Metze et al., 2005).

Uno de los mayores problemas que enfrenta el reconocimiento es la "no invariancia" de la voz, es decir, que la voz es producida de varias maneras distintas incluso por un mismo hablante y dependiendo del contexto (Andruski et al., 1994). Dos ejemplos clásicos de esto último podrían ser: el efecto Lombard (Lane & Tranel, 1971) que se da cuando los hablantes en entornos muy ruidosos tienden a subir el volumen de su voz para compensar el volumen del ruido del ambiente y poder seguir teniendo comunicación con sus interlocutores y, el conocido problema de los sistemas de diálogo telefónicos, que ocurre cuando una oración pronunciada por el usuario no es reconocida por el sistema y este le pide al usuario que la vuelva a decir, conforme esto ocurre el usuario se enoja más y más, modificando el tono y la velocidad de su voz, provocando que el sistema tenga aún más problemas para reconocerlo, hasta que finalmente se frustra y cuelga (Haas et al., 2005).

Así que, una buena manera de hacer frente a estos problemas es utilizando los rasgos articulatorios, ya sea de forma exclusiva (como en (Schultz & Wand, 2010)) o como fuente adicional de información para el reconocimiento (como en (Eide, 2001)), ya que de algún modo, los rasgos articulatorios son un tanto más inmunes a los cambios de tono, intensidad y volumen de la señal de voz.

Uno de los problemas con estos rasgos es que las mismas propiedades mecánicas

de los órganos articulatorios afectan no sólo los fonemas que intentan reproducir, sino también los fonemas que les anteceden y que les suceden, dependiendo del contexto en que cada fonema es pronunciado (Andruski et al., 1994). Esto último es debido a que los órganos articulatorios no pueden tener transiciones instantáneas en sus movimientos mecánicos naturales y es por eso que se ven obligados a "invadir" fonemas vecinos (Erler & Freeman, 1996).

3.5 Alfabetos fonéticos

Debido a que en el campo de la fonética existe la necesidad de registrar los sonidos del habla (fonemas y alófonos), existe también la necesidad del uso de alfabetos fonéticos. Un alfabeto fonético es un conjunto de símbolos que representa los sonidos del habla en determinado lenguaje. Al proceso de convertir una palabra de una representación ortográfica a una representación fonética se le llama "fonetizar", aunque vulgarmente, también se le conoce como "transcribir".

En el caso de la lengua española, se cuenta con el alfabeto fonético RFE, que es el acrónimo para "revista de filología española" fundada en 1914 en Madrid, España (Tomás, 1966). El alfabeto RFE fue publicado en 1915 (Tomás, 1915) y sigue siendo usado por la comunidad lingüística hasta nuestros días.

El Alfabeto Fonético Internacional (AFI), contiene los sonidos del habla de todas las lenguas del mundo y es considerado como un estándar entre la comunidad lingüística internacional. Sin embargo, el uso de estos alfabetos tiene desventajas en muchas de las áreas de las ciencias de la computación, por algo tan aparentemente trivial, como lo es la codificación de caracteres. Los alfabetos AFI, RFE y otros alfabetos fonéticos "clásicos" cuentan con símbolos que son imposibles de manejar en códigos de programación.

La solución a estos problemas es la creación de "alfabetos fonéticos computacionales ASCII" (AFCA). Algunos ejemplos de AFCA son el "speech assessment methodology phonetic alphabet" (SAMPA) (Wells, 1997), que fue diseñado para varios lenguajes, incluido el español (Llisterri & Mariño, 1993), el alfabeto WORLDBET que es una adaptación a ASCII de los símbolos del alfabeto AFI (Hieronymus, 1994) y el alfabeto OGIBET que fue creado por el Institute of Science and Technology (Lander & Metzler, 1994) y luego adaptado al español de México por el grupo Tlatoa (Serridge, 2000). Tlatoa es un grupo mexicano dedicado al desarrollo de tecnologías del habla de la Universidad de las Américas, Puebla, fundado en 1997 (Kirschning, 2001).

Otro buen ejemplo de un AFCA es precisamente el alfabeto Mexbet.

3.6 El alfabeto Mexbet

Mexbet es un AFCA especialmente diseñado para el español hablado en México. Está basado mayormente en el alfabeto WORLDBET, pero también un poco en el alfabeto OGIBET (Cuetara-Priede, 2004). La primera aparición de Mexbet en la literatura fue en (Uraga, 1999), como parte de una tesis de maestría. Luego, Mexbet ha sufrido actualizaciones y ha sido presentado en la literatura más de una vez (Uraga & Pineda, 2000), (Uraga & Pineda, 2002), (Pineda et al., 2009), (Hernández-Mena & Herrera-Camacho, 2014).

La versión actual de Mexbet apareció en el año de 2004 (Cuetara-Priede, 2004). Cuétara introdujo una serie de cambios y correcciones a la versión de Mexbet más reciente de aquel tiempo. Los datos con los que Cuétara hizo su estudio para actualizar Mexbet, vinieron del corpus DIMEx100 (Pineda et al., 2004).

No obstante, a pesar de la buena trayectoria que ha tenido este alfabeto fonético, Mexbet aún no ha sido adoptado en México como un alfabeto estándar para hacer reconocimiento de voz en español de México, como ocurre en el inglés con el alfabeto ARPAbet, que ha sido, y sigue siendo utilizado para diversas herramientas de reconocimiento de voz, como el diccionario de pronunciación de la Universidad Carnegie Mellon [3.6], el corpus TIMIT (Garofolo et al., 1993), o el reconocedor HTK (Young et al., 2006), entre muchas otras. El uso del alfabeto ARPAbet es muy extendido y argumentamos que puede considerarse como un alfabeto estándar para el reconocimiento automático de voz en el idioma inglés.

Por lo anterior, es claro que la adopción de un alfabeto estándar ayudaría al avance de la investigación en México, puesto que no se tendría que hacer una adaptación nueva de otro alfabeto cada vez, como ocurrió con el grupo Tlatoa, que adoptó OGIBET, para adaptarlo luego al español de México (Kirschning, 2001); o cuando se hizo una de las primeras adaptaciones del reconocedor CMU-SPHINX al español Mexicano en (Varela et al., 2003). Se hizo una adaptación de WORLDBET. También cuando se hace una versión mexicana del software CSLU Toolkit (Sutton et al., 1998), utilizado para facilitar el desarrollo de sistemas de diálogo, se hizo una adaptación diferente del alfabeto WORLDBET.

También existen existen ejemplos de productos desarrollados para el español de México, en donde ni siquiera se especifica un alfabeto fonético a utilizar. Tal es el caso de dos bases de datos para uso en procesamiento de voz, creadas recientemente, que únicamente cuentan con las transcripciones ortográficas de sus audios; se trata del corpus "West Point Heroico Spanish speech" (Morgan, 2006) que fue publicado por el "Linguistic Data Consortium" y el corpus VoCMex (Olguín-Espinoza et al., 2013) creado por un grupo de investigadores de Baja California, México, en 2012.

Solamente el corpus VOXMEX (Uraga & Gamboa, 2004) utilizó una versión de

[3.6] Ver: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Mexbet anterior a la del 2004. Por otro lado, aunque el corpus DIMEx100 sí utilizó la versión de 2004 en (Pineda et al., 2004), no se exponen sus reglas de transcripción con mucho detalle en ese artículo. Esa es quizá,una de las razones por las que ni Mexbet, ni ninguna otra adaptación de alfabeto fonético, es adoptada en México por la comunidad científica: porque raramente se difunden las reglas de transcripción de ese alfabeto u adaptación.

Por lo tanto, una de las aportaciones de esta tesis es difundir las reglas de transcripción de Mexbet con lujo de detalle y ejemplos; y hacer una ampliación de las mismas, para incluir a todos los fonemas y alófonos de la versión de 2004; agregando además, dos nuevos fonemas comunes en México, que provienen del náhuatl.

3.7 Reglas fonológicas de Mexbet

En esta sección se explica cómo realizar transcripciones a nivel fonológico de palabras en español usando el alfabeto Mexbet. Como antecedente, en la versión de 2004 se encuentran un total de 17 fonemas consonánticos y 5 fonemas vocálicos, para poder modelar el español hablado en el centro de México.

A las transcripciones de este nivel se las llama fonológicas, porque se componen únicamente de fonemas prototípicos. El nivel fonológico de Mexbet también recibe el nombre de nivel T22 o simplemente T22.

Para esta revisión de Mexbet se decidió incorporar dos fonemas que provienen del náhuatl, ya que son comunes en el español hablado en México, apareciendo sobre todo en nombres propios y topónimos. Finalmente, se decidió también considerar las vocales tónicas, que se marcan por medio del símbolo "_7".

Los nuevos fonemas son consonánticos: el fonema palatal fricativo sordo /S/, que es representado en la palabra "xoloitzcuintle" por la grafía "x", y el fonema lateral palatal /tl/, que sólo aparece a final de palabra, como en "popocatépetl" o "citlaltépetl". Sobre este último, es pertinente aclarar que su articulación es diferente a la secuencia de fonemas /t/ y /l/ en palabras que no provienen del náhuatl, como "atleta". De hecho, el fonema /tl/ constituye en sí mismo una sola sílaba.

Entonces, ya no es correcto designar a este nivel de Mexbet como "nivel T22", ya que los dos nuevos fonemas, más los cinco fonemas que indican las vocales tónicas, suman un total de 29 símbolos, por tanto, este nuevo nivel fonológico de Mexbet se le llamará "nivel T29".

La tabla H.1 en el apéndice H, muestra los símbolos Mexbet que representan a los fonemas del español del centro México. Aquí se puede encontrar información acerca del punto y del modo de articulación de cada uno. Esta información es usualmente útil en reconocimiento de voz, porque ayuda a agrupar modelos acústicos muy parecidos (ver "phone clustering" en (De Córdoba et al., 2002)), como lo requiere particularmente el

software HTK (Hernández-Mena & Herrera-Camacho, 2015a). Por otra parte, la tabla ubicada en el apéndice I muestra las equivalencias de los símbolos Mexbet y AFI.

Algunas de estas transformaciones "grafema-fonema", dependen del hecho de que muchos grafemas no representan los sonidos del habla de forma correcta, pero que sin embargo, deben ser tomados en cuenta por razones de tradición o históricas. Un ejemplo de esto es el grafema "h" que en español es usualmente mudo, o el grafema "z", que representa al fonema alveolar fricativo sordo /s/. Se puede encontrar mucha información acerca de las transformaciones necesarias para hacer transcripciones fonológicas en (Ríos Mestre, 1993), para el idioma español y en (Paul & Baker, 1992), para el idioma inglés.

Ahora bien, se dividirá el proceso de transcripciones fonológicas en cinco pequeñas partes, más sencillas, que llamaré: vocal tónica, pretranscripción, contextos del grafema "x", división silábica, y transcripción final.

Se asume que cualquiera de las palabras que se desee transcribir, estará en español y que no se tratará de aplicar estas reglas a: palabras en otros idiomas, como "ballet" o "capuccino"; abreviaturas, como "PRD" o "CCADET"; palabras inventadas, como "feisbookear". Además, se deberán aplicar con precaución a: nombres propios como "Xavier" o "Michael"; extranjerismos como "collage"; verbos conjugados como "acerco" y "acercó"; y en general, a palabras en dónde ocurra el fenómeno del desplazamiento acentual, como en "opera" y "ópera" o "médico", "medico" y "medicó".

Las transcripciones fonológicas son un paso previo a las transcripciones fonéticas. Estas últimas involucran una cantidad considerablemente mayor de reglas, debido a todos los contextos de cada alófono incorporado por Mexbet.

Es pertinente aclarar que para la realización automática de varios de los diversos pasos para la creación de transcripciones tanto fonéticas como fonológicas, se diseñaron una serie de funciones en lenguaje Python, a las que se agrupa bajo del nombre de "librería fonetica2". Estas funciones son evaluadas en el capítulo 6 y su funcionamiento es explicado a detalle en el apéndice G.

3.7.1 Paso 1. Vocal tónica

En cualquier transcripción fonológica se debe indicar cuál es la vocal tónica, para considerarla como una transcripción correcta y completa. Al proceso de indicar esta vocal se le llamará con el nombre de "acentuar" dicha palabra. En esta tesis, es una convención que las palabras que se desean transcribir estén en minúsculas y que su vocal tónica este indicada en mayúscula (ej. "elefAnte", "caEr", "botIn", "raUl").

El problema de acentuar una palabra se resuelve de manera instantánea y automática cuando esta cuenta con acento gráfico, pero cuando esto no es así, la vocal tónica puede determinarse de forma empírica, si se sabe cómo debe sonar la palabra que la contiene y en qué sentido se le usa. Por ejemplo, si se quiere acentuar la palabra "co-

cino" cuando proviene de la frase "yo cocino", la vocal tónica va en la "i": "cocIno", pero si proviene de la frase "el cocino", claramente se trataría de un error ortográfico por no llevar acento, pero por el contexto podemos deducir que la vocal tónica va en la segunda "o": "cocinO".

Para esta tesis se desarrolló una herramienta automática que sirve para acentuar palabras. Se trata de la función llamada "vocal_tonica()", que es parte de la librería fonetica2. Esta función toma en cuenta ciertas reglas empíricas para acentuar palabras que no cuenten con acento gráfico como, por ejemplo, todas las palabras terminadas en "ar", "er", "ir" deben ser acentuadas en la última sílaba, así como las palabras terminadas en "ad" (como libertad, paridad, calidad, etc). Toda vez que llegue una palabra que no presente algún caso especial contemplado por esta función, se acentuará siempre, por defecto, en la penúltima sílaba,

Ahora bien, el proceso de acentuar una palabra también resulta útil si después se le quiere dividir en sílabas, ya que la vocal tónica ayuda a distinguir diptongos de hiatos. Por ejemplo, si en la palabra "seria" el acento está en la "i": "serIa", como en "sería todo por hoy", resulta que la división silábica correcta es: "se.rí.a", es decir, que la "i" y la "a" forman un hiato; en cambio, si esa palabra proviene de la frase "estas muy seria", el acento va en la "e": "sEria", por lo tanto la división silábica correcta es: "se.ria", lo que significa que ahora la "i" y la "a" forman un diptongo. Otros ejemplos de casos como este podrían ser: "sabia" y "sabía"; "fío" y "fió"; "río" y"rió".

3.7.2 Paso 2. Pretranscripción

La tabla 3.1 muestra cómo realizar un conjunto de sustituciones de caracteres. Estas sustituciones constituyen el paso de "pretranscripción", para hacer transcripciones fonológicas en Mexbet. En la tabla también se puede ver que ciertos símbolos no ASCII, como "ñ" o "ü" debe ser sustituidos por "N" y "W", respectivamente. Algo similar se procede con los acentos gráficos de las vocales ("á", "é", "í", "ó", "ú") que deben ser sustituidas por la misma vocal pero en mayúsculas.

En general, en este punto del proceso de transcripción se considera que las vocales en mayúscula son vocales tónicas, procedan o no de una vocal con acento gráfico.

Algo que tienen en común las sustituciones de la tabla 3.1 es que pueden ser muy problemáticas para un algoritmo automático porque representan casos singulares en los que un grupo de grafemas (morfema) debe ser tratado como un solo elemento. Tal es el caso del morfema "rr", que ocurre en la palabra "carro". Si a "carro" se le aplican las reglas de división silábica discutidas en el paso anterior, su división silábica tendría que ser: "car.ro", lo cual es obviamente un error. Es por eso que la "rr" primero se sustituye por una "R" para que la división silábica se pueda hacer correctamente: "ca.Ro". El grafema "R" se sustituye al final por el fonema que designa a la vibrante múltiple /r/.

Otras transformaciones de ciertos grupos de grafemas encadenados (o morfemas,

Tabla 3.1: Transformaciones necesarias para el proceso de pretranscripción

Símbolo No ASCII : Ejemplo	Transformación : Ejemplo	Irregularidad Ortográfica : Ejemplo	Fonema Equivalente : Ejemplo	Irregularidad Ortográfica : Ejemplo	Fonema Equivalente : Ejemplo
“á” : “cuál”	“cuAl”	“cc” : “accionar”	/ks/ : “aksionar”	“gui” : “guitarra”	/g/ : “gitaRa”
“é” : “café”	“cafE”	“ll” : “llamar”	/Z/ : “Zamar”	“que” : “queso”	/k/ : “keso”
“í” : “maría”	“marIa”	“rr” : “carro”	“R” : “caRo”	“qui” : “quizá”	/k/ : “kisA”
“ó” : “noción”	“nociOn”	“ch” : “chamaco”	“C” : “Camaco”	“ce” : “cemento”	/s/ : “semento”
“ú” : “algún”	“algUn”	“ge” : “gelatina”	/x/ : “xelatina”	“ci” : “cimiento”	/s/ : “simiento”
“ü” : “güero”	“gWero”	“gi” : “gitano”	/x/ : “xitano”	“y” Fin de palabra : “buey”	/i/ : “buei”
“ñ” : “niño”	“niNo”	“gue” : “guerra”	/g/ : “geRa”	“sh” : “sharon”	/S/ : “Saron”
-	-	“ps” : “psicología”	/s/ : “sicologIa”	“h” (muda) : “hola”	: “ola”

como la “ch”), que se muestran en la misma tabla 3.1 se explican por sí mismas y las razones para escogerlas están muy bien expuestas en (Salcedo, 2010) y (Ríos Mestre, 1993).

Ahora bien, para esta tesis se diseñó una función que efectúa automáticamente las transformaciones de la tabla 3.1. Se trata de la función TT() y forma parte de la librería fonetica2.

Un detalle importante para destacar es que según la notación adoptada en esta tabla y en todas las reglas de transcripción tanto fonéticas como fonológicas, es que un fonema o conjunto de fonemas se representa entre diagonales, así: /g/, /tS/, /ks/, /p/, /k/, etc; y un conjunto de caracteres o morfema (puede ser un sólo carácter o una palabra completa) va entre comillas: “R”, “ch”, “sh”, “palabra”, “si”, “azul”, etc.

3.7.3 Paso 3. Contextos del grafema “x”

Existe un problema muy particular con el grafema “x”, que en el español de México puede presentar uno de cuatro diferentes sonidos y no existe una regla de transcripción fija para todos ellos (Hernández-Mena & Herrera-Camacho, 2014). La tabla 3.2 muestra estos cuatro casos del grafema “x” y la sustitución que debe ser hecha de forma manual para transcribirlo correctamente.

Si se quiere hacer un algoritmo de transcripción automática de palabras que contengan el grafema “x”, quizá tenga que considerarse el uso de listas de paro (“stop word lists” en inglés). Nótese que la sustitución del grafema “x” en palabras como “ximena” y “xavier”, es el grafema “j” y no un fonema como en los demás casos.

Tabla 3.2: Los cuatro sonidos diferentes del grafema "x".

Casos de "x"	Fonema Equivalente : Ejemplo
"sexto", "oxígeno"	/ks/ : "sexto", "oksIgeno"
"xochimilco", "xilófono"	/s/ : "sochimilco", "silOfono"
"xolos", "xicoténcatl"	/S/ : "Solos", "SicotEncatl"
"ximena", "xavier"	"j" : "jimena", "javier"

3.7.4 Paso 4. División silábica

En una transcripción fonológica no es absolutamente necesario que se indique la división silábica, pero hay que recordar que la transcripción fonológica es un paso previo a la transcripción fonética, y en esta última sí es muy importante conocer la división silábica.

Las reglas de división silábica son bastante sencillas y pueden hallarse en cualquier libro de fonética y fonología, por ejemplo en (Quilis, 1993), se da este conjunto de reglas:

1. Cuando una consonante se encuentra entre dos vocales, en virtud de la tendencia que posee el español a la sílaba abierta, la consonante se agrupa con la vocal siguiente: "kA.sa" casa, "mi.rA.ron" miraron, "de.mO.ra" demora, etc.

2. Cuando dos consonantes se encuentran entre dos vocales, hay que tener en cuenta:

   a) Son inseparables los grupos que están formados por consonantes bilabiales o labiodentales más una líquida[3.7]: /pr, br, pl, bl, fr, fl/, como: "o.prI.mo" oprimo, "o.brE.ro" obrero, "a.plO.mo" aplomo, "a.blAn.do" hablando, "kA.fre" cafre, "a.flo.xar" aflojar.

   Igualmente, los formados por consonantes linguovelares más líquidas: /gr, gl, kr, kl/, como: "lo.grAr" lograr, "lA.cre" lacre, "a.kla.mar" aclamar, "a.gru.pAr" agrupar.

   Y, finalmente, los formados por consonantes linguodentales más vibrantes: /dr, tr/, como: "cuA.dro" cuadro, "cuA.tro" cuatro.

   b) Cualquier otra pareja de consonantes que se encuentre entre dos vocales queda dividida, de manera que la primera consonante cierra la sílaba inmediatamente anterior y la segunda forma parte de la rama explosiva de la sílaba siguiente. Por ejemplo: "in.se.pa.rA.ble" inseparable, "kuEn.ta" cuenta, "ar.tIs.ta" artista, etc.

[3.7] los fonemas líquidos son el conjunto formado por las vibrantes /r/, /r(/ y el fonema /l/.

3. Cuando tres o más consonantes se encuentran entre dos vocales puede ocurrir:

   a) Que las dos últimas formen un grupo consonántico, una de las cuales sea una líquida; “in.fla.mAr” inflamar, “kon.tra.Er” contraer, “em.ple.A.dos” empleados, “en.glo.bAr” englobar, en donde permanece inseparable el grupo consonántico formado por consonante + líquida.

   b) Que las dos primeras formen una secuencia “ns” o “bs”, también inseparables en estas circunstancias: “kons.tru.Ir” construir, “ins.tau.rAr” instaurar, “kons.tAr” constar, “obs.tA.cu.lo” obstáculo, “obs.truk.siOn” obstrucción, etc.

4. El contacto entre dos vocales que no sean débiles[3.8] da origen a dos sílabas distintas: “a.E.re.o” aéreo, “pe.le.Ar” pelear, “lE.a” lea, etc.

5. El contacto de una vocal fuerte y otra débil o viceversa, si forma diptongo, constituye una sílaba: “Ai.re” aire, “eu.rO.pa” Europa, “A.sia” Asia, “buE.no” bueno, etc.

6. Un triptongo, del mismo modo que el diptongo, forma sílaba o parte de ella: “a.so.siAis” asociáis, “buEi” buey, etc.

7. Cuando se encuentran en contacto una vocal fuerte no acentuada y una vocal débil acentuada, se originan dos sílabas distintas: “a.bI.a” había, “paIs” país, “re.U.no” reúno, “ba.Ul” baúl, etc.

La función diseñada en esta tesis para efectuar división silábica se conoce con el nombre de div_sil(), y es parte de la librería fonetica2.

3.7.5 Paso 5. Transcripción final

Después de aplicar las transformaciones indicadas en las tablas 3.1 y 3.2, y hacer la división silábica, se está en condiciones de aplicar el último paso en el proceso de creación de transcripciones fonológicas, que no es más que convertir los fonemas no procesados hasta ahora. La tabla 3.3 muestra estas últimas transformaciones de grafemas a fonemas.

Nótese que en la tabla 3.3 el símbolo “_7” indica en Mexbet que se trata de la vocal tónica. Además, grafemas como “p” o “t” no están en la tabla 3.3 porque su equivalente en fonema, utiliza los mismos símbolos /p/ y /t/, respectivamente.

Una función de la librería fonetica2 que efectúa transcripciones fonológicas en el nivel T29 de Mexbet se llama T29(). La función T29() depende de otras funciones de la librería fonética2, a las que manda a llamar de manera automática, según sean necesarias.

[3.8] Las vocales débiles son la “u” y la “i” y las fuertes son la “a”, la “e” y la “o”.

Tabla 3.3: Transformaciones finales grafema a fonema

Grafema	Fonema	Grafema	Fonema
"A"	/a_7/	"C"	/tS/
"E"	/e_7/	"c"	/k/
"I"	/i_7/	"j"	/x/
"O"	/o_7/	"v"	/b/
"U"	/u_7/	"z"	/s/
"N"	/n∼/	"r"	/r(/
"y" no a final de palabra	/Z/	"R"	/r/
"W"	/u/		

3.7.6 Ejemplos de transcripción fonológica

En esta sección se dan algunos ejemplos que muestran cómo con estos pasos se puede hacer una transcripción fonológica fácilmente.

Supóngase que se desea transcribir fonológicamente en alfabeto Mexbet, las siguientes palabras en español:

1. `Palabra: "llanamente"`

- El paso 1 es indicar la vocal tónica en mayúscula. En este caso la palabra se trata de un adverbio terminado en "mente". En español, las únicas palabras que tienen dos vocales tónicas son estos adverbios, por lo tanto la palabra queda:

  ```
  llAnamEnte
  ```

- El paso 2 es sustituir ciertos caracteres de la tabla 3.1. En este caso sólo se cambia la "ll" por el fonema /Z/:

  ```
  ZAnamEnte
  ```

- El paso 3 se ignora en este caso porque la palabra no contiene la grafía "x".

- El paso 4 es dividir a la palabra en sílabas:

```
ZA.na.mEn.te
```

- El paso 5 consiste en efectuar las últimas transformaciones de caracteres de la tabla 3.3. Sólo se cambian los caracteres “A” y “E” por los símbolos Mexbet que indiquen a la correspondiente vocal acentuada. En este caso /a_7/ y /e_7/, respectivamente:

```
/ Z a_7 . n a . m e_7 n . t e /
```

2. `Palabra: "alheña" (un tipo de arbusto)`

- Paso 1. Se indica la vocal tónica:

```
alhEña
```

- Paso 2. Sustituciones de la tabla 3.1. La “h” se quita porque es muda y la “ñ” se cambia por “N”:

```
alENa
```

- Paso 3. Se ignora porque la palabra no contiene la grafía “x”.
- Paso 4. Dividir a la palabra en sílabas:

```
a.lE.Na
```

- Paso 5. Sustituciones de la tabla 3.3. Se cambia “E” por /e_7/ y “N” por /n∼/:

```
/ a . l e_7 . n~ a /
```

3. `Palabra: "desoxirribonucleótido"`

- Paso 1. Se indica la vocal tónica, que en este caso coincide con la vocal con acento gráfico:

  ```
  desoxirribonucleOtido
  ```

- Paso 2. Sustituciones de la tabla 3.1. La "rr" se cambia por "R":

  ```
  desoxiRibonucleOtido
  ```

- Paso 3. Sustituciones de la grafía "x" por las mostradas en la tabla 3.2. La "x" en este caso suena como /ks/:

  ```
  desoksiRibonucleOtido
  ```

- Paso 4. Dividir a la palabra en sílabas:

  ```
  de.sok.si.Ri.bo.nu.cle.O.ti.do
  ```

- Paso 5. Sustituciones de la tabla 3.3. Se cambia "R" por /r/, la "O" por /o_7/ y la "c" por /k/:

  ```
  / d e . s o k . s i . r i . b o . n u . k l e . o_7 . t i . d o /
  ```

3.8 Reglas fonéticas de Mexbet

En esta sección se explica cómo hacer transcripciones fonéticas a partir de transcripciones fonológicas hechas en Mexbet.

Un programa que sirve para realizar transcripciones fonéticas o fonológicas se conoce como fonetizador. Dentro de la librería fonetica2 existe un fonetizador especializado en realizar transcripciones en el nivel T66 de Mexbet. Se trata de la función T66() que internamente manda a llamar a la función T29(), y que efectúa transcripciones fonéticas por medio del procedimiento descrito en esta sección.

Ahora bien, de la versión de 2004 de Mexbet se sabe que el nivel T22 es el único nivel para transcripciones fonológicas, sin embargo, en el caso de las transcripciones fonéticas, existen un total de dos niveles de granularidad llamados T44 y T54. No obstante, estos niveles no contienen todos los alófonos de Mexbet que aparecen en la tesis de Cuétara, aunque si incluyen el símbolo "_c" que se aplica a todos los fonemas oclusivos (ej. /k_c/, /p_c/, /t_c/). Este símbolo representa el "momento de cierre" de la oclusiva, que es cuando el aire está totalmente obstruido por los órganos articulatorios de la boca, toda vez que se pronuncia una consonante oclusiva.

Por tanto, en esta tesis se introduce un nuevo nivel fonético de Mexbet, llamado T66, que contiene todos los fonemas y alófonos de la versión de Cuétara (exceptuando el fonema /m_n/) más los dos fonemas extraídos del náhuatl /S/ y /tl/, y sin que se utilice el símbolo "_c".

La lista en el apéndice F muestra todas las reglas fonéticas de esta nueva versión de Mexbet. Los símbolos en / / denotan fonemas y los símbolos en [ ] denotan alófonos. En esta lista se pueden encontrar también, ejemplos de palabras transcritas en T29 entre / /, y en T66 entre [ ]. Las palabras en " " son las palabras que se quieren fonetizar y en estas se indica manualmente la vocal tónica con una mayúscula.

Las reglas vienen numeradas para poder hacer una rápida referencia a estas. Si se observa con cuidado, se verá que presentan una especie de código de colores, esto es, las reglas de la 1 a la 3 están en negritas porque están relacionadas al fonema /a/, las reglas 4 y 5 no están en negritas y se relacionan con el fonema /e/, las reglas 6 y 7 vuelven a estar en negritas, pero se relacionan con el fonema /o/, y así sucesivamente. Es probable que este intercalar entre reglas en negritas y sin negritas, ayude a leerlas un poco más fácilmente.

En la siguiente sección se muestran ya directamente ejemplos de fonetización de palabras en nivel T66 porque las reglas del apéndice F son bastante explícitas. Esto quiere decir que los ejemplos que se darán tienen únicamente la finalidad de familiarizar al lector con la correcta interpretación del listado de reglas.

Es preciso recordar que en estos ejemplos, se parte de transcripciones en nivel T29, de la forma:

```
/ d e . s o k . s i . r i . b o . n u . k l e . o_7 . t i . d o /
/ Z a_7 . n a . m e_7 n . t e /
/ a . l e_7 . n~ a /
```

Y para aplicar cada regla se recorre fonema por fonema de izquierda a derecha.

3.8.1 Ejemplos de transcripción fonética

a) Convertir a T66 la transcripción:

```
/ a . Z a . n a . m i e_7 n . t o /
```

De izquierda a derecha el primer fonema que nos topamos es /a/, así que aplicamos la regla 1, que dice:

[a_j]: Ante consonante palatal; En diptongo decreciente con /i/

Porque si se revisa el apéndice H se verá que el fonema /Z/ es consonante y es palatal. Por lo tanto, la transcripción pasa a:

```
/ a_j . Z a . n a . m i e_7 n . t o /
```

Ahora sigue el fonema /Z/, así que aplicamos la regla 34 y no la 33, porque /Z/ no está al inicio de la palabra, ni después de pausa. Como referencia, la regla 34 dice:

/ Z /: En todos los demás contextos

Esto quiere decir que hasta este momento la transcripción permanece igual que en el paso anterior.

Sigue otro fonema /a/, pero en este caso aplicamos la regla 3, que dice:

/ a /: En todos los demás contextos

Debido a que no se puede dar cumplimiento ni a la regla 1 ni a la regla 2 en este caso; la transcripción sigue sin cambios.

Sigue el fonema /n/, y como en este caso está al inicio de la sílaba, se aplica la regla 40:

/ n /: Inicio de sílaba, sin condición

Por lo tanto, la transcripción sigue sin cambios.

Sigue otro fonema /a/ en dónde se vuelve a aplicar la regla 3. La transcripción sigue sin cambios.

Sigue el fonema /m/ y se le aplica la regla 35, que dice:

/ m /: En todos los contextos, principalmente inicial de sílaba

La transcripción sigue sin cambios.

Sigue el fonema /i/, y como está al inicio de un diptongo, se aplica la regla 10, que dice:

[j]: En posición inicial de diptongo

Por lo tanto, la transcripción cambia a:

```
/ a_j . Z a . n a . m j e_7 n . t o /
```

Sigue el fonema /e_7/, pero lo consideramos como si no estuviera acentuado (/e/) para poder buscar entre las reglas.

Aplicamos la regla 4 porque este fonema /e/, está en una sílaba trabada por /n/ ("mien"). La regla 4 dice:

/ e /: Libre, en sílaba abierta; En sílaba cerrada trabada por /m, n, s, d/

La transcripción no sufre cambios.

Sigue otro fonema /n/, pero esta vez le aplicamos la regla 39 porque está ante el fonema dental /t/. La regla 39 dice:

[n_[ ]: Ante dental se dentaliza

Por lo tanto, la transcripción cambia a:

```
/ a_j . Z a . n a . m j e_7 n_[ . t o /
```

Sigue el fonema /t/, al que sólo se le puede aplicar la regla 17, que dice:

/ t /: Inicial de sílaba, mantiene sus rasgos originales

Y la transcripción no cambia.

Por último, sigue el fonema /o/, al que le aplicamos la regla 6 porque está en sílaba abierta. La regla 6 dice:

/ o /: Libre, en sílaba abierta

Y como la transcripción no cambia aquí, se tiene que la transcripción fonética final de la palabra "allanamiento" es:

```
[ a_j . Z a . n a . m j e_7 n_[ . t o ]
```

b) Convertir a T66 la transcripción:

```
/ a_7 u n . k e /
```

De izquierda a derecha aparece primero el fonema /a_7/, que consideramos sin acento (/a/) para poder indagar en las reglas. Le aplicamos la regla 2, que dice:

[a_2]: En diptongo decreciente con /u/; Ante /o/; En sílaba cerrada trabada por /l/; Ante el fonema consonántico /x/

Porque está en diptongo decreciente con /u/. La transcripción cambia a:

```
/ a_2_7 u n . k e /
```

Sigue el fonema /u/ y se le aplica la regla 15, que dice:

[u( ]: En posición final de diptongo

Porque está efectivamente en posición final de diptongo y la transcripción cambia a:

```
/ a_2_7 u( n . k e /
```

Sigue el fonema /n/, que está ante el fonema velar /k/ (ver apéndice H), por lo tanto, se aplica la regla 42, que dice:

[N]: Ante velar se velariza

Y la transcripción cambia ahora a:

```
/ a_2_7 u( N . k e /
```

Sigue el fonema /k/ que está ante la vocal anterior /e/ (ver apéndice H), por lo que se aplica la regla 18, que dice:

[k_j]: Se palataliza ante vocales anteriores (palatales)

Y la transcripción cambia a:

```
/ a_2_7 u( N . k_j e /
```

Finalmente, sigue el fonema /e/ que está en sílaba abierta, por lo tanto, se le aplica la regla 4, que dice:

/ e /: Libre, en sílaba abierta; En sílaba cerrada trabada por /m, n, s, d/

Y como esta última regla no genera ningún cambio, la transcripción fonética final de la palabra “aunque” es:

```
[ a_2_7 u( N . k_j e ]
```

Capítulo 4

Recursos lingüísticos

En este capítulo se habla sobre recursos lingüísticos disponibles para el área de reconocimiento de voz en español, así como de la creación del corpus CIEMPIESS, que fue desarrollado en parte para los experimentos de esta tesis[4.1], y en parte para contribuir con los recursos de reconocimiento de voz en español de México, que a la fecha, son bastante escasos.

4.1 Corpus lingüísticos

En el campo de la lingüística, el término corpus puede resultar ambiguo, porque por lo regular denota simplemente una colección de textos de cualquier tipo. Sin embargo, debe de cumplir con ciertas características muy específicas para que pueda llamarse así.

En (Torruella & Llisterri, 1999) se muestran las distinciones que hay entre diferentes tipos de colecciones:

- **Archivo/colección (informatizado)**: Es un repertorio de textos en soporte informático, sin buscar ningún tipo de relación entre ellos.
- **Biblioteca de textos electrónicos**: Es una colección de textos en soporte informático, guardados en un formato estándar, siguiendo ciertas normas de contenido, pero sin un criterio riguroso de selección.
- **Corpus informatizado**: Es una recopilación de textos seleccionados según criterios lingüísticos, codificados de modo estándar y homogéneo, con la finalidad

[4.1] La información sobre corpus lingüísticos textuales y orales presentada en este capítulo se basa principalmente en estos cuatro trabajos: (Casacuberta et al., 1992), (Torruella & Llisterri, 1999), (Llisterri, 2004), (Sierra-Martínez & Rosas-González, 2010).

> de poder ser tratados mediante procesos informáticos, y destinados a reflejar el comportamiento de una o más lenguas.

Por lo tanto, un corpus lingüístico debe cumplir con ser una selección de textos destinada a estudiar cierto fenómeno, y lo más importante es, que tiene que estar presentado de una forma tal, que permita a los investigadores utilizar medios informáticos para su manipulación y consulta.

Ahora bien, existe una clasificación bastante general que divide a los corpus lingüísticos en dos tipos: corpus textuales y corpus orales, pero en realidad, los corpus pueden ser categorizados de muchas más maneras. Por ejemplo, Sierra (Sierra-Martínez & Rosas-González, 2010) nos presenta un estudio en dónde nos enlista diferentes criterios de clasificación, que son:

- El origen de los elementos
- La codificación y anotación
- La especificidad de los elementos
- La temporalidad
- El propósito
- El lenguaje
- La cantidad de texto
- La distribución de los elementos
- La documentación
- La accesibilidad
- La autoría
- La representatividad
- La espontaneidad

Aunque puedan existir muchas otras clasificaciones más, el objetivo de esta sección es servir de puente entre los conceptos propios de la lingüística y los del campo de reconocimiento de voz. En la siguiente sección, se rescatan estas ideas para seleccionar las que puedan relacionarse más con el campo del reconocimiento de voz, pero antes, es preciso conocer cuáles son los aspectos generales que, según la lingüística, deben tomarse en cuenta a la hora de diseñar un corpus lingüístico. Al respecto, Toruella (Torruella & Llisterri, 1999) apunta los siguientes:

- Finalidad del corpus.
- Límites del corpus.
- Tipo de corpus.
- Proporciones de los diferentes grupos temáticos del corpus.
- Población y muestra.
- Número y longitud de los audios/textos de muestra.
- Captura de los audios/textos y etiquetado.
- Procesamiento del corpus.
- Crecimiento del corpus y "feedback".
- "Hardware" y "software".
- Aspectos legales.
- Presupuesto y etapas.
- Adquisición de los datos.
- Selección de locutores (corpus orales).
- Transcripción fonética segmental y suprasegmental.
- La transcripción y codificación de la lengua oral.

4.1.1 Corpus orales para uso en reconocimiento de voz

Basado en los criterios expuestos en la sección precedente, un corpus oral para uso en reconocimiento automático de voz, puede ser clasificado con base en lo siguiente:

- **Origen de los elementos**: Un corpus de este tipo puede clasificarse según el origen del audio con el que fue creado, lo cual es relevante en la fase de reconocimiento.
- **La codificación y anotación**: Lo más recomendable es que todos los archivos de texto que conforman al corpus, tengan una codificación de caracteres ASCII o UTF-8. También debe seleccionarse un AFCA para crear los diccionarios de pronunciación, que de preferencia siempre deberían estar incluidos en un corpus de este tipo. No es vital que un corpus grande esté anotado a nivel palabra, sin embargo, el porcentaje de reconocimiento de un sistema entrenado con un corpus anotado a nivel fonema, será mucho más elevado que con uno anotado a nivel

palabra. Debe entenderse por "anotado", a que cada audio del corpus cuente con información sobre dónde comienza un elemento, como una palabra o un fonema y dónde termina. Es así como un corpus puede ser clasificado según su nivel de anotación o según su codificación de caracteres[4.2].

- **Especificidad de los elementos**: Un corpus oral de este tipo, no requiere que su audio tenga necesariamente algún sentido. Puede diseñarse un corpus con pedazos de audio sin orden, siempre y cuando contenga representatividad de los elementos que se desea reconocer (palabras, silabas, fonemas, etc.) y, en el contexto deseado (habla espontánea, habla leída, con ciertas condiciones de ruido, etc.).

- **Propósito**: Un corpus de este tipo generalmente se utiliza en la creación de modelos acústicos para uso en sistemas ASR.

- **Lenguaje**: Es muy importante que se tenga claro si el corpus será creado con habla leída, conversaciones espontáneas, o si será hecho por actores, o quizá extraído de algún programa de noticias, o de la radio, etc, ya que esto impacta en la calidad del reconocimiento. Así, el habla espontánea es más difícil de reconocer que el habla leída, y el habla espontánea de un programa de noticias de la radio es más uniforme que el habla de una entrevista.

- **Cantidad de audio**: Una manera científica de determinar la cantidad de entrenamiento requerida para las necesidades de reconocimiento de un problema dado, es la "curva de aprendizaje", que consiste en una gráfica en dónde en el eje horizontal se coloca la cantidad de entrenamiento (que puede ser en horas, en megabytes, en número de palabras o de archivos, etc.), y en el eje vertical se pone el porcentaje de error en el reconocimiento o "word error rate" (WER). Ahora bien, para trazar la gráfica se requiere tener un conjunto de audio de prueba fijo. Luego, se ejecuta un ronda de entrenamiento con sólo un pequeño porcentaje del corpus de entrenamiento; digamos el 10 %, después se hace el reconocimiento del conjunto de audio de pruebas fijo y se traza el WER en la gráfica. El proceso se repite ahora con el 20 %, el 30 %, etc. hasta llegar al 100 % del corpus de entrenamiento. Al principio, siempre se observará que el WER baja conforme la cantidad de entrenamiento aumenta, pero si esta se sigue incrementando, llegará un momento en que la curva sea paralela al eje horizontal. Este punto significa que aunque incrementemos el audio de entrenamiento, el WER permanecerá constante y, por lo tanto, se determina así el tamaño óptimo del corpus, para resolver cierto problema de reconocimiento en particular. Aún más importante que la cantidad de audio para incrementar la precisión de un sistema, está la cantidad de hablantes en el corpus (Barnard et al., 2009). Ahora bien, los expertos en reconocimiento de voz de la universidad Carnegie Mellon nos comparten, con base en su experiencia, los siguientes montos de audio y hablantes recomendados por ellos para resolver diferentes problemas de reconocimiento[4.3]:

[4.2] Otras codificaciones muy comunes aparte de la ASCII son UNICODE y UTF-8.

[4.3] Ver: http://cmusphinx.sourceforge.net/wiki/tutorialam

  - Una hora de corpus para reconocimiento de comandos por parte de un solo hablante.
  - Cinco horas de corpus y 200 hablantes para reconocimiento de comandos de voz por parte de muchos hablantes.
  - Diez horas de corpus para un sistema de dictado por parte de un solo usuario.
  - Cincuenta horas de corpus y 200 hablantes para un sistema de dictado para muchos usuarios.

- **Distribución de los elementos**: En reconocimiento de voz, un corpus balanceado fonéticamente es aquel que contiene todos los fonemas de la lengua en que fue creado y en sus mismas proporciones. Sin embargo, dependiendo el problema que se pretenda resolver, el criterio para llamar al corpus "balanceado" podría variar. Por ejemplo, si sólo se desea reconocer cierto número de frases equiprobables, el corpus estará balanceado si cuenta con todas las frases en igual número. Otro aspecto importante para el balanceo de un corpus es el género de los hablantes. Se dice que un corpus está balanceado en género si la mitad de su audio proviene de hablantes masculinos y la otra mitad de hablantes femeninos.
- **Accesibilidad**: Un corpus oral puede clasificarse en público o privado.
- **Representatividad**: Un corpus de este tipo puede estar anotado a nivel palabra, a nivel fonema, a nivel sílaba, etc. Esto significa que la representatividad del corpus tendrá que ver con la cantidad y la variedad de los elementos anotados.
- **Espontaneidad**: Es muy relevante si el audio contiene habla leída, o si fue una conversación espontánea, o si fue hecho a partir de locutores entrenados. Cualquier tipo de audio es válido para conformar el corpus, e incluso combinarlos, siempre y cuando se especifique.

Una vez conocidos los criterios de clasificación más importantes en el campo del reconocimiento de voz, es preciso hablar sobre los aspectos de diseño de este tipo de corpus. Estos aspectos son:

- **Finalidad del corpus**: Un corpus oral para uso en reconocimiento de voz tiene la finalidad principal de que se use para crear modelos acústicos para un sistema de reconocimiento de voz.
- **Límites del corpus**: Normalmente, un corpus de este tipo se diseña para los hablantes de una región en particular. Siempre tiene que delimitarse y especificarse la variante del idioma en el que está basado.
- **Proporciones de los diferentes grupos temáticos del corpus**: En un corpus oral de este tipo, no es relevante el tipo de temas que se traten en los audios que lo componen, sino solamente la calidad del audio de las grabaciones, el acento regional de los hablantes y la espontaneidad.

- **Población y muestra**: Un corpus debe tener representatividad de los elementos lingüísticos que se desean estudiar. Estos pueden ser: frases, palabras, fonemas, etc. Un sistema ASR entrenado con un corpus que cuente con un número insuficiente de estos elementos, tendrá peor desempeño que un sistema con un corpus bien balanceado.

- **Número y longitud de los audios/textos de muestra**: La cantidad de audio óptima para un sistema de reconocimiento particular puede determinarse mediante una curva de aprendizaje. Ahora bien, también se recomienda que cada audio tenga una longitud no menor a 5 segundos y no mayor a 30 segundos, para que el proceso de extracción de características ocurra de manera adecuada.

- **Captura de los audios/textos y etiquetado**: Por lo general, la recopilación de audio para un corpus es una tarea meticulosa y ardua. En reconocimiento de voz, un conjunto de audio es casi inútil si no se cuenta con sus transcripciones debidamente acomodadas en archivos con cierto formato predeterminado. El nivel de anotación mínimo útil de un corpus se consigue con sólo las transcripciones ortográficas de los archivos de audio que conforman al corpus. Según unos cálculos contenidos en la "Lecture 1" del "Speech Tools Minicourse 2009" de la universidad de Illinois en Estados Unidos[4.4], este proceso de transcripción ortográfica toma aproximadamente 4 horas-hombre por cada hora de audio. No obstante, existen más niveles posibles de transcripción, como por ejemplo, la transcripción alineada en tiempo a nivel palabra, que sería ligeramente más costosa; o la transcripción fonética alineada en tiempo, que requeriría de 100 horas-hombre por cada hora de audio.

- **Procesamiento del corpus**: La cantidad de herramientas requeridas para procesar un corpus oral de este tipo, está determinada por los objetivos del mismo. Para un corpus simple, con transcripciones a nivel palabra no alineadas en tiempo, sólo se requiere un software de procesamiento de audio y un editor sencillo de archivos de texto. Sin embargo, si se va a hacer anotado fonético alineado en tiempo, se requiere lo mismo, más un software que permita hacer análisis espectrográfico de los archivos de audio. En el primer caso, se requerirá también un fonetizador automático para crear los diccionarios de pronunciación que se deben incluir con el corpus, y en el segundo, no es necesario el fonetizador porque el diccionario de pronunciación se puede extraer de las etiquetas de tiempo del corpus, mediante un pequeño script computacional.

- **Crecimiento del corpus y "feedback"**: Este punto se refiere principalmente a que es preciso monitorear el corpus, conforme se va creando, para preservar en todo momento la representatividad de sus elementos.

- **"Hardware" y "software"**: Es preciso contar con software especializado para crear todas las partes del corpus. Algunos de estos softwares podrán conseguirse

[4.4] Ver en: http://www.isle.illinois.edu/sst/courses/minicourses/2009/

de manera comercial o libremente en la red si son de tipo "open source". Otros son más específicos, tal es el caso de los fonetizadores automáticos, que son muy especializados, dependiendo de la variante del idioma. Es conocido que el tipo de formato en el que el audio viene codificado, e incluso el dispositivo con el que es grabado, puede influir en el reconocimiento (ver (Tsiakoulis & Potamianos, 2010), (Huerta & Stern, 1998) y (Huerta, 2000)). La recolección del audio debe ser lo más homogénea y ordenada posible, aunque después se mezcle con audio de otro tipo.

- **Aspectos legales**: En el caso de México, un corpus oral no se puede registrar más que bajo la figura de los derechos de autor. Si el audio proviene de grabaciones hechas por los mismos creadores del corpus, es muy recomendable hacer firmar un documento a cada hablante en donde aprueben donar su voz. Si el audio proviene de internet, es conveniente asegurarse de que el contenido es libre para poder usarlo sin problemas, y si viene de la radio comercial o de la televisión, probablemente sería necesario contactar con algún representante de ese medio para que nos autorice hacer uso de su material. En la red existen sitios como https://archive.org/ que comparten contenido 100 % libre, ya que los usuarios ceden expresamente sus derechos. Si se desea crear un corpus sin fines de lucro es conveniente registrarlo. Una forma fácil de hacerlo puede ser recurriendo a una licencia de "creative commons"[4.5] que tiene la particularidad de ser gratuita y reconocida a nivel mundial, o donándolo a alguna institución especializada como el "linguistic data consortium"[4.6] o la "european language resources association"[4.7].

- **Presupuesto**: Es preciso tomar en cuenta los costos de transcripción, la compra del equipo, la renta de la cabina de grabación y el registro del corpus, entre otros aspectos.

- **Adquisición de los datos**: En este punto, se hace hincapié en que se debe ser meticuloso y ordenado al adquirir los datos. Estos deben ser en principio, lo más homogéneos posible, es decir, deben ser recolectados con el mismo modelo de micrófono, el mismo modelo de grabadora, o el mismo formato de audio, con el mismo tipo de variante dialectal.

- **Selección de locutores (corpus orales)**: En reconocimiento de voz, por lo regular se clasifica a los hablantes en las siguientes grandes categorías: hombres, mujeres, niños, adultos maduros y ancianos. En (López Escobedo, 2011) se explica cómo las características de la voz de una persona varían conforme a su edad.

- **Transcripción fonética segmental y suprasegmental**: El nivel más básico, anotar un corpus significa tener las transcripciones sin alinear con sus respectivos audios. Un nivel más fino, sería tener las transcripciones alineadas en tiempo, a

[4.5] En http://creativecommons.org/

[4.6] En https://www.ldc.upenn.edu/

[4.7] En http://www.elra.info/

nivel palabra. Con esto se deduce que se puede anotar un corpus a nivel sílaba, fonema, sufijo, alófono, difonema y un largo etcétera; también se podría hacer etiquetado prosódico y hasta de emociones (suprasegmental). La mayoría de las veces, el nivel de granularidad con el que se anote el corpus, dependerá del problema a resolver y de la cantidad de audio disponible. Por ejemplo, para un corpus muy grande, de más de 50 horas de audio, el nivel de transcripción no alineada en tiempo, provee un buen porcentaje de reconocimiento para ciertos problemas de reconocimiento acotados, como el reconocimiento de comandos de voz. Sin embargo, cuando se tiene un corpus pequeño de menos de 10 horas, el etiquetado fonético alineado en tiempo, puede ayudar un poco a mejorar el porcentaje de reconocimiento.

- **Transcripción y codificación de la lengua oral**: Es preciso elegir un único AFCA para todo el corpus, que ayude a representar toda la información que deseamos rescatar en la etapa de reconocimiento. Por ejemplo, un AFCA a nivel fonológico, no podría representar la información de los alófonos de la lengua deseada; de la misma manera, la falta de un símbolo para la vocal tónica, nos privaría de la posibilidad de poder reconocerla.

4.1.2 Corpus textuales para uso en reconocimiento de voz

Los corpus textuales para uso en reconocimiento suelen ser por lo regular, mucho más simples que los corpus textuales lingüísticos, y se utilizan básicamente para hacer los modelos de lenguaje y diccionarios de pronunciación que se utilizarán en el sistema ASR.

Por lo tanto, en reconocimiento de voz, estos corpus tienen un papel un tanto más secundario que los corpus orales. Sin embargo, al igual que en la sección anterior, se retomarán los elementos de clasificación y aspectos de diseño que sean más adecuados para el campo del reconocimiento de voz, omitiendo por razones de espacio, los que sean iguales o muy similares a los de los corpus orales.

Los criterios de clasificación más adecuados para este tipo de corpus son:

- **Origen de los elementos**: Estos corpus pueden venir de periódicos en línea, revistas, redes sociales o de transcripciones de conversaciones de diversa índole. Los libros también pueden servir, pero hay que ser cuidadosos con ellos, ya que pueden tener un lenguaje demasiado formal o muy alejado del estilo de lenguaje que se pretende reconocer.
- **Codificación y anotación**: En este caso la presentación del texto en formato ASCII, UNICODE, UTF-8, etc, y con cierto formato convenido por los diseñadores del corpus es más que suficiente. Normalmente, para el uso de herramientas de modelado de lenguaje automáticas, es mejor si se tiene una oración por cada

línea del archivo de texto. También podría resultar útil para un sistema de diálogo, anotar categorías gramaticales o hacer etiquetado POS (Part of Speech). Sin embargo, para reconocimiento de voz en sentido estricto, sólo hace falta que el texto esté en un formato como el UTF-8 (para poder manejar acentos, la "ñ" y la "ü" sin problemas) y sin faltas de ortografía.

- **Especificidad de los elementos**: Este punto se refiere a que es preciso diferenciar los textos del corpus por su estilo de lenguaje. Si por ejemplo, el corpus es recopilado de diarios de noticias, el lenguaje será neutral, aunque bien cuidado. Si por ejemplo, el corpus viene de transcripciones de entrevistas, el texto tendrá muchas muletillas como "em", "este", afirmaciones como "aja", "okay", y todo tipo de disfluencias[4.8]. Por lo tanto, al igual que en los corpus orales es preciso ser ordenado al clasificar el corpus, según el estilo del lenguaje que maneja.

- **Temporalidad**: Este punto va en el mismo sentido que el anterior. Si se eligen textos muy viejos para conformar el corpus, se corre el riesgo de obtener modelos de estilos de lenguaje antiguos, y que ya no estén en uso. Un claro ejemplo sería el uso de palabras como "comisteis", "vinisteis", "vuestro", etc. Por lo tanto, en este punto, sólo se reitera la importancia de clasificar un corpus textual por el estilo de lenguaje que manifiesta.

- **Propósito**: Se ha dicho que los propósitos principales de este tipo de corpus, son la creación de modelos de lenguaje y de diccionarios de pronunciación. Si el objetivo es estrictamente el primero, se debe buscar texto con el estilo exacto o más parecido al estilo del lenguaje que se quiere reconocer, pero si el objetivo es estrictamente el segundo, bastaría con tener listas con las palabras que se desee que el sistema reconozca.

- **Lenguaje**: En este tipo de corpus se podría hablar de dos grandes categorías: la del lenguaje bien cuidado y la del lenguaje espontáneo. El primero puede encontrarse en periódicos o transcripciones de programas de televisión o radio (incluidos programas de noticias) en dónde lo más común es que el lenguaje provenga de un guión premeditado o que las elocuciones las haga un locutor entrenado. El lenguaje espontáneo puede venir de transcripciones de entrevistas o de redes sociales como Twitter o Facebook, en dónde muchas veces, la gente escribe tal como habla, sólo que sin disfluencias como "este" y "em" , pero sí con muchas faltas de ortografía.

- **Cantidad de texto**: Teóricamente no existe una cantidad límite para el texto que pueda ser útil en la creación de un modelo de lenguaje y, de manera empírica, se sabe que entre más texto se tenga es mejor. Por lo tanto, se podría decir que los corpus textuales de este tipo siempre están en el orden de millones de palabras

[4.8] Las disfluencias en un corpus oral son pasajes en los que el hablante emite sonidos por medio de su aparato fonador, pero que no son palabras como tos, risa, chasquido de labios, sonidos de duda, etc.

(contando repeticiones), y si un corpus tiene menos de un millón de palabras, se le podría considerar como pequeño. Es importante monitorear la perplejidad [4.9] del corpus conforme va creciendo, para asegurarnos de que siempre sea baja, ya que entre más baja sea la perplejidad, mejor será el modelo de lenguaje (Jurafsky & Martin, 2000).

- **Distribución de los elementos**: Este punto se refiere a la idea de clasificar los corpus con base en la distribución y al porcentaje de los diferentes textos que lo componen, y de su estilo de lenguaje. Una diferencia muy importante entre los corpus textuales lingüísticos y los corpus textuales para uso en reconocimiento de voz, es que en los primeros importa el tema que abordan sus textos y en los segundos, importa más el estilo de lenguaje que contienen.
- **Representatividad**: En (Sierra-Martínez & Rosas-González, 2010) se definen corpus representativos y corpus oportunistas. Un corpus en el que se cuide que el tipo de textos que se guardan, sea equilibrado y que ayude a modelar adecuadamente una lengua determinada, puede llegar a ser fácilmente un corpus representativo; sin embargo, los corpus oportunistas son aquellos que recogen muestras sólo de cierto fenómeno que se quiere estudiar.

Análogamente a la sección anterior, a continuación se enlistan los aspectos de diseño de corpus textuales para uso en reconocimiento de voz, que presentan una gran diferencia con respecto a los corpus orales de este mismo tipo. No se desglosan, ya que la mayoría aparece en los criterios de clasificación y otros son obvios.

- **Finalidad del corpus**
- **Límites del corpus**
- **Proporciones de los diferentes grupos temáticos del corpus**
- **Población y muestra**
- **Número y longitud de los audios/textos de muestra**
- **Captura de los audios/textos y etiquetado**
- **Procesamiento del corpus**
- **Crecimiento del corpus y "feedback"**

[4.9] La perplejidad, es una medida que se utiliza para evaluar la calidad de los modelos de lenguaje. Esto es, se busca que el modelo de lenguaje en un sistema ASR, prediga con cierta exactitud la siguiente palabra que será pronunciada. Luego, si el modelo de lenguaje es muy predictivo, su perplejidad será baja.

4.1.3 Comparaciones entre corpus para uso en reconocimiento de voz

Una vez que se han conocido los criterios de clasificación y los aspectos de diseño aplicables a corpus para uso en reconocimiento de voz, sólo resta hablar sobre otro conjunto de características un tanto más pragmático, pero que resulta sumamente útil en la implementación concreta de sistemas ASR. Se trata de los criterios de comparación entre corpus.

Si se tiene que elegir entre varios corpus orales o textuales, para la resolución de un problema específico, es necesario contar con criterios que nos digan cuál corpus podría ser mejor para resolver dicho problema. Estos criterios son resultado tanto de la investigación bibliográfica realizada, como de la experiencia recabada, y son:

- **Duración**: La duración total de un corpus se calcula sumando la duración de todos los archivos individuales de audio que lo componen. En general, un corpus de más duración debería poder generar mejores modelos acústicos, pero también debería ser más costoso (en caso de ser un corpus de paga) y ciertamente, sería más difícil de procesar computacionalmente en la etapa de entrenamiento del sistema. Por lo tanto, si el problema es demasiado sencillo de resolver, no es necesario tener un corpus de tan larga duración.
- **Número de frases**: Los corpus orales para uso en sistemas ASR generalmente se componen de múltiples archivos de audio de una duración menor a los 30 segundos, en los que sólo esta contenida una única voz, la cual pronuncia una frase suelta sobre cualquier tema. Por lo tanto, el número de frases del corpus es relevante para estimar, aunque sea de manera superficial, su riqueza fonética ya que en general, entre más frases distintas tenga, es más probable que cuente con un mayor número de fonemas pronunciados, lo cual es relevante en la generación de los modelos acústicos.
- **Número de palabras**: Al igual que las características anteriores, el número de palabras puede decir mucho sobre la riqueza fonética del corpus, ya que como regla general, sería más rico entre más palabras distintas tuviera. Generar un archivo de frecuencia de palabras del corpus, puede ayudar a visualizar más fácilmente este punto.
- **Número de fonemas y alófonos**: Cuando los parámetros de los puntos anteriores no reflejen una idea clara de la riqueza fonética del corpus, lo mejor sería hacer un conteo de los fonemas y/o alófonos del mismo, para determinar si se adapta a nuestras necesidades. En el caso de un corpus oral, se puede tomar el diccionario de pronunciación para conocer cuáles son los fonemas existentes en el corpus, pero para conocer su proporción en el mismo, es preciso contar los fonemas y/o alófonos directamente del archivo de transcripción con ayuda del diccionario de pronunciación o de un fonetizador automático, como los que se encuentran en la librería *fonetica2* (ver apéndice G). En el caso de un corpus textual, el conteo de

fonemas y/o alófonos sólo podrá hacerse mediante el fonetizador automático, a menos que el corpus cuente con la transcripción fonética o fonológica de cada una de las palabras que lo componen.

- **Duración promedio de los audios**: Según los expertos en reconocimiento de voz de Carnegie Mellon, se recomienda que los archivos de audio de un corpus oral, tengan una duración de entre 5 y 30 segundos, y que empiecen y terminen en silencios[4.10]. Un corpus oral con archivos de audio demasiado largos, podría implicar que contiene grabaciones con más de una única voz, o que hay muchas partes en las que el locutor no habla, o que en el peor de los casos, existe música de fondo o ruidos ajenos a la voz del hablante; sin mencionar que un corpus con audios más largos, es más difícil de transcribir por parte de transcriptores humanos.
- **Palabras promedio en cada oración**: Si el corpus oral tiene audios con una duración promedio constante, implica que el número de palabras promedio por archivo de audio, será también constante. Un corpus que tiene una duración promedio de sus audios muy grande y que por el contrario, posee un promedio bajo de palabras por archivo de audio, reflejaría claramente la situación de que el archivo contiene muchos eventos donde no hay habla. El promedio de palabras por archivo puede además ayudar a detectar las partes del corpus en los que el habla se realiza a mayor o menor velocidad que en el resto del corpus. En el caso de un corpus textual que será usado para hacer modelos de lenguaje, lo ideal es que se acomode de tal forma que quede una oración bien formada por cada línea del archivo. Por tanto, un archivo de texto que contiene demasiadas palabras por línea indicaría claramente que no ha sido acomodado con el cuidado requerido para generar dicho modelo de lenguaje.
- **Perplejidad del modelo de lenguaje**: La perplejidad es una variable importante en el modelo de lenguaje de un sistema ASR. Por lo tanto, si se tiene que escoger entre varios modelos de lenguaje distintos, lo mejor sería elegir el que contenga el estilo de lenguaje que se quiere reconocer, y de entre esos, elegir el modelo que produzca una perplejidad más cercana a cero.

Ahora bien, aparte de estos criterios de comparación, existen también técnicas básicas para poder comparar la precisión de varios corpus que fueron construidos para la misma variante dialectal. Para esto, se requiere recopilar de cada corpus, un conjunto pequeño de audios de prueba, no mayor al 10 %, formando así el conjunto de pruebas único con el que se compararán todos los corpus.

Una vez hecho esto, se mezcla el audio restante de todos los corpus en una misma ronda de entrenamiento y se hace la ronda de reconocimiento del conjunto de pruebas único, por medio del modelo acústico generado con la mezcla de todos los corpus. De

[4.10] Ver: http://cmusphinx.sourceforge.net/wiki/tutorialam

este modo, el WER obtenido será la cota superior de precisión en el reconocimiento, es decir, que se supone que ningún corpus de manera individual, podría generar un modelo acústico con el que se obtuviera ese mismo WER.

Luego entonces, la prueba más sencilla sería entrenar un modelo acústico diferente con cada corpus y hacer una prueba de reconocimiento del conjunto de pruebas único. Es importante mencionar que los audios del conjunto de pruebas no deben estar presentes en la creación de ningún modelo acústico y que las rondas de reconocimiento que se hacen para cada modelo acústico, deben hacerse con el mismo sistema ASR y con la misma configuración. De este modo, se puede determinar cuál corpus produce los mejores modelos acústicos.

4.2 Algunos corpus disponibles para uso en sistemas ASR

Los corpus mencionados en esta sección, son aquellos que de un modo o de otro han servido para desarrollar el trabajo en la presente tesis. No todos los corpus mencionados son gratuitos; la mayoría de estos hay que solicitarlos al grupo que los desarrolló, por lo cual, este pequeño listado tiene el doble objetivo de mostrar al lector, la importancia de los recursos lingüísticos libres y de fácil acceso para el desarrollo de tecnologías de habla en México y en todos los países de habla hispana.

En el área de las tecnologías del lenguaje existen dos asociaciones muy importantes en las que se pueden descargar y distribuir corpus de muchos tipos y en diversas lenguas, tanto gratuitos como de paga. Se trata del "linguistic data consortium" (LDC)[4.11] que es un consorcio estadounidense de universidades, bibliotecas, corporaciones y laboratorios de investigación gubernamentales, que fue fundado en 1992 con el objetivo inicial de ser un repositorio y punto de distribución de recursos lingüísticos, pero que ahora realiza investigación propia en tecnologías del lenguaje. El LDC tiene su sede en la universidad de Pensilvania, Estados Unidos.

La otra asociación a la que se hace alusión es la "european language resources association" (ELRA)[4.12], que tiene como misión poner recursos lingüísticos a disposición de las tecnologías en ingeniería del lenguaje; y para lograrlo, la ELRA recolecta, distribuye, valida, estandariza, mejora y promueve la producción de recursos lingüísticos a través de campañas de evaluación de recursos y conferencias científicas.

Por lo tanto, algunos de los corpus mencionados en esta sección pueden conseguirse en alguna de estas dos asociaciones.

[4.11] Página: https://www.ldc.upenn.edu/
[4.12] Página: http://www.elra.info/

4.2.1 Corpus de referencia del español actual (CREA)

El corpus de referencia del español actual (CREA)[4.13], es un conjunto de textos de diversa procedencia, almacenados en soporte informático, del que es posible extraer información para estudiar las palabras, sus significados y sus contextos.

Un corpus de referencia es aquel que está diseñado para proporcionar información exhaustiva acerca de una lengua en un momento determinado de su historia, y por tanto, ha de ser lo suficientemente extenso para representar todas las variedades relevantes de la lengua en cuestión. Atendiendo a este criterio, el CREA cuenta hasta el momento con algo más de 160 millones de formas. Se compone de una amplia variedad de textos escritos y orales, producidos en todos los países de habla hispana desde 1975 hasta 2004. Los textos escritos, procedentes tanto de libros como de periódicos y revistas, abarcan más de cien materias distintas. La lengua hablada está representada por transcripciones de documentos sonoros, procedentes, en su mayor parte, de la radio y la televisión.

Lo que viene a ser más relevante del corpus CREA en esta tesis, es que está disponible su lista de frecuencia de palabras, que puede ser descargada por medio un archivo de texto plano con más de setecientas mil trescientas palabras distintas. Sin embargo, es preciso tener cuidado con estas palabras, ya que no todas están necesariamente en español, algunas están mal escritas o no existen, por ejemplo, una de sus entradas es la palabra "zzzzzetas", que obviamente está mal escrita. Esto puede resolverse fácilmente por medio del valor de frecuencia asignado a cada palabra, es decir, que las palabras con una muy baja frecuencia, es muy probable que sean palabras que no existan o que estén mal escritas.

El corpus CREA puede consultarse gratuitamente en la siguiente dirección:

http://corpus.rae.es/creanet.html

Y la liga directa para descargar la lista de frecuencia de palabras de todo el corpus es:

http://corpus.rae.es/frec/CREA_total.zip

4.2.2 Diccionario del español de México (DEM)

El diccionario del español de México (DEM)[4.14] es resultado de un conjunto de investigaciones del vocabulario utilizado en la República Mexicana a partir de 1921.

[4.13] La información sobre el corpus CREA mostrada en esta sección, fue obtenida de la página de la Real Academia Española. Ver: http://www.rae.es/recursos/banco-de-datos/crea

[4.14] La información sobre el corpus DEM mostrada en esta sección, fue obtenida de la página del COLMEX. Ver http://dem.colmex.mx/

Las investigaciones se llevan a cabo desde 1973 en el centro de estudios lingüísticos y literarios de el Colegio de México.

El diccionario del español de México es un diccionario integral del español en su variedad mexicana, elaborado sobre la base de un amplio estudio del corpus del español mexicano contemporáneo (1921-1974) y un conjunto de datos posteriores a esa última fecha hasta el presente.

Se trata de una obra original, de carácter descriptivo, hecha con criterios exclusivamente lingüísticos. Todo el vocabulario que incluye ha sido usado o se usa en México, al menos desde 1921.

Aunque el DEM no sea en sí mismo un corpus textual, puede resultar muy útil para la creación de diccionarios de pronunciación en español de México debido a que siendo un diccionario, su nomenclatura está correctamente escrita y además, es posible solicitar a sus desarrolladores que proporcionen la transcripción fonológica de cada palabra de su nomenclatura. Esta transcripción incluye la vocal tónica, lo que la hace sumamente valiosa para los temas tratados en esta tesis.

El corpus DEM puede consultarte en línea junto con los datos de los desarrolladores, en la siguiente dirección:

http://dem.colmex.mx/

4.2.3 Corpus VOXMEX

Se trata de un corpus fonéticamente balanceado de 200 hablantes que contiene los 5 principales dialectos de México (ver (Uraga & Gamboa, 2004)).

Las transcripciones de VOXMEX cubren todos los fonemas y alófonos del español de México en cada contexto posible y fue diseñado para hacer estudios sobre fonética y modelado acústico en el área del reconocimiento de voz.

Por lo tanto, este corpus serviría directamente para entrenar sistemas ASR en español. Se puede encontrar mencionado en el catálogo de la ELRA en la siguiente dirección:

http://universal.elra.info/product_info.php?cPath=37_39&products_id=86

4.2.4 Corpus DIME

El corpus DIME[4.15] consiste en un conjunto de diálogos en el dominio del diseño de cocinas que fueron recopilados bajo el protocolo del experimento del "mago de Oz", en el que un ser humano familiarizado con la tarea pretende ser el sistema computacional y otro ser humano soluciona un problema, en este caso, diseñar una cocina de manera cooperativa con el "mago". La interacción tanto lingüística como gráfica, se da a través de un sistema computacional.

En el experimento, el sujeto humano sabía que el "mago" era también un ser humano. Los diálogos fueron grabados de una manera controlada, y tanto el audio como el video de la interacción fueron recopilados automáticamente. Este tipo de tarea constituye un ejemplo de diálogo práctico, y el material recolectado ha sido objeto de análisis y etiquetado en varios niveles de representación lingüística, tales como el fonético, fonológico, prosódico y entonativo, léxico, sintáctico y pragmático. Esta información constituye la base empírica para la investigación del lenguaje conversacional en diálogos cooperativos.

Actualmente se cuenta con varias metodologías y herramientas de transcripción para este corpus, y el recurso se emplea en varios experimentos de lingüística computacional en el contexto del proyecto.

Como tal, este corpus no sirve directamente para el reconocimiento de voz, porque no viene en una arquitectura de archivos adecuada para ello. Sin embargo, este corpus sí puede servir para extraer diccionarios de pronunciación y modelos de lenguaje útiles para sistemas ASR.

Para poder tener acceso al corpus DIME es preciso contactar a sus desarrolladores cuya información de contacto es:

http://turing.iimas.unam.mx/~luis/DIME/CORPUS-DIME.html

4.2.5 Corpus DIMEx100

El corpus DIMEx100 tiene por objetivo la creación de modelos acústicos y diccionarios de pronunciación para uso en sistemas ASR de propósito general, con un vocabulario amplio, locutores diversos y en el español hablado en México. El corpus es también útil para la investigación en reconocimiento de locutores, la creación de voces en sistemas de síntesis de voz, y para la realización de estudios fonéticos de orientación computacional de la lengua española, con una sólida base empírica.

[4.15] DIME es el acrónimo de: "diálogos inteligentes multimodales en español". Parte de la información del corpus DIME mostrada en esta sección, fue obtenida de la página del IIMAS. Ver http://turing.iimas.unam.mx/~luis/DIME/CORPUS-DIME.html

Es claro que el corpus DIMEx100 puede usarse directamente para entrenar un sistema ASR, pero no fue compartido al público en general sino hasta mediados del año 2015. Es por eso que en esta tesis no fue posible utilizar sus audios para hacer experimentos. Sin embargo, en su sitio de internet siempre se ha compartido: el diccionario de pronunciación, el diccionario de relleno, el modelo acústico, el modelo de lenguaje y la lista de fonemas, todo esto en el formato del sistema Sphinx.

Los recursos antes mencionados pueden descargarse de la siguiente dirección:

http://turing.iimas.unam.mx/~luis/DIME/recursos/reconocedor-T22/rec-DIMEx100-T22.html

El corpus DIMEx100 tiene una duración total de 6.1 horas y puede descargarse completo en:

http://turing.iimas.unam.mx/~luis/DIME/CORPUS-DIMEX.html

4.2.6 Corpus Glissando

El corpus Glissando (Garrido et al., 2013) fue creado en la universidad española Pompeu Fabra en el año 2013. Se trata de un corpus oral etiquetado a nivel palabra, a nivel fonema y a nivel prosódico, en los idiomas: español de España y catalán. El alfabeto fonético utilizado para las etiquetas a nivel fonético fue X-SAMPA[4.16].

Para cada idioma, el corpus se divide en dos subconjuntos, uno de ellos se trata de audio conversacional y el otro se trata de emisión de noticias. El corpus Glissando tiene una duración total de 25 horas, con 28 hablantes por cada idioma, entre los cuales se cuentan 8 locutores profesionales, 4 de ellos presentadores de noticias y 4 de ellos actores de anuncios publicitarios.

Los objetivos principales que guiaron el diseño del corpus fueron (1) ofrecer una amplia gama de situaciones en las que haya una comunicación de la vida real que permita la caracterización de diferentes fenómenos prosódicos y (2) permitir estudios para la investigación de los diferentes estilos de discurso entre diferentes hablantes.

El corpus Glissando se comparte mediante una licencia "Creative Commons versión 3.0" y puede ser solicitado directamente, y de manera gratuita para fines de investigación, a su autor principal, el Dr. Juan María Garrido Almiñana, a la siguiente dirección:

http://glissando.upf.edu/es/node/9

[4.16] Para más información sobre X-SAMPA ver: http://www.phon.ucl.ac.uk/home/sampa/x-sampa.htm

Finalmente, resulta conveniente mencionar que el corpus Glissando en su estado normal no es apto para ser utilizado en un sistema de reconocimiento de voz, ya que aún hace falta generar todos los archivos de entrada discutidos en la sección 2.4, mediante la programación de scripts sencillos. El autor trabajó en la creación de esos archivos de entrada aptos para un sistema ASR, correspondientes sólo a la sección de noticias en español del corpus (6.5 horas de audio).

4.2.7 Corpus LATINO-40 Spanish Read News

El LATINO-40 es un corpus oral diseñado para entrenar sistemas de reconocimiento independientes del hablante en el español de Latinoamérica.

Contiene 20 hablantes mujeres, 20 hablantes varones, 4994 archivos de audio en formato SPHERE de NIST, con sus respectivas transcripciones, y una duración total de 6.8 horas. Los hablantes son originarios de los siguientes países de América Latina: Argentina, Cuba, Colombia, Guatemala, Chile, México, Nicaragua, Perú, Costa Rica, El Salvador y Venezuela.

Es claro que este corpus puede ser usado directamente para sistemas ASR en español y otras aplicaciones, como identificación de dialecto y de hablante. En el disco del corpus, los desarrolladores incluyeron un software para UNIX que permite leer los archivos de audio, ya que contienen un encabezado particular del formato SPHERE de NIST, que no es tan común en estos días y no puede ser leído por cualquier reproductor de audio.

LATINO-40 es un corpus de paga que puede solicitarse al LDC en la siguiente dirección:

https://catalog.ldc.upenn.edu/LDC95S28

4.2.8 Corpus Westpoint Heroico Spanish Speech

El “Heroico” es un corpus oral en español, que fue diseñado y recolectado por miembros del “department of foreign languages” (DFL) y el “center of technology enhanced language learning” (CTELL), para la creación de modelos acústicos para sistemas ASR.

Algunas partes de este corpus fueron diseñadas para modelar diálogos tipo pregunta/respuesta para usarse en sistemas de traducción en vivo. Por tanto, el corpus consiste en dos subconjuntos, uno recolectado en septiembre de 2001 en el “Heróico Colegio Militar” (HEROICO), en la ciudad de México, y el otro en la “United States Military Academy” (USMA), múltiples veces desde 1997. El subconjunto proveniente de la USMA incluye audio de hablantes no nativos, recolectado a través de un micrófono de garganta (throat microphone).

Este corpus cuenta con 19,021 archivos de audio con sus respectivas transcripciones y tiene una duración de 16.6 horas. Estos archivos están codificados a 16 bits, en formato WAV, con una frecuencia de muestreo igual a 22,050 hz.

El Heroico es un corpus de paga que puede solicitarse al LDC, en la siguiente dirección:

https://catalog.ldc.upenn.edu/LDC2006S37

4.2.9 Proyecto VoxForge

El proyecto VoxForge tiene como objetivo, guardar habla donada por personas en internet, para ser usada con herramientas de reconocimiento de voz libre y de código abierto (open-source).

Todas las voces que se recojan estarán disponibles bajo la licencia GPL y se utilizarán para crear un modelo acústico que se use con software "open-source" de reconocimiento de voz, como es el caso de Sphinx, Julius, HTK y Kaldi.

Sin embargo, los desarrolladores advierten en su pestaña de descargas lo siguiente:

"No existe, todavía, un corpus de voz o un modelo acústico en castellano para poder ser descargado. Si se quiere ver las grabaciones que están disponibles, es preciso visitar la página "Listen" para ver las voces recibidas hasta ahora."

VoxForge funciona de la siguiente manera: primero se accede a su sitio de internet y se crea una cuenta, luego se busca la liga para hacer la donación de voz, en la que se pide hacer ciertas comprobaciones del micrófono que se esté usando, como el volumen y el tipo de micrófono. Después de hacer las comprobaciones, se puede acceder a las oraciones que el mismo sitio genera de manera automática y que son extraídas de la novela del Quijote. Una vez que se han leído todas las oraciones generadas por el sistema y el audio ha sido grabado exitosamente, es posible descargar los audios donados por los demás usuarios.

El proyecto VoxForge hace esto mismo para muchos otros idiomas y su dirección de internet dedicada para el idioma español es la siguiente:

http://www.voxforge.org/es

4.3 Diseño de dos corpus orales en español de México

Dos textos clásicos, de corte lingüístico, creados específicamente para el diseño de corpus orales, son (Torruella & Llisterri, 1999) y (Llisterri, 2004); mientras que en (Hayamizu et al., 1993) y (Graff, 2002) se pueden apreciar dos textos creados por

ingenieros sobre este mismo tema. Ahora bien, en reconocimiento de voz, estos cuatro artículos son más la excepción que la regla, debido a que la mayoría de los demás artículos, no presentan una metodología de creación de corpus como tal, sino que son presentados de una forma "documentativa" del proceso que siguieron para crear sus respectivos corpus. Algunos ejemplos de esto son: (Garofolo et al., 1997), (Ostendorf et al., 1995), (Wang et al., 2005) y (Federico et al., 2000).

Ahora bien, para este trabajo, se crearon los corpus CHM150 y CIEMPIESS. Estos corpus son representativos porque muestran procesos de recolección de audio muy distintos: grabación de locutores y obtención de audio de un programa de radio, respectivamente. Por tanto, en las siguientes secciones se muestra el proceso seguido para su creación. Esto se hace para contribuir a la escasa información existente sobre metodologías de creación de corpus para uso en reconocimiento automático de voz.

4.3.1 Creación del corpus CHM150

El objetivo del corpus CHM150[4.17] es el de crear una base de datos de habla espontánea, en condiciones de ruido similares a las de una oficina tranquila y con los sexos de los hablantes balanceados.

El tamaño total de este corpus fue de aproximadamente 1.5 horas y se recopiló de entre 150 estudiantes universitarios (75 hombres y 75 mujeres) con edades que oscilaban entre los 18 y 39 años de edad, cuya lengua materna era el español que se habla en la zona metropolitana del Valle de México. La formulación del número de hablantes proviene del trabajo de (Barnard et al., 2009), en donde se demuestra que un corpus con sólo 50 hablantes puede bastar para ser útil en un sistema de reconocimiento de voz.

Para el proceso de grabación se utilizaron: una computadora de escritorio que era usada por el experimentador para almacenar la grabación y una lap-top conectada a un monitor externo, que tenia la finalidad de desplegar imágenes que el hablante pudiera describir. De esta manera, el hablante se quedaba sentado frente al monitor a una distancia aproximada de 50 cm del micrófono dinámico cardioide utilizado y se le pedía que describiera con sus palabras una de cinco imágenes posibles (ver figura 4.1). Cuatro de estas imágenes eran cuadros del muralista Diego Rivera, que contenían escenas de la conquista de México por parte de España, y la imagen restante era un autorretrato del pintor Juan O'Gorman.

Las imágenes se seleccionaron debido a la cantidad de objetos que contenían (utensilios, herramientas, armas, etc.), así como por la cantidad de emociones que expresaban; de esta manera les era muy sencillo a los hablantes describir detalladamente cada objeto en la pintura y las emociones que les provocaban.

Fue así como cada voluntario habló en promedio entre 2 y 3 minutos, hasta que

[4.17] CHM significa "corpus hecho en México" y el "150" significa el número de hablantes.

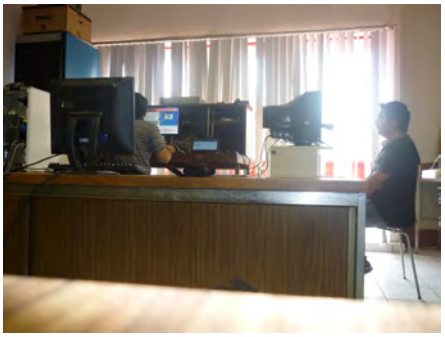

Figura 4.1: Disposición del equipo de grabación del corpus CHM150.

consideraba que ya había descrito todo lo que veía en la imagen. Un registro detallado del hablante fue recabado junto con su firma, dando autorización a utilizar su voz para fines de investigación. Entre los datos recabados se tiene la edad y sexo del hablante, así como su lugar de nacimiento, residencia actual, nacionalidad de los padres y se les pedía especificar los idiomas extranjeros que hablaran, aunque fuera de manera muy básica.

Aquí es preciso puntualizar un error de metodología que ocurrió al grabar. Sucedió que como muestra la figura 4.1, el hablante se sentaba en una silla frente al micrófono. Esto provocaba que la silla crujiera y ciertos hablantes tendían a mover las piernas con nerviosismo, lo cual contaminaba las grabaciones con ruido indeseado. Otro error fue que algunos hablantes conservaron en las manos, la hoja del consentimiento que se les hizo firmar, así como la pluma con la que lo llenaron, y eso provocó otra serie de ruidos causados por el botón de la pluma y el doblar de la hoja por parte del hablante. Lo anterior indica que la mejor manera de grabar es con los hablantes de pie, con la instrucción de estar quietos y sin que lleven nada en las manos.

Para el proceso de edición del audio, que consiste en seleccionarlo, limpiarlo y convertirlo al formato deseado, se utilizó la herramienta de software libre llamada Audacity[4.18]. Fue necesario escuchar todas las grabaciones para determinar que sólo la voz del hablante era la que se escuchaba y que todo fragmento de audio con sonidos de fondo, o más de una persona hablando al mismo tiempo, fuera descartado del corpus.

Después de seleccionar los fragmentos de audio que cumplían con los requerimientos, fueron sometidos a una herramienta de eliminación de ruido del propio Audacity, que tenia por parámetros un tiempo de ataque de 0.15 segundos y una reducción del ruido en 24dB.

A los fragmentos de audio resultantes, finalmente se les asignó un nombre de archivo con el que serían reconocidos dentro del corpus y fueron exportados a un formato tipo

[4.18] Puede descargarse en: http://audacity.sourceforge.net/?lang=es

SPHERE de NIST[4.19], a una frecuencia de muestreo de 16 kHz con muestras de 16 bits. Este formato es ideal para trabajar con el sistema de reconocimiento de voz Sphinx3.

En este punto se seleccionaron los archivos para conformar el corpus de prueba. El criterio de selección consistió en elegir 2 archivos de audio al azar por cada hablante, lo que daba un total de 300 archivos diferentes, es decir, un 11.26 % del corpus total.

Por último, es importante mencionar el equipo de grabación utilizado, todo de la marca Behringer:

- Interfase USB para digitalizar la señal de audio modelo UCA200
- Mezcladora analógica de 5 canales modelo XENYX502
- Micrófono electrodinámico cardioide modelo XM8500

Por lo tanto, según los criterios de clasificación de corpus orales para uso en reconocimiento de voz, el corpus CHM150 se ha medido que cumple con lo siguiente:

- **Origen de los elementos**: Sus elementos provienen de hablantes entrevistados en un pequeño laboratorio con condiciones de ruido similares a las de una oficina tranquila.
- **Codificación y anotación**: El corpus CHM150 fue rendido en formato SPHERE de NIST, con nivel de transcripción únicamente ortográfico y diccionario de pronunciación basado en la versión de 2004 del alfabeto Mexbet.
- **Propósito**: Fue hecho para la creación de modelos de acústicos para uso en sistemas ASR.
- **Lenguaje**: El lenguaje que registra es habla espontánea, de hablantes educados de clase media (todos eran universitarios entre 18 y 39 años de edad)
- **Cantidad de audio**: El corpus CHM150 se compone de 2,363 archivos de audio (1.63 horas), con sus respectivas transcripciones y un diccionario de pronunciación con 1,774 entradas, obtenidas del archivo de transcripción.
- **Accesibilidad**: Se comparte gratuitamente en LDC con el número de item LDC2016S04, a partir de junio del 2016.
- **Representatividad**: Es un corpus de 1.6 horas; por lo tanto es demasiado pequeño para hacer experimentos serios de reconocimiento de voz convencional, sin embargo, puede servir muy bien en experimentos de reconocimiento de voz forense.

[4.19] http://www.nist.gov/

- **Espontaneidad**: Se trata de un corpus de habla espontánea porque fue recabado a partir de entrevistas a personas sin un guión.

Existe más información sobre el corpus CHM150 en (Hernández-Mena & Herrera-Camacho, 2012) y puede solicitarse en:

https://catalog.ldc.upenn.edu/LDC2016S04

4.3.2 Creación del corpus CIEMPIESS

El objetivo principal del corpus CIEMPIESS es el de crear modelos acústicos en español del centro de México, para uso en reconocimiento automático de voz. La distinción de "español del centro de México", hecha para especificar la variante dialectal con la que fue creado el corpus CIEMPIESS se basa en el conocimiento de que el español hablado en otras partes de México, presenta diferencias fonéticas notables (ver (Cuetara-Priede, 2004)).

El corpus está compuesto por un total de 16,717 archivos de audio "limpios", con un promedio de 12 palabras por archivo, extraído de 43 programas de radio que tenían una duración aproximada de una hora, haciendo un total de 17 horas.

El total de archivos con voces femeninas fue de 3,701. Con voces masculinas el total fue de 13,016, lo que representa un 77.86 % de todo el corpus. Este desbalance se debe a que la mayoría de los programas son presentados por hombres y sus invitados son igualmente varones.

Ahora bien, "audio limpio" se refiere a que en cada uno de los archivos sólo se escucha una sola voz fuerte y clara de hombre o de mujer, lo que significa que se eliminó todo el audio que contuviera: música o ruido de fondo, dos o más personas hablando al mismo tiempo, acentos extranjeros y audio con demasiado "shh" o ruido blanco.

Es así como con ayuda de un script en Python, se tomaron las primeras transcripciones ortográficas y se crearon etiquetas (TextGrids) compatibles con el software de análisis de habla llamado PRAAT[4.20]. Se alinearon las etiquetas "a mano" con el audio del CIEMPIESS. Además de alinear, se marcó la vocal tónica y se identificaron los silencios y las disfluencias.

Del archivo de transcripción del corpus CIEMPIESS, se pudieron extraer alrededor de 12 mil palabras sin repetición; sin embargo, el corpus cuenta con dos diccionarios de pronunciación en alfabeto Mexbet, uno en nivel T29 y otro en nivel T66, con alrededor de 50 mil entradas cada uno, que fueron extraídas del archivo de transcripción del CIEMPIESS y de textos de noticias en línea. Ese pequeño corpus textual de noticias

[4.20] Para más información sobre PRAAT consultar: http://www.fon.hum.uva.nl/praat/

tiene una extensión de 1,642,782 palabras (no únicas), que son al menos 10 veces más palabras que las contenidas en el archivo de transcripción del CIEMPIESS (153,456 palabras no únicas). El corpus de noticias sirvió además, para crear el modelo de lenguaje para el CIEMPIESS, que fue hecho por medio de la herramienta de software llamada "the CMU statistical language modelling (SLM) toolkit"[4.21] en formato ARPA[4.22].

La creación del diccionario de pronunciación se hizo con ayuda de scripts en Python y utilizando reglas de grafema a fonema proporcionadas en la tesis del 2004 del maestro Cuétara (Cuetara-Priede, 2004); que fueron actualizadas por el autor en (Hernández-Mena et al., 2014). Dichas reglas se explican detalladamente en el apéndice F, además, en el apéndice G se presentan una serie de funciones en Python que implementan estas mismas reglas.

Por otro lado, en el apéndice H pueden verse detalladamente los símbolos Mexbet para los niveles T29 y T66 utilizados en los diccionarios de pronunciación del CIEMPIESS; en el apéndice I puede consultarse la equivalencia entre los símbolos Mexbet y los símbolos del alfabeto fonético internacional (AFI); y finalmente, en el apéndice C se muestra cómo utilizar el SLM Toolkit para crear modelos de lenguaje de forma automática.

Ahora bien, debido a que el corpus sólo contiene archivos de audio "limpio", no ha sido necesario utilizar demasiadas convenciones en su archivo de transcripción. Se utiliza la etiqueta <**sil**> para indicar las regiones donde hay silencios y la etiqueta ++**dis**++ para indicar disfluencias.

Adicionalmente a esto, se les pidió a los transcriptores que expandieran las abreviaturas, es decir, que las escribieran tal como se pronunciarían, por ejemplo, PRD se expandiría así: "pE Erre dE" o "pErrede" según la velocidad a la que se pronunciara. También se les pidió que transcribieran los números con letra y marcaran la vocal tónica con una mayúscula (ej pErro, gAto, ambulAncia, etc).

Todas las transcripciones fueron escritas en letras minúsculas, con excepción de las vocales tónicas. Otra excepción ocurre con las "N" que se utilizan para representar a la letra "ñ" y con las "W", utilizadas para indicar las "ü". Se decidió no utilizar la letras "ñ" ni "ü", debido a que no pertenecen al alfabeto ASCII regular.

Otra excepción importante respecto a la escritura en minúsculas tiene que ver con los contextos de la letra "x", que en el español de México puede tomar uno de cuatro sonidos diferentes, dependiendo la palabra que la porte, por ejemplo:

- Se pone $ cuando la "x" suena como "s". Como en: xochimilco ≡ $ochimilco
- Se pone KS cuando la "x" suena como "equis". Como en: sexto ≡ seKSto
- Se pone S cuando la "x" suena como el fonema "esh". Como en: xolos ≡ Solos

[4.21] Ver: http://www.speech.cs.cmu.edu/SLM_info.html

[4.22] Ver: http://cmusphinx.sourceforge.net/wiki/sphinx4:standardgrammarformats

- Se pone J cuando la “x” suena como el fonema “xi”. Como en: mexico ≡ meJico

Descartando estas excepciones, todo el corpus se encuentra transcrito en minúsculas y los únicos caracteres no alfanuméricos en el archivo de transcripción son: **$**, <,> y +; sin haber tampoco ningún dígito, ni ninguna letra “ñ” o “ü” en dicho archivo.

Por lo anterior, se concluye:

- **Origen de los elementos**: El audio del corpus CIEMPIESS proviene de programas de radio. Sus diccionarios de pronunciación y su modelo de lenguaje, provienen de un pequeño corpus textual de noticias.
- **Codificación y anotación**: Los diccionarios de pronunciación vienen en alfabeto Mexbet, niveles T29 y T66; el archivo de transcripción viene en letras minúsculas, a excepción de las vocales tónicas, ciertos contextos de la letra “x”, la letra “N” que representa a la “ñ” y la letra “W” que representa a la “ü”.
- **Propósito**: Fue hecho para crear modelos acústicos en español del centro de México para uso en sistemas ASR.
- **Lenguaje**: Se trata de lenguaje espontáneo, porque aunque algunos moderadores de los programas de radio son locutores entrenados, lo cierto es que tienen muchos invitados que no lo son, y por lo tanto, tienen conversaciones no premeditadas en un habla educada.
- **Cantidad de audio y texto**: El CIEMPIESS se compone de 16,717 archivos de audio, de los cuales 3,701 contienen voces femeninas y 13,016 contienen voces masculinas, lo cual hace un total de 17 horas de audio continuo de corpus oral. Los diccionarios de pronunciación en Mexbet T29 y T66 contienen alrededor de 50 mil entradas cada uno, y el modelo de lenguaje se creó a partir de un pequeño corpus textual con 1,642,782 palabras no únicas. El archivo de transcripción del CIEMPIESS contiene 153,456 palabras no únicas y unas 12 mil palabras sin repetición.
- **Distribución de los elementos**: Con base en un conteo de fonemas, se sabe que el corpus CIEMPIESS guarda una proporción de fonemas semejante a la reportada en (Cuetara-Priede, 2004), sin embargo, la proporción de alófonos es deficiente, ya que no todos tienen una buena representatividad.
- **Accesibilidad**: El corpus CIEMPIESS es un corpus libre que puede ser descargado gratuitamente de internet.
- **Representatividad**: Debido a la duración de 17 horas del corpus CIEMPIESS, a que no incluye acentos extranjeros, a que contiene en su totalidad el habla común de la Ciudad de México, y a una ocurrencia suficiente de fonemas, se concluye que es representativo del español hablado en el centro de México.

- **Espontaneidad**: El corpus CIEMPIESS contiene audio espontáneo al 100 %.

Para una información técnica mucho más detallada sobre el corpus CIEMPIESS se puede consultar (Hernández-Mena & Herrera-Camacho, 2014) y para conocer más sobre su arquitectura se puede revisar el apéndice B, y también la página:

http://www.ciempiess.org/CIEMPIESS_Statistics.html

El corpus CIEMPIESS también puede solicitarse al LDC, en la página:

https://catalog.ldc.upenn.edu/LDC2015S07

Capítulo 5

Definición y algoritmo PFS

Las PFS (palabras fonéticamente similares) son motivo de estudio en diversas áreas. En este capítulo se hace una revisión extensa de su estado del arte, se muestra la necesidad de una definición precisa, medible, más funcional, y menos intuitiva de este término, que a su vez permita su estudio en el área de reconocimiento de voz y promueva la inclusión de varios tipos de PFS que no eran alcanzados por otras definiciones previas, pero que pueden tener un impacto real en sistemas ASR.

5.1 Las PFS

En la literatura, existen diferentes acepciones al término "palabras fonéticamente similares", que al parecer, se ha entendido por mucho tiempo de manera intuitiva. Ciertos autores tratan de atribuirles algunas propiedades a estas palabras, pero la mayoría de las veces, dejan como sobreentendido el término "similar" o "confuso" en su definición.

La elección de un término que defina a un conjunto de palabras que, siendo parte del vocabulario de un sistema de reconocimiento automático de voz, puedan ser seleccionadas erróneamente por dicho sistema, y substituidas por palabras distintas de este mismo conjunto, es una tarea que debe realizarse cuidadosamente.

El término de "palabras confusas" puede resultar irónicamente confuso, puesto que no parece ser correcto utilizarlo en un sistema computarizado, ya que la palabra "confusa" denota una falta de claridad sobre algún asunto. Sin embargo, una computadora no puede "confundirse" en el sentido humano de la palabra, puesto que sólo sabe efectuar cálculos matemáticos.

La palabra "fonética" nos habla de alguna manera, sobre la pronunciación de las palabras, lo cual resulta adecuado en el contexto de la definición, puesto que desde el punto de vista de los sistemas ASR, dos palabras con una pronunciación fonética

parecida, tendrán mayores probabilidades de ser intercambiadas erróneamente por el sistema.

Por otro lado, el calificativo de "similar" podría parecer muy subjetivo. No obstante, esta subjetividad desaparece cuando se especifica una manera precisa de medir esa "similitud".

En esta sección se hace mención de diversos términos con los que diferentes autores nombran a las PFS, las maneras en las que se ha abordado su estudio y las soluciones que se han propuesto para evitar que degraden el desempeño de sistemas ASR.

5.1.1 Acepciones al término: "palabras fonéticamente similares"

Con base en la investigación bibliográfica sobre el tema de las PFS, se pueden distinguir dos tipos de paradigmas en el estudio de estas. Uno de ellos es el del procesamiento de señales y el reconocimiento de patrones, mientras que el otro tiene un enfoque más lingüístico.

El término "palabras confundibles" ("confusable words" en inglés (Bahl et al., 1990; Riley & Roe, 1998)) indica que son palabras que pueden ser "confundidas" por otras, por parte de un sistema ASR. Este término parece ser más cercano al área de procesamiento de señales, puesto que sugiere que las palabras son confundidas debido a que estas son representadas por medio modelos de Markov, que a su vez, son parecidos con respecto a cierta métrica.

El término de "vecinos léxicos" se refiere a palabras que varían en un solo fonema (Goldrick et al., 2013; Peereman, 1997). Esta definición es de estilo más lingüístico e incluso psicoacústico. En el caso de un sistema ASR, conforme los fonemas de una palabra van siendo reconocidos, se cargan en memoria todas las otras palabras que comienzan con los fonemas que han sido pronunciados hasta el momento. Por ejemplo, al detectar el fonema /m/ se cargan las palabras "masa", "marimba", "memorama", "memoria", etc. Después, al pronunciar el fonema /e/, cargan sólo "memoria"y "memorama". Estas dos palabras seguirán cargándose hasta el punto: /m e m o r(/, pero cuando se pronuncia el fonema /i/, se sabe en este caso que la palabra es "memoria". En este ejemplo, las palabras "memoria" y "memorama" son vecinos léxicos hasta el punto en que el fonema /i/ es pronunciado.

Ahora bien, es frecuente encontrar el nombre de "word confusability" (Anguita et al., 2004; Chen et al., 2007) ("confusión de palabras"). Es también común referirse a este concepto como, palabras "acústicamente similares" ("acoustically similar words" en inglés (Davis & Mermelstein, 1980; Mills et al., 2004)), en donde, una vez más, el término "acústicamente" trata de hacer referencia unas veces a la idea de fonética y otras a la idea del análisis de los modelos de Markov de cada fonema en dichas palabras.

La mayoría de estos términos podría sustituirse por el de "palabras fonéticamente similares" ("phonetically similar words" en inglés (Davis & Mermelstein, 1980; Mills

et al., 2004)), sin que esto haga mella alguna en la comprensión superficial del término.

En la literatura existe un término muy parecido al de "similar"; se trata de "pares doblemente confusos" o "pares doblemente confundibles" ("double confusable pairs" en inglés), que aparece en (Goldwater & Jurafsky, 2010) y que trata de definir a dos palabras que tienen la misma probabilidad de ser seleccionadas, pero que al mismo tiempo son "fonéticamente similares". Esta definición al menos contempla la probabilidad de selección, que es la otra cualidad más importante que hace que dos o más palabras puedan ser intercambiadas erróneamente, después de la similitud fonética.

En el campo de la lingüística existe también un conjunto de palabras que se suele tomar como excelente ejemplo para estudiar la confusión entre palabras, tanto en humanos, como en sistemas ASR. Se trata de los "pares mínimos", que se definen así, según (Gil Fernández, 2007; Llisterri, 2012):

"Un par mínimo está constituido por dos palabras que difieren sólo por un segmento fonológico situado en idéntico contexto"

Esta definición es muy parecida a la que Buchwald expresa de los vecinos léxicos (Buchwald et al., 2008):

"These competitors - called lexical neighbors - are classically defined as differing by a single segment, with little attention paid to other levels of phonological structure such as featural content or syllabic structure[5.1] *".*

Ambas definiciones son bastante restrictivas. La primera sólo acepta intercambios de un solo fonema en la misma posición, es decir, las palabras "paga" y "paca" sí serían pares mínimos, porque difieren en los fonemas /g/ y /c/ que se encuentran en la misma posición en la palabra. No así las palabras "paga" y "pagas", que no serían considerados pares mínimos por la definición antes dicha.

La definición de los vecinos léxicos es un poco menos restrictiva que la primera, ya que en este caso, "paga" y "paca" entrarían en esta sin problemas, así como también "paga" y "pagas".

En las siguientes subsecciones se muestra cómo es que una idea determinada de PFS, configura marcadamente cierto tipo de análisis en una dirección, demostrando así la necesidad de una definición mucho más general.

[5.1] Estos competidores - llamados vecinos léxicos - son clásicamente definidos por variar en un sólo segmento, poniendo poca atención en otros niveles de la estructura fonológica como el contenido de rasgos característicos o la estructura silábica.

5.1.2 Definición que opera sobre el modelo acústico

Una interpretación que se centra en el modelo acústico del sistema ASR, es el caso de (Printz & Olsen, 2002), en donde se calcula una medida de la tendencia de dos palabras a ser confundidas, por medio del modelo de Markov de cada una.

Para hacer frente a un sistema con muchas PFS, se han creado métricas que permiten comparar directamente HMMs entre sí, con el fin de determinar si son parecidos o no (ver por ejemplo (Basseville, 1989)). Algunos ejemplos de estas métricas aplicadas a sistemas ASR son: la distancia de Bhattacharyya (Olsen & Hershey, 2007), la distancia de Mahalanobis (Jiang et al., 2006), y la divergencia de Kullback–Leibler (Chen et al., 2007; Du et al., 2007).

Entonces, es posible definir con estas métricas el concepto de "similitud" entre dos cadenas de HMM que representen palabras (Chen et al., 2007):

> *"The focus is on defining a word confusability that is accurate in the sense of predicting artificial speech recognition errors"*[5.2]

Es así como el detectar un conjunto de palabras que siendo similares o no fonéticamente, puedan ser confundidas por el sistema y producir errores en el reconocimiento hace posible la creación de estrategias para minimizar su efecto.

El problema con este tipo de paradigma basado en métricas diseñadas para HMM, es que si el sistema no está basado en HMM, estas métricas no podrían ser aplicables.

5.1.3 Definiciones que operan sobre el modelo de lenguaje

Una interpretación del término "palabras fonéticamente similares" de un estilo más lingüístico da lugar a métodos de comparación entre palabras que utilizan la transcripción fonética (llamados algoritmos fonéticos). En la literatura se puede encontrar una gran cantidad de algoritmos fonéticos, como por ejemplo:

- Soundex (Russell & Odell, 1918)
- NYSIIS (Taft, 1970)
- Metaphone (Philips, 1990)
- Phonix (variante de soundex) (Gadd, 1990)
- Agrep (Wu & Manber, 1992)

[5.2] "El objetivo está en definir el concepto de confusión entre palabras, para que sea preciso en el sentido de predecir errores en sistemas de reconocimiento de voz."

- Algoritmos basados en la distancia "Q-Gram" (Ukkonen, 1992; Zobel & Dart, 1996)
- Editex (que es como soundex, pero sus grupos no tienen que estar separados) (Zobel & Dart, 1996)
- Double Metaphone (Philips, 2000)

El más antiguo es el algoritmo Soundex (Russell & Odell, 1918), presentado en 1918 como una manera de codificar nombres propios en inglés, que se pronunciaban igual, pero que su escritura era distinta (por ejemplo, Susy, Sussy, Sussie, etc.).

La estrategia del algoritmo Soundex, consiste en sustituir grafemas o grupos de grafemas pertenecientes a la palabra de entrada, por símbolos que representen una categoría fonética que sea considerada confusa (por ejemplo, las categorías /p/ y /b/ , /t/ y /d/ ó /k/ y /g/). Una vez hecho esto, la palabra de entrada queda transformada en una especie de código. Por tanto, dos palabras cuya pronunciación sea idéntica, tendrán dos códigos Soundex igualmente idénticos.

Una posible adaptación del algoritmo Soundex al idioma español se muestra a continuación:

1. Retener la primera letra de la cadena. Tener en cuenta las letras dobles como CH y LL, ya que se comportan como una sola.
2. Remover todas las ocurrencias de las siguientes letras a partir de la segunda posición: a, e, i, o, u, h, w, y (cuando suena como la vocal i).
3. Asignar números a las siguientes letras (luego de la primera):
   a) b, f, p, v = 1
   b) c, g, j, k, q, s, x, z = 2
   c) d, t = 3
   d) l = 4
   e) m, n = 5
   f) r = 6
   g) ll, y, ch = 7
4. Si hay números consecutivos, dejar solamente uno en la serie.
5. Retornar los cuatro primeros caracteres, si son menos de cuatro, completar con ceros.

Si por ejemplo, se aplica este algoritmo a las palabras “cuatro” y “cuadro”, se obtiene un código Soundex igual a “c360”. Ahora, lo más importante que hay que notar aquí, es que estas dos palabras pueden considerarse como “similares” entre sí, gracias a la categoría fonética 3, que agrupa a los grafemas “d” y “t”.

En principio, la estrategia de Soundex es bastante buena, porque permite predecir y agrupar fácilmente un conjunto de palabras que tendrán muchas probabilidades de ser confundidas. Sin embargo, existen varios estudios que afirman que la precisión de Soundex en su concepción original, es bastante baja, de alrededor del 33 % (Christian, 1998; Stanier, 1990) y es por eso que se han estudiado maneras de mejorarlo, como en (Branting, 2003; Christian, 1998).

No obstante, la idea principal sigue sobreviviendo y ha dado lugar a una gran cantidad de algoritmos basados en esta (ver (Branting, 2003)).

Es un hecho que aunque Soundex agrupa palabras cuya pronunciación es similar, no es capaz de decir en qué grado son dos palabras similares. Para resolver esta cuestión, es necesaria una variante de los algoritmos fonéticos, que se conoce como algoritmos fonométricos. Los algoritmos fonométricos (Zobel & Dart, 1996) son capaces de medir qué tan parecidas son las pronunciaciones de dos palabras, por medio de la transcripción fonética de cada una. Para esto es necesario definir algún tipo de medida.

La medida más famosa es la llamada “distancia de Levenshtein”, que también es conocida como “distancia de edición” o “distancia entre palabras”. La distancia de Levenshtein se define como el número de operaciones requeridas para transformar una cadena de caracteres en otra (Levenshtein, 1966). Las operaciones a las que se refiere esta definición son tres (aplicadas a un carácter): inserción, eliminación y sustitución. Esta medida suele ser utilizada en correctores ortográficos, pero también se ha usado en trabajos de reconocimiento de voz (Reddy & Rose, 2008).

A manera de ejemplo, obsérvese a las palabras “macho” y “cachos”. Si se toma como referente fijo a la palabra “cachos”, se verá que para igualar a esta con la palabra “macho”, hace falta sustituir en “macho” la letra “m” por una letra “c” (1 sustitución) y añadir la letra “s” (1 inserción). Por lo tanto, la distancia de Levenshtein entre las palabras “macho” y “cachos” es dos.

Finalmente, el último elemento faltante en el paradigma de análisis de PFS, enfocándose en el modelo de lenguaje, es la relación entre la medida de similitud entre dos palabras y su relación con el modelo de lenguaje de un sistema ASR determinado.

En un sistema ASR no tiene ninguna utilidad seleccionar todas las palabras del vocabulario y hacer grupos de PFS indiscriminadamente, por medio de un algoritmo fonométrico. Hace falta algún otro elemento distinto a la simple medida de “similitud” entre dos palabras. Ese elemento faltante es la probabilidad de selección de esa palabra, por parte del sistema.

5.1.4 Definición combinada

En (Anguita et al., 2004) se muestra un método que combina un algoritmo fonométrico, con una métrica de comparación entre HMM, con el fin de detectar lo que en este artículo se conocen como "confusable words".

En este método, primero se toma la transcripción fonética de la palabra que se va a analizar y luego esta transcripción se alinea con el HMM que resulte de concatenar los HMM de los fonemas de aquella transcripción, por medio del algoritmo de "Dynamic Time Warping" (DTW). Ahora, para comparar dos palabras se calcula una distancia "interfonema" a partir de los HMM de cada una y también se hace una medición fonométrica, a partir de las transcripciones fonéticas iniciales de cada palabra. Finalmente, con toda esta información, los autores lograron crear un clasificador que evaluaba si dos palabras eran "confusas".

No obstante, se necesita una metodología lo suficientemente general, para no depender de un paradigma de reconocimiento en particular, pero a la vez, lo suficientemente especializada para que pueda tener una aplicación real en la mayoría de sistemas ASR.

5.1.5 Definición del término "palabras fonéticamente similares"

Ni la definición de pares mínimos, ni la de vecinos léxicos son adecuadas para definir a las PFS en un sistema ASR, puesto que las palabras pueden variar en varios segmentos, y no sólo en uno. Una prueba documentada de ello sería la definición de "vecinos remotos" ("remote neighbors" en inglés) que en (Bashford et al., 2006) aparece como: "those differing by two phonemes" (es decir, "aquellos que difieren por dos fonemas").

Por lo anterior, es necesaria una nueva definición de PFS que incluya los casos en que las palabras cambian en más de un fonema, y cuando los fonemas no se encuentran alineados, es decir que no se encuentran en la misma posición dentro de la palabra. También se tiene que tener en cuenta la probabilidad de selección y la definición no debe estar ligada con un paradigma de reconocimiento en particular. Luego entonces, la definición funcional de PFS que se propone en este trabajo es la siguiente:

> ***Se define como palabras fonéticamente similares a un subconjunto de palabras que forman parte del diccionario de pronunciación de un sistema de reconocimiento automático de voz, y que cumplen con las siguientes dos condiciones: La primera es que deben tener una probabilidad normalizada*** p ***de ser seleccionadas por dicho sistema durante una ronda de reconocimiento, en donde*** p ***se encuentra dentro del intervalo cerrado definido por*** ϵ***, tal que*** $\epsilon = [0 \leq \epsilon_1 \leq p \leq \epsilon_2 \leq 1]$***, por lo tanto,*** $p \in \epsilon$ ***y*** $\epsilon = [\epsilon_1, \epsilon_2]$; $p, \epsilon \in \mathbb{R}$***. La segunda condición es que la distancia de Levenshtein*** d ***entre sus transcripciones fonéticas, formadas a***

partir de los símbolos fonéticos en la lista de fonemas del sistema de reconocimiento en cuestión, debe ser estrictamente mayor a 0***, siendo entonces que*** $d \in \mathbb{N}$***.***

Ahora bien, con base en la definición anterior, se propone la siguiente clasificación de PFS para facilitar su estudio por casos:

Dos palabras fonéticamente similares se clasifican con base en su distancia de Levenshtein d***, en donde*** $d \in \mathbb{N}$***, como palabras de grado*** n***, siendo*** $n = d$***. Y al mismo tiempo se clasifican con base en la posición de los símbolos fonéticos que forman las transcripciones fonéticas de cada una como: alineadas y no alineadas. Se dice que dos PFS son alineadas si la transcripción fonética de ambas palabras tiene un mismo número de símbolos fonéticos, es decir, que ambas tienen una misma longitud*** l***; de otro modo se dice que las PFS son no alineadas. Si las PFS son no alineadas, deben cumplir con que la diferencia*** $l - n > 0$ ***en cada una de las transcripciones fonéticas de ambas palabras, de lo contrario se asume que no son PFS.***

Finalmente, se debe hacer una aclaración acerca de los límites de n, para evitar que tome valores sin sentido dentro de la definición:

Dos palabras no pueden considerarse como PFS si su grado $n = 0$***, o si*** n ***coincide con la longitud de la transcripción fonética de alguna de ellas (es decir que*** $n = l$***), sin embargo, la relación entre*** n ***y*** l ***que ayude a imponer un límite superior al valor de*** n***, cumpliendo estrictamente que*** $0 < n < l$ ***de ambas transcripciones, sólo puede ser determinada de manera heurística.***

5.1.6 Características de la definición de PFS

La primera parte de la definición circunscribe a las PFS justo en el contexto de un sistema de reconocimiento real, ya que habla de que estas deben hallarse en el diccionario de pronunciación, lo cual es relevante puesto que las palabras en este diccionario son las únicas que el sistema "conoce", y por lo tanto sería inútil efectuar algún análisis sobre palabras obtenidas de alguna otra parte.

La sutileza de elegir el diccionario de pronunciación como el único proveedor posible de PFS, permite también extender la definición a sistemas más pequeños, que no cuenten con un modelo de lenguaje, como ciertos sistemas de reconocimiento de palabras aisladas basados en gramáticas. Ahora, el caso de un sistema de reconocimiento de

palabras aisladas que no esté basado en diccionario de pronunciación, ni en fonemas, queda fuera del alcance de la definición PFS proporcionada.

La definición de PFS también habla sobre los dos atributos necesarios para que en un sistema existan altas probabilidades de elegir erróneamente una palabra por otra. Se trata en primer lugar de la probabilidad de selección que tiene la palabra durante la etapa de reconocimiento, lo cual es muy importante, puesto que podrían procesarse grandes cantidades de palabras para determinar si son PFS o no, pero a final de cuentas, si estas jamás llegan al sistema en el momento de hacer reconocimiento, entonces el problema de las PFS deja de existir. Obviamente, sin PFS que afecten al sistema, todas las precauciones contra estas resultarían inocuas.

El otro elemento que puede causar problemas en un sistema ASR, además de la probabilidad de selección, es que las transcripciones fonéticas de esas palabras resulten parecidas en cierto grado. Para esto, la definición también ofrece un acotamiento del problema imponiendo que las transcripciones deben ser creadas sólo a partir de la lista de fonemas del sistema. Esto es relevante porque el sistema no "conocerá" otros símbolos fonéticos que no sean los que estén en esa lista.

Además de este acotamiento, la definición ofrece una manera de medir la similitud entre dos palabras. Esto se hace a través de la distancia de Levenshtein que exista entre sus transcripciones fonéticas. La distancia de Levenshtein entrega siempre un entero mayor o igual a cero, que en este caso, representará el número de símbolos fonéticos distintos entre dos transcripciones fonéticas.

También se habla de grados en los que las PFS pueden ser diferentes. Es decir, si la distancia de Levenshtein es de 1, significa que el grado es 1, y por lo tanto, las trasnscripciones sólo varían en un símbolo fonético. Si la distancia es de 2, significa que varían en 2 símbolos, y el grado es 2, y así sucesivamente. Sin embargo, existe una cota inferior, y una cota superior en el grado de las PFS. Si la distancia de Levenshtein entre dos transcripciones es igual a cero, es porque las pronunciaciones de palabras homónimas son exactamente iguales, como "casa" y "caza"; por lo tanto, no son fonéticamente similares, sino fonéticamente idénticas.

El grado tampoco puede ser demasiado grande, ya que si es igual al número de fonemas de alguna de las dos palabras, significará que se trata de palabras demasiado distintas. Sin embargo, el límite máximo del grado dependerá de las condiciones y de la longitud de las palabras analizadas.

Para explicar de una manera más sencilla la definición de PFS propuesta aquí, se recurre a la figura 5.1. Esta muestra un diagrama de flujo en donde se observan las preguntas clave que se deben hacer para determinar si dos palabras son fonéticamente similares, asumiendo que ambas tienen una probabilidad de ser seleccionadas dentro de cierto intervalo ϵ.

Ahora, a manera de ejemplo[5.3], si se desea saber si las palabras "cama" y "gama"

[5.3]En todos los ejemplos de ahora en adelante, a menos que se indique lo contrario, se asume por

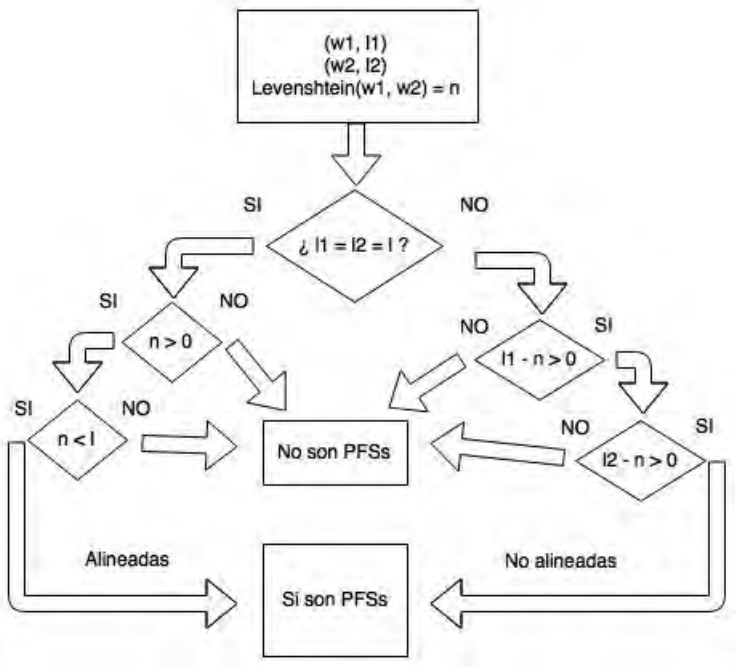

Figura 5.1: Diagrama de flujo que muestra gráficamente la definición de PFS propuesta.

son fonéticamente similares, asumiendo que tienen la misma probabilidad de selección por parte del sistema, se analizarían de la siguiente manera, con base en la definición de PFS:

```
Se genera su transcripción fonética

cama k a m a

gama g a m a
```

Ahora bien, la longitud de ambas transcripciones es de 4 ($l = 4$), y como varían sólo en un fonema, resulta que su grado es igual a 1 ($n = 1$). Así que cumplen con ser palabras fonéticamente similares porque cumplen con la relación $0 < n < l$, y son alineadas porque sus longitudes l son iguales.

Ahora se intentará analizar las palabras “cosa” y “miel”:

```
Se genera su transcripción fonética
```

simplicidad que las palabras analizadas tienen la misma probabilidad de ser seleccionadas por el sistema ASR.

```
cosa k o s a
miel m i e l
```

Estas transcripciones tienen una longitud de 4 y una distancia de Levenshtein también de 4, es decir que $n = l$, así que por definición, no son fonéticamente similares, aunque esto se constata también a simple vista.

Ahora se analizarán las palabras "varón" y "barón":

```
Se genera su transcripción fonética
varón b a r( o_7 n
barón v a r( o_7 n
```

Las transcripciones tienen una longitud de 4, pero la distancia de Levenshtein es de 0 ($n = 0$), por lo tanto sus transcripciones fonéticas son idénticas, lo que significa que en realidad, sólo podrían ser diferenciadas por un sistema ASR, a través de su probabilidad de selección, pero no por su transcripción, así que por definición, no son PFS.

La definición también habla sobre PFS alineadas y no alineadas. El primer ejemplo en donde se analizan las palabras "cama" y "gama", es un ejemplo de PFS alineadas, puesto que se cumple que sus transcripciones fonéticas tengan la misma longitud.

Ahora sólo resta mostrar un ejemplo de PFS no alineadas, y para ello se analizarán las palabras "solamente" y "mente":

```
Se genera su transcripción fonética
solamente s o l a m e n t e
mente m e n t e
```

La primer palabra tiene una longitud de 9, mientras que la palabra segunda tiene una longitud de 5. Ahora bien, la distancia de Levenshtein entre ambas es de 4, porque hay que añadir los fonemas / s o l a / a la segunda palabra, para transformarla en la primera palabra, por lo tanto, $n = 4$. Para analizar si son PFS, ambas tienen que cumplir la relación $l - n > 0$. Para la primera se tiene que $l_1 = 9,\ n = 4\ ;\ l_1 - n\ =\ 9 - 4 > 0$, y para la segunda $l_2 = 5,\ n = 4\ ;\ l_2 - n\ =\ 5 - 4 > 0$; por lo tanto, ambas palabras son PFS aunque no estén alineadas. Sin embargo, se observa que ambas palabras son demasiado diferentes, lo cual es evidenciado por su elevado grado. Una heurística bien razonada nos tendría que informar que "solamente" y "mente" tienen un grado demasiado alto para ser PFS. No obstante, hay que tener cuidado, puesto que otras palabras si podrían sonar más parecidas con una n igual de alta.

Por ejemplo, las palabras “gallitito” y “cabellito”:

```
Se genera su transcripción fonética
gallitito g a Z i t i t o
caballito k a b e Z i t o
```

Estas palabras cumplen con tener la misma longitud $l = 8$, aunque tienen una $n = 4$, lo cual es una n tan grande como en el ejemplo anterior; sin embargo, estas dos palabras tienen a simple vista una mayor similitud que las palabras “solamente” y “mente”. Además, cumplen con ser PFS porque guardan la relación $0 > n > l$ y son alineadas porque sus longitudes son iguales.

5.1.7 Casos de estudio

Un par mínimo puede explicarse a partir de la definición de PFS como dos PFS alineadas de grado 1. Así mismo, un par de vecinos remotos o “vecinos léxicos remotos” puede definirse exactamente como dos PFS alineadas de grado 2. El término más general de “vecino léxico” también encuentra cabida en la definición de PFS como dos PFS alineadas o no alineadas, pero de grado 1.

La definición de PFS también puede ayudar a detectar cierto tipo de palabras que no son PFS, y en las que la distancia de Levenshtein entre sus transcripciones fonéticas es exactamente igual a cero. Se trata de la siguiente lista de tipos de palabras, cuyo recuento y ejemplos aparecen en (Garrido-Galindo & Herrera-Camacho, 2011):

- Palabras polisémicas: Palabras que se escriben igual, pero que varían en significado. Como “banca” de asiento, o “banca” de conjunto de bancos.
- Palabras homónimas: Palabras con etimologías distintas, pero que se escriben igual. Como “vino” del verbo venir, o “vino” de bebida.
- Palabras homógrafas: Palabras homónimas que se escriben de la misma manera. Como “haya” de árbol, o “haya” del verbo haber.
- Palabras homófonas: Palabras que no se escriben de la misma manera, pero que se pronuncian de la misma forma. Como “tuvo” del verbo tener, o “tubo” de cañería.

Hasta ahora, pareciera que sólo las PFS alineadas son de verdadera utilidad, sin embargo, las PFS no alineadas también tienen su lugar dentro de los sistemas ASR, especialmente a causa de una situación particular en la etapa de reconocimiento. Después

de que el sistema ASR toma un audio de prueba y lo convierte a vectores de características, para luego poder compararlos con el modelo acústico, entrega al final de este proceso un tren de símbolos fonéticos continuo, es decir, que suele no indicar silencios entre palabras, para luego mapearlo a palabras del diccionario de pronunciación, por medio del modelo de lenguaje. Por lo tanto, es muy común que ocurra que una palabra termine con el mismo símbolo fonético con el que empieza la palabra siguiente, es decir una especie de sinalefa[5.4] que no sólo incluye vocales. Por ejemplo:

```
Se transcribe el fragmento de oración: "galletas saladas".

La transcripción esperada en un sistema ASR sería:
g a Z e t a s s a l a d a s

La transcripción que realmente entrega un sistema ASR es:
g a Z e t a s a l a d a s
```

Esta situación es muy común en reconocedores estándar. Por lo tanto, ocurre que palabras como “galletas” y “galleta”, o “salada” y “saladas”, resultan ser PFS de grado 1, pero no alineadas. Mientras que las palabras “saladas” y “alada” son PFS no alineadas de grado 2.

Ahora bien, con respecto a esto, se puede ilustrar una situación que podría ser detectada desde el modelo de lenguaje de un sistema ASR, con ayuda de un algoritmo basado en la definición de PFS. Se trata de grupos de pares de palabras, en donde justo como en el ejemplo anterior, la primera palabra termina con la misma letra con la que comienza la segunda, así como en: “el loro” y “el oro” o “un narco” y “un arco”.

En un modelo de lenguaje de n-gramas, estos pares de palabras serían conocidos como bigramas. Por lo tanto, si por medio de un algoritmo sencillo se agrupa la letra final de la primera palabra, con la letra inicial de la segunda, en una sola letra, podrían ahora buscarse otros bigramas que tengan una distancia de Levenshtein igual a cero, con respecto al primer bigrama. De este modo sería posible detectar esta especie de “bigramas fonéticamente similares” desde el modelo de lenguaje, para así prevenir que afecten al sistema. Un ejemplo de esto sería:

Se detecta el bigrama: “el loro” en el modelo de lenguaje, cuya transcripción debería de ser:

[5.4] Estrictamente una sinalefa es la unión de la última vocal de una palabra, con la primera vocal de la palabra siguiente.

```
e l l o r( o
```

Pero al someterlo al algoritmo sencillo planteado arriba, se tendría la "palabra": "eloro" cuya transcripción sería:

```
e l o r( o
```

Si se busca otro bigrama con distancia de Levenshtein cero, con respecto a la transcripción anterior, se encontraría el bigrama: "el oro", cuya transcripción fonética es:

```
e l o r( o
```

Por lo tanto, los bigramas "el oro" y "el loro" serían fonéticamente similares

5.1.8 Las PFS y el desplazamiento acentual

Otro grupo de palabras que es explicado por la definición de PFS, es el de palabras despojadas de su acento ortográfico aunque lo necesiten, y que tienen una transcripción fonética exactamente igual. Un ejemplo es: "médico", "medico" y "medicó", ya que sin acento ortográfico, las transcripciones de estas tres palabras serían /m e d i k o /. En lingüística, a este fenómeno se le conoce con el nombre de desplazamiento acentual.

El problema del desplazamiento acentual se aminora al marcar las vocales tónicas, como ocurrió durante la creación del corpus CIEMPIESS. De esta manera, es importante que las palabras con desplazamiento acentual se incluyan en la definición establecida de PFS.

Respecto a la definición de PFS, dos palabras que sin considerar su vocal tónica tengan una transcripción fonética igual, pero que considerándola la tengan diferente, pueden ser consideradas como palabras fonéticamente similares alineadas, de grado 1. Por ejemplo:

```
1. "sabia" (mujer sabia)
s a_7 b i a
```

```
2. "savia" (de los árboles)
s a_7 b i a

3. "sabiá" (nombre de ave)
s a b i a_7

4. "sabía" (del verbo: saber)
s a b i_7 a
```

En este ejemplo, resulta que las palabras 1 y 2 no son PFS entre sí, porque tienen transcripciones fonéticas idénticas, sin embargo, en las palabras 1, 3 y 4, y también 2, 3 y 4, se observa que la única diferencia es la posición de la tónica, por lo tanto, se trata de PFS alineadas de grado 1. Hay que notar también que si no fuera por la vocal tónica (representada por el símbolo fonético "_7") las 4 transcripciones serían totalmente idénticas.

5.1.9 Algoritmo PFS

Este trabajo está enfocado al estudio de PFS alineadas de grado n, en donde se incluyen las que presentan desplazamiento acentual. Por lo tanto, el algoritmo PFS[5.5] que se plantea, es un algoritmo basado en Soundex. Este nuevo algoritmo fue creado para agrupar PFS alineadas de cualquier grado, a partir de una lista de entrada de palabras en español que tienen marcada su vocal tónica.

En la literatura, este tipo de algoritmos se conocen como de "phonetic matching" (Zobel & Dart, 1996) en inglés, o de "equiparación fonética". Las reglas de la equiparación fonética comienzan con una lista de categorías fonéticas, que son las siguientes:

```
1 p b
2 t d
3 k g
4 tS S
```

[5.5] Este algoritmo PFS lo desarrollé en parte, con la ayuda de la licenciada en letras hispánicas Alejandra Chavarría, quien trabajó en la definición de categorías fonéticas para el algoritmo.

```
5 m n
6 Z n~
7 r( r l
8 f s x
A a_7
E e_7
I i_7
O o_7
U u_7
9 a e i o u
```

Nótese que en esta lista de categorías fonéticas hay 14 filas. En la fila 1 están los elementos 1, p y b, mientras que en la fila 14 se encuentran los elementos 9, a, e, i, o, u.

La interpretación de la lista de reglas es muy sencilla: El símbolo de la primera columna es el símbolo que representará a la categoría fonética y las demás columnas contienen símbolos que serán sustituidos por este. De este modo, en la primera fila, las símbolos “p” y “b” serán sustituidos, en cualquier transcripción de entrada al algoritmo PFS, por el símbolo “1”.

Este algoritmo entrega una lista de grupos de palabras con el mismo código PFS. Por lo tanto, entregaría un archivo de texto similar al siguiente:

```
#------------------------#
79-19-8A7-798   rebasArlas
79-19-8A7-798   revisArlos
#------------------------#
99-5E5-29   aumEnta
99-5E5-29   aumEnto
#------------------------#
89-19-7A-598   separAmos
89-19-7A-598   soberAnos
#------------------------#
```

En donde se aprecia el código PFS con guiones intermedios, los cuales son producto de la transcripción fonética, ya que esta se encuentra dividida en sílabas.

Ahora, con estos datos se puede plantear el algoritmo que obtiene grupos de PFS, el cual es:

```
Algoritmo PFS

1: define algoritmo_PFS(lista_de_palabras_de_entrada, lista_de_reglas):
2: {

3:   For cada palabra de lista_de_palabras_de_entrada:
4:   {
5:         trans = transcribe_a_T29(palabra)
6:      For cada línea de la lista_de_reglas:
7:      {
8:         For cada elemento de la línea actual de la lista_de_reglas,
           a partir del elemento 1:
9:         {
10:           codigo_PFS = En trans sustituye(elemento[n], por elemento[0])
11:        }
12:        Añade el codigo_PFS a lista_PFS
13:     }
14:  }

15:  Ordena alfabéticamente a lista_PFS
16:  var = lista_PFS(0)
17:  añade var a lista_grupo_PFS

18:  For cada elemento de lista_PFS a partir del elemento 1:
19:  {
```

```
20:      if elemento_actual == var:
21:         añade elemento_actual a lista_grupo_PFS
22:      else:
23:         imprime lista_grupo_PFS
24:         imprime separador de grupo PFS (#----------#)
25          limpia lista_grupo_PFS
26:         var = elemento_actual
27          añade var a lista_grupo_PFS
28:      endif
29:   }


30: }
```

Este algoritmo es bastante eficiente; computacionalmente es casi de tiempo lineal, ya que sólo visita cada elemento de la lista de entrada una vez para transcribir fonéticamente, y luego otra vez para formar los grupos. El elemento que hace que este algoritmo no sea completamente de tiempo lineal es la ordenación alfabética, que típicamente se hace en tiempo $nlog(n)$.

Es preciso recordar que dos palabras no se consideran como PFS si su probabilidad de selección de ambas no pertenece a cierto intervalo ϵ, lo cual, en condiciones de un sistema ASR real, reduce enormemente las palabras analizadas.

La manera de incorporar el elemento de "probabilidad de selección" al algoritmo PFS se discute en la sección 5.2.3.

5.1.10 Diccionario PFS

La presentación planteada en esta tesis como ideal para mostrar el resultado de un algoritmo PFS y utilizarlo en un sistema ASR es la del diccionario PFS.

Un diccionario PFS es un archivo de texto plano, en donde cada línea represente a un grupo de PFS distinto. La primera columna de este archivo, es decir, la que se forma con todas las palabras que están más pegadas al margen de la izquierda, debe estar ordenada alfabéticamente. Ahora bien, en una fila en particular del diccionario se encuentra la primera PFS pegada al margen izquierdo, pero luego, separadas por espacios, deben colocarse todas las demás palabras que sean fonéticamente similares a la primera, es decir, su cohorte. Y lo mismo debe ocurrir en cada línea del diccionario.

Tabla 5.1: Ejemplo esquemático de un diccionario PFS

PFS_1.1	PFS_1.2	PFS_1.3	. . .
PFS_2.1	PFS_2.2	PFS_2.3	. . .
PFS_3.1	PFS_3.2	PFS_3.3	. . .
.			
.			
.			

Si ocurre que una palabra en particular no cuenta con una cohorte de palabras fonéticamente similares a esta, significará que no es fonéticamente similar a ninguna palabra del sistema y, por lo tanto, no es PFS; por lo cual, no debe ser incluida en el diccionario PFS. Un ejemplo esquemático de cómo debe verse un diccionario PFS puede apreciarse en la tabla 5.1.

Otro detalle importante es que todos los miembros de la cohorte de una PFS deberían de ir, de preferencia, ordenados de izquierda a derecha, en orden de probabilidad de selección, es decir, que en la tabla 5.1 la PFS_1.2 debería tener mayor probabilidad que la PFS_1.3, y así sucesivamente. De esta manera, resultaría muy fácil tanto para el sistema como para un humano hacer consultas al diccionario PFS.

Finalmente, en la primera columna deben ir todas las palabras que hayan aparecido al menos una vez, en algún grupo PFS. De este modo, el efecto es que la primera columna tendría palabras únicas, mientras que en las demás columnas podría haber palabras que se repitieran en varias filas, lo cual debería ser considerado como normal.

5.2 Técnicas de creación del diccionario PFS

En esta sección se hace mención de técnicas documentadas en la literatura destinadas a generar una lista de PFS que fácilmente podría ser colocada en forma de un diccionario PFS.

En la sección 5.2.1, se muestra una manera de cómo generar esta lista de PFS en tiempo de reconocimiento.

En la sección 5.2.2 se muestra cómo crear esta lista, pero en tiempo de diseño, es decir, que se hace durante la configuración del sistema ASR, con ayuda del modelo de lenguaje.

Finalmente, en la sección 5.2.3, se presenta la manera en que la definición de PFS

crea un diccionario PFS en tiempo de diseño, aplicando la técnica diseñada por el autor y denominada “cortes al modelo de lenguaje”.

5.2.1 Creación dinámica del diccionario PFS

En el artículo titulado “A fast approximate acoustic match for large vocabulary speech recognition” (Bahl et al., 1993), se propone una técnica para generar dinámicamente una lista de PFS obtenidas directamente del audio de prueba, durante la etapa de reconocimiento.

Cuando un audio es reconocido por el sistema, este arroja un tren de fonemas que después es mapeado a palabras del diccionario de pronunciación, por medio del modelo de lenguaje. Ahora bien, este tren de fonemas carece regularmente del fonema de silencio, que de hecho, se exige forme parte de la lista de fonemas.

Por tanto, sin este fonema de silencio en el tren de fonemas, resulta complicado saber cuántas palabras están contenidas en este. Así que una manera de hacerlo, es ir cargando en memoria todas las palabras que vayan apareciendo hasta el momento. Por ejemplo:

```
El tren de fonemas de la frase:
"la telaraña de la araña" sería


l a t e l a r( a n~ a d e l a r( a n~ a
```

Cuando de izquierda a derecha se revisa el fonema /l/, se cargarían todas las palabras o ”vecinos léxicos”que comiencen con ”l”:

```
la
labor
lacra
lelo
lila
...etc
```

Cuando se llega al fonema /a/ sólo se cargan los vecinos léxicos que comiencen con "la"

```
la
labor
lacra

... Y así sucesivamente.
```

El método de Bahl toma a todos estos vecinos léxicos y hace lo que él denomina "fast match", es decir, con ayuda de lo modelos de Markov de esos vecinos léxicos, calcula rápidamente la probabilidad de que un grupo de estos vecinos sean PFS y, si es así, los guarda en una lista, que no debe ser mayor a cien elementos. Por lo cual, se deduce que la lista debe contener principalmente palabras clave, si es que empezara a crecer demasiado.

Para el momento en que la lista es creada, ya no es necesario hacer el "fast match" en rondas de reconocimiento subsiguientes, las cuales, deberían de ser mucho más precisas gracias a que el efecto nocivo de las PFS se vería ahora mitigado en el sistema.

Los contras que saltan a la vista en este algoritmo son que, en un sistema de reconocimiento de voz, como en un sistema de dictado por ejemplo, en donde los usuarios dicen lo que sea, probablemente este sistema no acabaría nunca de realizar el "fast match", sin mencionar que esta técnica quedará obsoleta en poco tiempo, debido a la llegada de los sistemas basados en "deep learning", como el sistema Kaldi[5.6].

No obstante, en condiciones más controladas de reconocimiento, esta técnica puede resultar bastante cómoda para el diseñador del sistema, y apropiada para detectar PFS poco comunes, que estén fuera de la definición de PFS de este trabajo.

5.2.2 Creación estática del diccionario PFS

En el artículo "Language model and acoustic model information in probabilistic speech recognition" (Ferretti et al., 1989), Ferreti utiliza el modelo de lenguaje para escoger palabras, luego usa los HMM de esas palabras para hacer una búsqueda rápida (una "fast match" como en la sección 5.2.1), determina cuales de ellas son fonéticamente similares[5.7] y, después, hace una búsqueda mucho más detallada (una "detailed match"), pero sólo sobre las palabras seleccionadas en el paso anterior.

[5.6] Por ahora, Kaldi soporta los dos paradigmas: "deep learning" y HMM

[5.7] Los autores usan la acepción: "acoustically most likely words" en inglés, o "palabras más parecidas acústicamente" en español.

Es pertinente mencionar que este autor nos presenta un método para calcular una medida, a la que llama "Speech Decoder Entropy" (SDE), que permite calcular la relación entre el modelo acústico y el modelo de lenguaje; lo cual, entre otras cosas, ofrece la posibilidad de tener una medida más precisa de qué tan predictivo puede ser o no un modelo de lenguaje, mediante una medida diferente a la de la perplejidad.

5.2.3 Cortes al modelo de lenguaje

La técnica de cortes al modelo del lenguaje es la alternativa presentada en esta tesis para generar un diccionario PFS de manera estática, en tiempo de diseño; sin necesidad de recurrir a los HMM del modelo acústico, y basándose en la definición de PFS.

Esta técnica tiene en consideración dos casos principales: el primero es cuando no se tiene predilección por ninguna palabra del vocabulario y se desea hacer un agrupamiento de todas las PFS posibles de alguno, o de todos los tipos posibles. El segundo ocurre cuando se desea que el sistema ponga atención sobre algún conjunto de palabras debido a las condiciones del problema.

Es claro que el primer caso es más adecuado cuando se diseñan sistemas ASR de propósito general, del tipo continuo, de habla espontánea e independientes del hablante, ya que en sistemas así, cualquier usuario debería poder ser capaz de decir cualquier cosa (o casi cualquier cosa), y el sistema debería poder responder, al menos parcialmente.

El segundo caso resulta ideal para sistemas que operen en condiciones más controladas, en las que de antemano se conozca el vocabulario que será requerido, y cuando se desea estar seguro de que no habrá casos en los que las PFS darán problemas.

Ahora bien, es preciso aclarar que la técnica de cortes al modelo de lenguaje se aplica a cualquier tipo de modelado de lenguaje, y no sólo al que está basado en n-gramas. Lo único que esta técnica requiere es poder medir la probabilidad de selección, por parte del sistema, de las palabras que se desean analizar; es decir, la probabilidad a priori $P(W)$. También se requiere que el sistema esté basado en fonemas.

Por tanto, aplicando la definición PFS en el primero de los dos casos, será preciso tomar todas las palabras del diccionario de pronunciación, calcular su probabilidad normalizada p, y ponerlas en una lista ordenada por probabilidad, en donde cada renglón sea un par (probabilidad, palabra), esto es:

```
p1 w1
p2 w2
p3 w3
...etc
```

Entonces, es preciso especificar el tamaño del "corte" con el que el modelo de lenguaje será "rebanado" , es decir, con múltiples cortes equidistantes de anchura ϵ. Para lograr esto, hay que definir un intervalo de probabilidad ϵ, a través de dos valores ϵ_1 y ϵ_2, tales que: $\epsilon = \{0 < \epsilon_1 < \epsilon_2 < 1\}$ en donde $\epsilon, \epsilon_1, \epsilon_2 \in \mathbb{R}$.

De esta manera, es posible definir un algoritmo que agrupe todas las palabras que pertenezcan a un intervalo de probabilidad en particular, mientras ese intervalo se recorre a lo largo de todo el modelo de lenguaje, pasando por la palabra con menor probabilidad, hasta llegar a la palabra con la mayor probabilidad. Este proceso se ilustra en el siguiente pseudocódigo:

```
Algoritmo de Cortes al  Modelo de Lenguaje

 1: Lista_P = Lista_de_Palabras_ordenadas_por_Probabilidades.txt;

 2: Define Cortes_ML(Lista_P):
 3: {

 4:   ep1 = <probabilidad de la palabra con menor probabilidad>;
 5:   ep2 = <tamaño del corte definido por el usuario>;
 6:   epsilon = ep2 - ep1;
 7:   Añade elemento Lista_P(0) a Lista_Grupo;

 8:   For cada elemento de la Lista_P a partir del elemento 1:
 9:   {
10:     if prob_actual esta en epsilon:
11:       Añade palabra_actual a Lista_Grupo;
12:     else:
13:       Imprime Lista_Grupo;
14:       Imprime Separador("#---------#");
15:       Limpia Lista_Grupo;
16:       Añade palabra_actual a Lista_Grupo;
17:       epsilon = epsilon + ep2; //Se avanza al siguiente corte
```

```
18:     endif
19:   }
20: }
```

Este algoritmo da como resultado una lista de grupos de palabras separadas por "#- - - - #", en donde cada grupo se encuentra dentro de un intervalo de probabilidad bien definido. Una vez hecho esto, sería posible aplicar nuestro algoritmo PFS a cada grupo.

Finalmente, el segundo caso es mucho más sencillo que el primero. Se desea que el sistema ponga atención a ciertas palabras en particular, y se requiere buscar PFS con respecto a estas. Sólo se necesita especificar un intervalo ϵ fijo y recorrer la lista de palabras ordenadas por probabilidad descrita arriba, para ir "recogiendo" todas aquellas palabras que se encuentren dentro del rango ϵ, para aplicarles al final, el algoritmo PFS.

5.3 Soluciones al problema de las PFS

A lo largo de este capítulo se ha propuesto y analizado una definición del término palabras fonéticamente similares, mostrándose varias maneras de detectar este tipo de palabras, incluyendo la técnica de cortes al modelo de lenguaje. También se ha explicado que para añadir una lista de PFS a un sistema ASR, debería de hacerse mediante la creación de un diccionario PFS, y se han discutido varias maneras maneras de generarlo, unas de manera dinámica en tiempo de reconocimiento, y otras de manera estática en tiempo de diseño del sistema.

En las siguientes secciones se discuten algunos de los problemas que podría enfrentar un sistema ASR con una gran densidad de PFS, se presentan algunas maneras documentadas de qué hacer con la lista de PFS para resolver problemas en el sistema, y se presentan algunos ejemplos de aplicación para las PFS, después de haber sido detectadas.

5.3.1 Posibles causas de un aumento de PFS en el sistema

Hablando estrictamente de sistemas ASR, existen tres situaciones particulares por las que podría aparecer una densidad de PFS mayor a la esperada, y que deberían tomarse en cuenta durante el diseño del sistema, debido a que es posible detectarlas mucho antes de su implementación. Las tres situaciones son:

- Cuando el sistema no está lo suficientemente entrenado: Esto provoca que los

modelos acústicos sean muy poco específicos, lo cual provoca que confundan frecuentemente un fonema por otro.

- Cuando el medio ambiente absorbe o filtra ciertas frecuencias en el rango de la voz: Por ejemplo, si se habla a través de un muro, quizá sea muy fácil que las altas frecuencias se vean debilitadas, por lo tanto, es posible que al hacer reconocimiento en sistemas así, ciertos fonemas se vean comprometidos o incluso, que desaparezcan totalmente de la cadena hablada.
- Cuando en un sistema basado en gramáticas se escoge un vocabulario fonéticamente similar: Lo cual puede suceder en un vocabulario especializado.

5.3.2 Mejoramiento del modelo de lenguaje

Como se explica detalladamente en la sección 5.2.1, un tren de fonemas entregado por el sistema ASR, es luego mapeado a palabras del diccionario de pronunciación con ayuda del modelo de lenguaje. Por lo tanto, de aquí se deduce la importancia de tener un buen modelo de lenguaje.

Un buen modelo de lenguaje soluciona, hasta cierto punto, el problema de las PFS, debido a que es muy común que algunas de ellas nunca interaccionen entre sí, por causa de su probabilidad de selección.

Un ejemplo de esto es la palabra "tratamiento", que de alguna manera contiene las palabras "trata" y "miento". Ahora bien, estas tres palabras se encontrarían en contextos muy distintos dentro de un modelo de lenguaje extraído de textos reales en español.

Suponiendo que el sistema trabaja con un modelo de lenguaje de bigramas, y según las oraciones de ejemplo, habría algunos como:

```
el tratamiento

tratamiento es

la trata

trata de

no miento

miento en

...etc
```

Por lo tanto, para que un sistema relacionara a "trata" y "miento" como palabras independientes, y una seguida de la otra, tendría que haber bigramas como: "trata miento", lo cual, resulta muy improbable.

Sin embargo, si ocurre el caso en el que el sistema no cuenta con la palabra "tratamiento" en su diccionario de pronunciación, pero sí contara con las palabras "trata" y "miento", es seguro que al escuchar "tratamiento", el sistema resolviera poner "trata" seguido de "miento".

En el caso inverso, en el que el sistema no contara ni con "trata", ni con "miento", pero si con "tratamiento", es muy probable que al escuchar sólo la palabra "trata", el sistema la sustituyera por "tratamiento".

Por todo lo anterior, la conclusión es que la primera medida a tomar en un sistema ASR cualquiera, es mejorar el modelo de lenguaje, sin embargo, si esto no funciona, se puede recurrir a alguna técnica de tratamiento de PFS, como las que se discuten en las siguientes dos secciones.

5.3.3 Buscar sinónimos

En el artículo "Confusable word detection in speech recognition" (Riley & Roe, 1998), se presenta un método para encontrar PFS que consiste en qué, después de que el sistema transforma una palabra entrante en un tren de fonemas, lo compara con el tren de fonemas de otra palabra diferente. La comparación entre las dos palabras produce un "confusability index", producido a partir de los HMM de ambas, y que los autores muestran cómo calcular. Este índice representa una medida que predice qué tanto serán intercambiadas erróneamente ambas palabras por el sistema ASR.

De esta forma, analizando todas las palabras deseadas, se crea una lista de PFS, con la cual los autores deciden buscar sinónimos que eviten la confusión cuando el sistema esté funcionando. Por lo tanto, este método sirve para crear una lista de PFS en tiempo de diseño.

En conclusión, una solución perfectamente válida para eliminar la influencia negativa de las PFS, es suprimirlas mediante sinónimos.

5.3.4 Crear un módulo rápido de análisis de PFS

En el artículo "Constructing groups of acoustically confusable words"(Bahl et al., 1990), se discuten varios métodos para crear listas de PFS en tiempo de diseño. Sin embargo, la técnica presentada aquí es distinta a las anteriores. Primero se hace una lista "pre-computada" de posibles PFS. Se dice que es "pre-computada" porque la lista se deduce de manera independiente, sin involucrar al reconocedor. Luego, como se asume que esas palabras detectadas son de cuidado por ser posibles PFS, se pone a diferentes locutores a grabarlas.

Después de grabar esta lista de palabras pre-computadas, se vuelve a verificar que dichas palabras sean PFS, pero ahora por medio de sus HMM. Si alguna lo es, se incluye

su HMM en el sistema, hasta añadir así, todas las palabras que resultaron ser PFS.

Finalmente, cuando el sistema se encuentra en tiempo de reconocimiento y detecta alguna de las PFS cuyo HMM se incluyó en el sistema durante el paso anterior, pasa a ser analizada de manera más minuciosa por medio de un pequeño reconocedor rápido basado en los HMM añadidos en el paso anterior.

Por lo tanto, este pequeño módulo de reconocimiento "rápido", es la solución dada en este artículo para el tratamiento de las PFS en un sistema ASR.

Sin embargo, los autores especifican que sólo se graban las palabras que se consideran importantes para el sistema, ya que podría resultar muy costoso, computacionalmente hablando, el hecho de que todas las palabras pasen por el reconocedor rápido.

Para crear la lista de PFS pre-computada se utiliza un árbol de pronunciaciones que, esencialmente resulta ser un método muy parecido al algoritmo PFS diseñado en esta tesis, puesto que ambos son algoritmos fonéticos que detectan PFS para un sistema ASR.

5.3.5 Aplicaciones posibles del algoritmo PFS

En esta sección se muestran algunos ejemplos documentados de sistemas en los que podrían tener aplicación todos, o algunos de los conceptos sobre PFS discutidos a lo largo de este capítulo.

- En (Lambert et al., 1999) y (Kondrak & Dorr, 2004) se muestran estudios que tienen que ver con errores provocados por lo parecidos que son para las personas, los nombres de ciertos medicamentos y compuestos farmacéuticos. Naturalmente, en estos artículos se presentan estudios para detectar y medir de algún modo la cantidad de estos nombres confusos, por lo que, de alguna manera requieren construir listas de PFS.

- En (Branting, 2003) se presenta un sistema a nivel de texto que analiza en documentos legales si diferentes nombres de marcas o de agencias resultan ser o no los mismos, a pesar de que se escriban de manera distinta como "I.B.M" o "IBM", "Procter and Gamble" o "Procter & Gamble", etc. Aquí, mediante algoritmos fonométricos y ciertas heurísticas, analizan si dos nombres son o no son el mismo. Claramente, habrá una relación con el tema de las PFS, sobre todo cuando este tipo de técnicas pase al dominio del reconocimiento automático de voz.

- En (Gálvez, 2006) y (Gonzales-Cam, 2008) se muestran algoritmos similares al anterior, pero donde los datos no provienen de documentos legales, sino de de sistemas de recuperación de información. Es decir, que el usuario hace una consulta en algún sistema computacional, y este debe regresarle resultados con base en su búsqueda, sin que importe demasiado si el usuario tecleó correctamente el nombre

propio que deseaba encontrar. Aquí le dan uso a algoritmos basados en Soundex. La aplicación que se esperaría en un sistema ASR, es que el sistema tolere que el usuario no pronuncie muy bien algún nombre, y que aún así el sistema tenga cierta capacidad para encontrarlo.

- En (Elmagarmid et al., 2007) se muestra una recopilación de técnicas ("survey" en inglés) utilizadas para detectar registros repetidos en bases de datos. Aquí se hace un recuento de varios algoritmos fonéticos como el de Soundex, y de otros más de tipo fonométrico. Por lo tanto, es claro que de alguna manera se persigue lo mismo que en este capítulo: detectar PFS y solucionar con ello todos los problemas que ellas causan en cualquier sistema, y no sólo en sistemas ASR.

Resulta interesante observar los diferentes ambientes en los que se han aplicado técnicas de detección de PFS, aunque no todas necesariamente han funcionado en un contexto de reconocimiento de voz. Por ahora, muchos de los problemas que se han resuelto a nivel de texto, en el futuro podrán resolverse por medio del reconocimiento automático de voz.

Capítulo 6

Experimentos

En este capítulo se describen una serie de experimentos que tienen como objetivo validar las diferentes herramientas que fueron construidas para esta tesis. También se demuestran de manera experimental, las ideas presentadas en el capítulo 5 sobre la manera de cómo detectar palabras fonéticamente similares y cómo se aplican las PFS en un sistema ASR. Por lo tanto, los experimentos están divididos en las siguientes categorías:

- Experimentos para evaluar el comportamiento de sistemas ASR con el corpus CIEMPIESS.
- Experimentos para evaluar las herramientas automáticas de transcripción fonética.
- Experimentos para evaluar aspectos del modelo de lenguaje.
- Experimentos para evaluar aspectos del algoritmo PFS.
- Experimentos para evaluar la influencia de las vocales tónicas en el reconocimiento de voz.

6.1 Evaluación de sistemas ASR

En esta sección se muestran diferentes experimentos realizados utilizando el corpus CIEMPIESS. Se muestra su portabilidad para ser utilizado en diferentes sistemas de reconocimiento de voz.

Se muestran experimentos de reconocimiento en modo "batch"[6.1], para los sistemas

[6.1] El reconocimiento de voz por lotes o en modo "batch" se refiere a hacer la conversión de voz a texto de un conjunto de audios grabados previamente.

Sphinx3, HTK y Kaldi; así como experimentos en modo "live"[6.2] con los sistemas ASR Sphinx4 y Julius. Esta selección no es fortuita, ya que los sistemas Sphinx3, HTK y Kaldi fueron diseñados para trabajar en modo batch, y los sistemas Sphinx4 y Julius fueron diseñados para trabajar en modo live.

Como lo especifica el apéndice B, sobre las características y arquitectura del corpus CIEMPIESS, el corpus original se compone de un total de 16,717 grabaciones de voz con sus transcripciones. Esas grabaciones fueron divididas en el conjunto de entrenamiento (16,017 grabaciones) y en el conjunto de pruebas (700 grabaciones). Finalmente, al conjunto de pruebas se le añadieron otras 300 grabaciones de otros corpus, junto con sus transcripciones, lo que hace un total de mil frases en el conjunto de pruebas.

La tabla 6.1 muestra la distribución de fonemas y alófonos en el nivel T66 de Mexbet, de las 16,717 frases que componen el corpus CIEMPIESS. En esta se puede observar cómo algunos alófonos tienen el problema de tener muy pocas repeticiones, como por ejemplo el alófono /l_j/ que tiene apenas una repetición, o el alófono /j_7/ que no tiene ninguna. El fonema /tl/ sólo aparece una vez en todo el corpus CIEMPIESS. La poca frecuencia de algunos alófonos en el nivel T66 de Mexbet es una de las razones por las que la mayoría de los experimentos de esta tesis se hicieron sólo en el nivel T29.

La tabla 6.2 muestra una comparación entre los fonemas del nivel T22 en el corpus DIMEx100, y los fonemas del nivel T29 en el corpus CIEMPIESS. El corpus DIMEx100 es un punto de referencia adecuado para comparar con el corpus CIEMPIESS, ya que ambos son corpus del español del centro de México, y utilizan a Mexbet como alfabeto fonético.

De la tabla 6.2 se puede observar que los porcentajes de ocurrencias de fonemas son muy parecidos en ambos corpus.

6.1.1 Pruebas de reconocimiento en vivo con Sphinx4

En esta sección se presentan pruebas de reconocimiento de voz en vivo, utilizando el software de reconocimiento CMU-Sphinx4. Este sistema no cuenta con un módulo de entrenamiento propio, así que es necesario crear los modelos acústicos por medio del entrenador de Sphinx3.

El experimento consiste en reconocer los dígitos por medio de una gramática que sólo permite enunciar un dígito a la vez. Cada hablante repite los dígitos tres veces en total. Los hablantes reclutados para el experimento son todos del sexo masculino, en una edad entre 20 y 35 años, pero de diferentes zonas geográficas: dos de ellos son originarios del DF, uno del estado de Veracruz y el último de la República de Chile. El experimento se llevó a cabo en un ambiente de ruido moderado de oficina

[6.2] El reconocimiento de voz en vivo o en modo "live" requiere que una persona pronuncie palabras o frases frente a un micrófono, para que estas sean reconocidas de inmediato por el sistema ASR.

Tabla 6.1: Distribución de fonemas y alófonos del nivel T66 en el corpus CIEMPIESS.

No.	**Fonema**	**Ocurrencias Nivel T66 CIEMPIESS**	**Porcentaje Nivel T66 CIEMPIESS**	No.	**Fonema**	**Ocurrencias Nivel T66 CIEMPIESS**	**Porcentaje Nivel T66 CIEMPIESS**
1	p	20,023	2.81 %	34	r(_0	9,036	1.27 %
2	t	36,249	5.10 %	35	r(_\	16,254	2.28 %
3	k	19,805	2.78 %	36	r	3,605	0.50 %
4	k_j	10,348	1.45 %	37	j	15,204	2.13 %
5	b	6,691	0.94 %	38	i(	2,078	0.29 %
6	V	8,936	1.25 %	39	i	12,146	1.70 %
7	d	19,918	2.80 %	40	I	5,249	0.73 %
8	D	15,085	2.12 %	41	e	30,962	4.35 %
9	g	1,400	0.19 %	42	E	12,997	1.82 %
10	G	4,155	0.58 %	43	a_j	388	0.05 %
11	tS	1,591	0.22 %	44	a	40,485	5.69 %
12	f	4,685	0.65 %	45	a_2	1,455	0.20 %
13	s	60,823	8.55 %	46	o	25,331	3.56 %
14	z	750	0.10 %	47	O	17,298	2.43 %
15	s_[	8,083	1.13 %	48	u(	736	0.10 %
16	z_[	224	0.03 %	49	u	3,810	0.53 %
17	**S**	740	0.10 %	50	U	2,457	0.34 %
18	x	4,272	0.60 %	51	w	6,293	0.88 %
19	Z	1,270	0.17 %	52	j_7	0	0.00 %
20	dZ	1,858	0.26 %	53	i(_7	205	0.02 %
21	m	22,397	3.15 %	54	i_7	12,410	1.74 %
22	n	37,628	5.29 %	55	I_7	4,474	0.62 %
23	M	342	0.04 %	56	e_7	47,271	6.65 %
24	n_[	12,498	1.75 %	57	E_7	15,439	2.17 %
25	n_j	63	0.008 %	58	a_j_7	1,123	0.15 %
26	N	1,423	0.20 %	59	a_7	34,824	4.90 %
27	n∼	867	0.12 %	60	a_2_7	3,944	0.55 %
28	l	30,792	4.33 %	61	o_7	12,029	1.69 %
29	l_[	742	0.10 %	62	O_7	14,690	2.06 %
30	l_j	1	0.0001 %	63	u(_7	65	0.009 %
31	l_0	1,386	0.19 %	64	u_7	4,727	0.66 %
32	**tl**	1	0.0001 %	65	U_7	4,174	0.58 %
33	r(	13,844	1.94 %	66	w_7	609	0.08 %

Tabla 6.2: Distribución de fonemas en nivel T29 del CIEMPIESS comparada con la del nivel T22 en el DIMEx100.

No.	Fonema	Ocurrencias Nivel T22 DIMEx100	Porcentaje Nivel T22 DIMEx100	Ocurrencias Nivel T29 CIEMPIESS	Porcentaje Nivel T29 CIEMPIESS
1	p	6,365	2.58 %	19,628	2.80 %
2	t	11,592	4.69 %	35,646	5.10 %
3	k	9,317	3.77 %	29,649	4.24 %
4	b	5,206	2.11 %	15,361	2.19 %
5	d	13,744	5.56 %	34,443	4.92 %
6	g	2,140	0.87 %	5,496	0.78 %
7	tS	367	0.15 %	1,567	0.22 %
8	f	1,988	0.80 %	4,609	0.65 %
9	s	25,114	10.16 %	68,658	9.82 %
10	**S**	0	0.0 %	736	0.10 %
11	x	1,897	0.77 %	4,209	0.60 %
12	Z	777	0.31 %	3,081	0.44 %
13	m	7,104	2.87 %	21,601	3.09 %
14	n	17,323	7.01 %	51,493	7.36 %
15	n∼	331	0.13 %	855	0.12 %
16	r(	14,089	5.70 %	38,467	5.50 %
17	r	1,538	0.62 %	3,546	0.50 %
18	l	13,508	5.47 %	32,356	4.63 %
19	**tl**	0	0.0 %	1	0.00014 %
20	i	21,020	8.5 %	34,063	4.87 %
21	e	33,696	13.63 %	43,267	6.19 %
22	a	29,652	12.00 %	41,601	5.95 %
23	o	22,960	9.29 %	41,888	5.99 %
24	u	7,441	3.01 %	13,099	1.87 %
25	i_7	0	0.00 %	16,861	2.41 %
26	e_7	0	0.00 %	61,711	8.83 %
27	a_7	0	0.00 %	39,234	5.61 %
28	o_7	0	0.00 %	26,233	3.75 %
29	u_7	0	0.00 %	9,417	1.34 %

Tabla 6.3: Resultados de reconocimiento en vivo con el software Sphinx4.

Frases a reconocer	DF1	DF2	Chile	Ver.
Cero	ok	ok	ok*	ok
Uno	ok	ok*	ok*	ok
Dos	ok	ok	ok*	ok
Tres	ok	ok*	ok	ok*
Cuatro	ok	ok	ok*	ok*
Cinco	ok	ok*	ok*	ok*
Seis	ok	ok*	ok**	ok*
Siete	ok	ok*	ok*	ok
Ocho	ok*	ok**	ok*	ok*
Nueve	ok	ok	ok	ok*

(aproximadamente un SNR de 20 dB).

La tabla 6.3 muestra los resultados del experimento por cada hablante. Cada vez que un dígito era reconocido correctamente se indicaba con un "ok", y cada vez que hizo falta repetir el dígito una o más veces, se indicó por medio de un asterisco. El doble asterisco indica que en alguno de los tres intentos, el sistema fue incapaz de reconocer el dígito. En caso de que algún dígito no hubiera sido reconocido en ninguno de los 3 intentos, se indicaría con un "no".

Este pequeño experimento muestra la flexibilidad del corpus CIEMPIESS, para ser utilizado en diversos tipos de sistemas y con diversos tipos de hablantes. Se puede observar que un sistema entrenado con el CIEMPIESS, guarda cierta tolerancia para reconocer hablantes que no sean necesariamente del centro de México.

Es importante mencionar que el reconocimiento fue afectado por cambios en la intensidad de la voz. A los hablantes con voz menos alta, se aumentó el volumen de captación del micrófono; y para los hablantes con voz más potente, se redujo el volumen.

6.1.2 Pruebas de reconocimiento en vivo con Julius

En esta sección se muestra un experimento de reconocimiento en vivo utilizando el software Julius. Al igual que Sphinx4, Julius no cuenta con un módulo de entrenamiento propio, por lo cual debe utilizarse el sistema HTK.

En este experimento, tanto la construcción del modelo de lenguaje, como las pruebas de reconocimiento en vivo se realizaron a través del software HTK2SPHINX-CONVERTER, diseñado por el autor. Sus detalles pueden encontrarse en el apéndice D.

Tabla 6.4: Resultados de reconocimiento en vivo con el software Julius.

Frases a reconocer	DF1	DF2	Chile	Ver.
Marca cero	ok	ok	ok	ok
Marca uno	ok	ok	ok	ok
Marca dos	ok	ok*	ok	ok
Marca tres	ok	ok	ok	ok*
Marca cuatro	ok	ok*	ok	ok
Marca cinco	ok	ok*	ok*	ok*
Marca seis	ok	ok	ok*	no
Marca siete	ok	ok	ok	ok
Marca ocho	ok	ok*	ok	ok*
Marca nueve	ok	ok*	ok*	no
Llama a oficina	ok	ok*	ok	ok
Llama a casa	ok	ok*	ok**	ok*
Llama a trabajo	ok	ok*	no	no

El HTK2SPHINX-CONVERTER invoca a HTK y a Julius por medio de un entorno de control muy parecido al del sistema Sphinx3. Este software se distribuye con unos cuantos audios extraídos del corpus CIEMPIESS (407 en total) para poder hacer rondas de entrenamiento y reconocimiento, que ayudan a verificar que el sistema se instaló adecuadamente. En este experimento se utilizan para entrenar solamente los 407 audios con los que se distribuye el HTK2SPHINX-CONVERTER.

La tabla 6.4 muestra los resultados del experimento, en donde se aprecia que se reclutó a los mismos 4 hablantes, en las mismas condiciones de ruido y además, con la misma nomenclatura para mostrar los resultados.

En este experimento se aprecian resultados un poco peores que los de la sección anterior; sin embargo, no dejan de ser satisfactorios puesto que aquí solamente se utilizó una pequeña parte del CIEMPIESS para entrenar.

6.1.3 Pruebas de reconocimiento de voz continua en modo "batch"

En esta sección se presentan tres experimentos de reconocimiento en modo "batch", utilizando tres sistemas ASR diferentes: El CMU-Sphinx3, HTK[6.3] y Kaldi[6.4]. Este tipo de experimentos constituyen las pruebas clásicas en reconocimiento de voz; con respecto al español del centro de México, las pruebas para los dos últimos sistemas son novedosas, lo que constituye una aportación trascendente para la investigación en este campo en Latinoamérica.

Las pruebas hechas en los tres sistemas guardan en común lo siguiente:

- Se utiliza el alfabeto Mexbet en su nivel T29.
- Se utiliza el corpus CIEMPIESS en su totalidad, es decir, que el conjunto de entrenamiento consta de 16,017 audios y el conjunto de pruebas de 1,000 audios.
- Se utiliza el mismo modelo de lenguaje en formato ARPA, basado en 3-gramas
- Se utilizan los mismos diccionarios de pronunciación.
- Los sistemas están basados en el paradigma de modelos ocultos de Markov.
- La parametrización de los audios está basada en vectores de atributos MFCC.

Al final, cada sistema arroja los resultados que se muestran en la tabla 6.5, en donde se observa que son muy similares; en especial los sistemas Sphinx3 y HTK, lo cual resulta lógico ya que fueron diseñados utilizando un clasificador HMM muy similar. El sistema Kaldi, que obtiene mejores resultados, es un sistema aproximadamente 15 años más reciente que los otros dos y se atribuye su mejor desempeño a un mejor manejo del modelo lenguaje. De los resultados se puede concluir que el CIEMPIESS guarda la consistencia de que: a condiciones de entrenamiento similares, se obtienen resultados similares.

6.2 Evaluación de las herramientas de transcripción automática

Se diseña la librería "fonetica2" como una herramienta de software libre que contiene funciones para transcribir palabras en español a nivel fonético y fonológico; estas funciones son necesarias para pruebas de reconocimiento y son originales. La librería fonetica2 está programada en Python 2.7 e incluye un total de seis funciones que pueden

[6.3] El sistema en HTK es un software diseñado por el autor llamado HTK-BENCHMARK, que fue creado para ser compatible con los archivos de entrada del CIEMPIESS y hacer reconocimiento en modo "batch" con ayuda de un modelo de lenguaje basado en 3-gramas (ver apéndice E).

[6.4] Las pruebas en Kaldi fueron llevadas a cabo por el Dr. Iván Vladimir Meza Ruiz del IIMAS.

Tabla 6.5: Pruebas de reconocimiento en modo "batch" con diferentes sistemas.

System	WER
Sphinx3	44.0 %
HTK	42.45 %
Kaldi	33.15 %

ser incorporadas al código del usuario. Cada una de estas funciones acepta palabras en español en minúsculas como argumentos y pueden o no tener la vocal tónica indicada con una mayúscula. Se presentan los experimentos para tres funciones:

- vocal_tonica(): Devuelve la misma palabra de entrada, pero con su vocal tónica en mayúscula (ej. cAsa, pErro, gAto, etc.).
- T29(): Transcribe fonológicamente la palabra de entrada en alfabeto Mexbet T29.
- T66(): Transcribe fonéticamente la palabra de entrada en alfabeto Mexbet T66.

En la página del proyecto CIEMPIESS-UNAM se pueden hallar cuatro herramientas en línea para probar las funciones vocal_tonica(), div_sil(), T29() y T66(). El lector puede encontrar las 6 funciones en el apéndice G.

6.2.1 Evaluación de la función vocal_tonica()

Para la evaluación de la función vocal_tonica() se utilizaron palabras extraídas del corpus CIEMPIESS (Hernández-Mena & Herrera-Camacho, 2014). Este corpus cuenta con 12,105 palabras diferentes. Se tomaron 1,539 palabras de forma aleatoria, lo cual representa el 12.82 % de todas las palabras del CIEMPIESS. Luego, se eliminaron las palabras extranjeras (87) y es así como se obtuvieron 1,452 palabras para nuestro análisis. Manualmente se revisó que estuvieran correctamente acentuadas[6.5].

El resultado de este experimento es que un 90.35 % (1,312 palabras) fueron correctamente acentuadas, contra 140 mal acentuadas. Una de las razones por las que ocurrieron estos errores es por causa de nombres propios o verbos conjugados. La tabla 6.6 resume estos resultados. Se considera que el algoritmo proporciona una buena precisión.

[6.5] Este análisis lo hizo la estudiante de letras hispánicas: Nancy Norely Martínez

Tabla 6.6: Evaluación de la función vocal_tonica()

Palabras tomadas del corpus CIEMPIESS	1539
Palabras extranjeras omitidas	87
Palabras analizadas	1452
Palabras con la vocal tónica mal colocada	140
Palabras con la vocal tónica bien colocada	1312
Porcentaje de palabras con la vocal tónica bien colocada	**90.35 %**

Tabla 6.7: Comparación entre TRANSCRÍBEMEX y la Función T29().

Palabras en el DIMEx100	11575
Palabras analizadas	8738
Transcripciones no idénticas	67
Transcripciones idénticas	8670
Porcentaje de Transcripciones idénticas	**99.2 %**

6.2.2 Evaluación de la función T29()

Para la evaluación de la función T29() se cuenta con un elemento de comparación. Se trata de un software llamado TRANSCRÍBEMEX (no del autor), que es mencionado con ese nombre en (Cuetara-Priede, 2004).

El TRANSCRÍBEMEX es una herramienta de software con interfaz gráfica programada en Perl, creada por algunos de los que colaboraron en el corpus DIMEx100, capaz de producir transcripciones fonológicas en Mexbet T22 y transcripciones fonéticas en Mexbet T54.

Ahora bien, el experimento consiste en comparar las transcripciones hechas por el TRANSCRÍBEMEX, contra las transcripciones realizadas por la función T29(). Las palabras utilizadas para la evaluación vienen del corpus DIMEx100.

Para el experimento se utilizan sólo las transcripciones en T22 hechas por la herramienta TRANSCRÍBEMEX, para lo cual se adecuó el Mexbet T29 al Mexbet T22. El resultado es que ambas herramientas son similares en un 99.2 %. La tabla 6.7 resume el experimento y los resultados.

Este resultado confirma la precisión de la función T29(), y demuestra su confiabilidad.

6.2.3 Evaluación de la función T66()

Para la evaluación de la función T66() no se pudo contar con un algún elemento de comparación como ocurrió con la función T29(). Esto debido a que el nivel más cercano al T66, es el T54 en el corpus DIMEx100, siendo que estos niveles no son tan cercanos, como lo son el nivel T22 y el nivel T29. Por esta razón, se decidió validar la función T66() con la ayuda de una experta humana[6.6].

La corroboración manual de las transcripciones generadas por la función T66() se hizo por medio de 305 palabras. Estas palabras fueron cuidadosamente escogidas para abarcar todas las reglas de transcripción fonética y fonológica del español del centro de México.

La transcripción de esas 305 palabras ocurrió sin errores. El hecho de que la función T66() funcione perfectamente para este pequeño vocabulario es un indicador de su pertinencia.

6.3 Evaluación del modelo de lenguaje

En esta sección se presentan experimentos con el fin de evaluar algunas características del modelo de lenguaje incluido en la versión publicada del corpus CIEMPIESS.

En términos generales, entre mayor sea la variedad de voces en un corpus de entrada, mayor será la calidad del modelo acústico. Sin embargo, en el modelo de lenguaje es deseable que exista una redundancia de n-gramas.

Ahora bien, cuando se habla de grandes cantidades de discursos, de los más variados temas y autores, es un hecho que ciertas reglas son siempre respetadas, y al mismo tiempo, ciertos patrones pueden emerger. Por ejemplo, la gente por lo regular al hablar, conjuga bien la mayoría de los verbos que dice, o respeta muchas de las construcciones gramaticales convencionales como la de "artículo + sujeto + verbo + predicado", etcétera.

Esto último resulta muy valioso para el corpus CIEMPIESS, ya que con este se pretende entrenar sistemas de reconocimiento con grandes vocabularios, independientes del hablante y para un discurso genérico. Si bien, el ASR perfecto es inasequible con la tecnología actual, se le puede acercar con modelos de lenguaje mejor diseñados y diccionarios de pronunciación cada vez más ricos.

[6.6] Se trata de la licenciada en letras hispánicas, Alejandra Chavarría.

Tabla 6.8: Características del texto crudo extraído de Boletines-UNAM

Descripción	Valor
Cantidad de boletines	2489
Boletín más viejo	12:30 hrs. 01 de enero de 2010
Boletín más reciente	20:00 hrs. 18 de febrero de 2013
Líneas de texto totales	197,786
Total de palabras (contando repeticiones)	1,642,782
Total de palabras únicas	113,313
Promedio de palabras por boletín	660
Palabras en el boletín más largo	2710
Palabras en el boletín más corto	21

6.3.1 Características del texto utilizado en el modelo de lenguaje

Para fines explicativos, el texto "puro" extraído de los boletines será denominado como "texto crudo", y al texto que fue depurado para crear, tanto el modelo de lenguaje, como los diccionarios de pronunciación para todos los experimentos, será conocido como "texto procesado".

De esta manera, la tabla 6.8 muestra las características del texto crudo, el cual puede contener signos de puntuación, fragmentos de código HTML, palabras extranjeras, palabras inexistentes, errores ortográficos, dígitos, etcétera.

Ahora bien, el texto crudo no es adecuado para ser utilizado con las herramientas de modelado de lenguaje. Por lo tanto, es conveniente someterlo a un procesamiento que consiste en lo siguiente:

- Se eliminan los signos de puntuación y se utilizan los puntos para crear una segmentación del texto, tal que se lea una oración por cada línea del archivo.
- Se eliminan los espacios en blanco extras, es decir, que cada palabra en el texto debe ser separada por uno, y sólo un espacio en blanco.
- Se elimina cualquier línea del texto que no sea correctamente fonétizada por medio de la función T29(). Esto es esencial, para un sistema ASR, ya que todas las palabras del modelo de lenguaje deben estar también en el diccionario de pronunciación, de lo contrario, el sistema marca errores.
- Se eliminan las líneas del archivo que sólo contengan una sola palabra. Esto se debe a que el modelo de lenguaje está basado en n-gramas, en donde se toma en

Tabla 6.9: Características del texto procesado, utilizado para el modelo de lenguaje.

Descripción	Valor
Total de palabras (contando repetidas)	1,505,491
Total de palabras únicas	49085
Líneas de texto totales	279,507
Promedio de palabras por línea	5.38
Palabras en la línea más larga	43
Palabras en la línea más corta	2

cuenta la palabra actual, la palabra que sigue, y también la palabra anterior, por lo tanto, no se puede hacer un modelo de n-gramas con oraciones que contengan una sola palabra.

- A cada palabra del texto que queda, se le indica la vocal tónica en mayúscula, por medio de la función vocal_tonica().

Todo este proceso, eleva la calidad del modelo de lenguaje.

Es importante añadir que el texto de las 16,017 transcripciones del conjunto de entrenamiento del CIEMPIESS fue añadido al texto procesado, cuyas características principales se muestran en la tabla 6.9.

6.3.2 Mediciones al modelo de lenguaje

Desde un punto de vista estricto, el texto procesado con el que se creó el modelo de lenguaje del CIEMPIESS es un corpus textual, y para poder medir algunas de sus características es necesario contar con otro corpus más grande para comparar. Para este fin se utilizó el corpus CREA.

Al comparar la distribución de frecuencia entre el texto procesado para el modelo de lenguaje del CIEMPIESS y el corpus CREA, se observa que las 20 palabras más frecuentes entre ambos corpus son casi exactamente las mismas y que están casi en la misma proporción, tal y como se ilustra en la tabla 6.10. Este resultado indica que el modelo de lenguaje del corpus CIEMPIESS es un representativo válido del idioma español. Para confirmarlo, se muestra el siguiente experimento.

Una medida más precisa de qué tan parecidas son las distribuciones de frecuencia entre ambos corpus, es el error cuadrático medio (ECM), cuya expresión matemática se aprecia en la ecuación 6.1:

Tabla 6.10: Distribución de frecuencias del corpus CREA comparada con la del texto procesado (t.p.).

No.	Palabras del CREA	Frec. Norm. CREA	Frec. Norm. t.p.	No.	Palabras del CREA	Frec. Norm. CREA	Frec. Norm. t.p.
1	de	0.065	0.076	11	las	0.011	0.011
2	la	0.041	0.045	12	un	0.010	0.009
3	que	0.030	0.028	13	por	0.010	0.010
4	el	0.029	0.029	14	con	0.009	0.010
5	en	0.027	0.034	15	no	0.009	0.006
6	y	0.027	0.032	16	una	0.008	0.008
7	a	0.021	0.017	17	su	0.007	0.005
8	los	0.017	0.016	18	para	0.006	0.010
9	se	0.013	0.015	19	es	0.006	0.008
10	del	0.012	0.013	20	al	0.006	0.005

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{6.1}$$

En donde $\hat{Y}_i$ representa la frecuencia normalizada de una palabra dada dentro del corpus más pequeño de los dos que se comparan, que en este caso sería el texto procesado, y Y_i representa la frecuencia normalizada de una palabra dada dentro del corpus CREA.

El resultado de esta comparación arroja el resultado $EMC = 7.5 \times 10^{-9}$. Para poder comprenderlo mejor es preciso normalizarlo de esta manera.

Resulta que si la distribución de frecuencias de ambos corpus fuera la misma, la diferencia $\hat{Y}_i - Y_i$ siempre daría cero, siendo este es el mejor de los casos. De este modo, el peor de los casos ocurriría si por ejemplo, ninguna de las palabras del texto procesado fuera parte del corpus CREA, por lo tanto, la misma diferencia $\hat{Y}_i - Y_i$ sería siempre igual a $\hat{Y}_i$. Si esto ocurre, la ecuación 6.1 cambiaría por la ecuación 6.2, que se muestra a continuación:

$$ECM_{peor\_caso} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i)^2 \tag{6.2}$$

Considerando que un EMC del peor caso es un 100 %, que se tendría cuando no

son nada parecidos, se obtiene el resultado de $ECM_{Normalizado} = 2.65\,\%$, lo cual puede interpretarse como que las distribuciones entre ambos corpus son muy parecidas, y eso significa que el texto procesado es una buena muestra del lenguaje que se quiere modelar.

Finalmente, otra medición que apoya la hipótesis de que el texto procesado representa al mismo sistema que el corpus CREA, es decir, el idioma español, es la correlación medida mediante la función de Python llamada pearsonr(), la cual arrojó un valor de 0.98, lo que indica una alta correlación, lo cual puede interpretarse como que se trata efectivamente del mismo sistema.

6.3.3 Perplejidad del modelo de lenguaje

La perplejidad es una medida típica utilizada para medir lo predictivo que puede llegar a ser un modelo de lenguaje. Los elementos necesarios para medirla son obviamente, el modelo de lenguaje que se quiere evaluar, y un texto de prueba. De este modo, la perplejidad es un número que indicará qué tanto ese modelo de lenguaje puede "predecir" al texto de prueba, y cuanto más bajo sea ese número, mejor será la predicción.

Ahora bien, en esta sección se muestran mediciones de perplejidad con diferentes textos de prueba, efectuadas por medio de la herramienta "statistical language modelling toolkit" (ver apéndice C); específicamente con uno de sus comandos llamado "evallm".

El primero de estos textos, se trata del archivo de transcripciones del conjunto de pruebas del CIEMPIESS, que se compone de mil oraciones. La medición de la perplejidad arroja un valor de 506.48.

A modo de comparación se usó el archivo de transcripciones del corpus DIMEx100, en particular un 54.5 %[6.7] de todas las transcripciones, lo que arroja un valor de perplejidad de 541.95.

La interpretación es que los dos valores de perplejidad presentados aquí son bastante similares; debido a que fueron obtenidos con textos no contenidos en el modelo de lenguaje a evaluar, representan un nivel de predicción aceptable.

Para dar una idea de cota inferior, se muestra el valor de perplejidad utilizando como texto de prueba a las transcripciones de entrenamiento del CIEMPIESS, que como ya se dijo, están incluidas en el modelo de lenguaje, y por lo tanto, deben presentar una perplejidad baja. Esta hipótesis se confirma al arrojar un valor de 22.64.

[6.7] Esto es así porque deben descartarse las transcripciones del DIMEx100 que contengan palabras que no estén en el modelo de lenguaje del CIEMPIESS, para que la comparación sea más justa.

6.3.4 Influencia del modelo de lenguaje

En esta sección se pretende ilustrar la enorme importancia que el modelo de lenguaje tiene para un sistema ASR, y cómo es que este podría convertirse en un agente dañino para el sistema, si no se diseña adecuadamente.

Los experimentos de las secciones 6.1.1 y 6.1.2 muestran como sistemas ASR entrenados con el CIEMPIESS obtienen resultados favorables cuando se usan gramáticas simples.

Ahora bien, para este experimento se graban 30 oraciones basadas en la misma gramática de la sección 6.1.2, se entrena al sistema Sphinx3 con una pequeña parte del corpus CIEMPIESS (sólo 407 audios) y se le configura con el modelo de lenguaje del CIEMPIESS basado en 3-gramas.

El resultado de este experimento es un WER del 100 %. La causa de esto es que el modelo de lenguaje de 3-gramas utilizado, no alcanza a predecir el estilo tan "rígido" de la gramática con la que se formaron las oraciones de prueba.

Analizando los resultados, puede verse que en ciertas oraciones, el sistema parece reconocer bastante bien, es decir, equivocándose en pocas palabras, como por ejemplo en las frases:

```
Original: <s> llama a escuela </s> (CAR_TEST-0024M_CAR_06ABR14)
Reconocida: <s> llamada escuela </s> (CAR_TEST-0024M_CAR_06ABR14)

Original: <s> marca siete uno seis tres cinco cinco cuatro
seeis </s> (CAR_TEST-0013M_CAR_06ABR14)

Reconocida: <s> marca siete uno a seis pues cinco cinco cuatro
a seis </s> (CAR_TEST-0013M_CAR_06ABR14)
```

En el primer ejemplo se puede observar un error por sinalefa. En el segundo se puede observar que se confunde el dígito "tres" por la palabra "pues", lo cual puede dar lugar a una especie de reacción en cadena, o efecto dominó que atrofie totalmente el reconocimiento de una frase.

Un ejemplo de efecto dominó es el siguiente:

```
Original: <s> marca dos siete uno nueve cuatro cinco cuatro
siete </s> (CAR_TEST-0022M_CAR_06ABR14)

Reconocida: <s> que marca ocho seis y siete sienten ser tres
nueve los seis </s> (CAR_TEST-0022M_CAR_06ABR14)
```

Pudiendo llegar a casos tan extremos como el siguiente:

```
Original: <s> marca ocho dos cuatro ocho seis seis cero
cero </s> (CAR_TEST-0025M_CAR_06ABR14)

Reconocida: <s> llamada escuela o petroleo no es una ligera
parte de la mezcla mexicana estudio de la
fuente <s> (CAR_TEST-0025M_CAR_06ABR14)
```

En donde no sólo se pierde absolutamente todo el sentido de la frase reconocida, sino que además, se añaden palabras aparentemente absurdas.

La razón de que esto ocurra es que las oraciones provenientes de una gramática requieren cierto tipo de 3-gramas que en este modelo de lenguaje son difíciles, sino es que imposibles de encontrar. Lo cual provoca que se elijan otros 3-gramas con mayor probabilidad, aunque estos no tengan nada que ver con lo que dice la frase a reconocer.

6.4 Evaluación de conceptos sobre PFS

Las secciones 6.4.1 y 6.4.2 implementan el algoritmo PFS presentado en la sección 5.1.9, haciendo detección y conteo de PFS en diferentes grupos de textos.

Tabla 6.11: Conteo de PFS considerando a las vocales iguales

Objeto de medición	M.L.	Train	Test
Palabras analizadas	49,085	12,100	1,176
Grupos PFS con 2 o más elementos	7,691	1,578	54
Número total de PFS	18,270	3,613	116
Porcentaje de PFS	37.22 %	29.85 %	9.86 %

6.4.1 Conteo de PFS considerándolas equiprobables

La tabla 6.11 muestra los resultados de detectar PFS en diversos conjuntos de texto, que tienen que ver directamente con un sistema ASR entrenado con el corpus CIEMPIESS. Las categorías fonéticas utilizados para este experimento son los siguientes:

```
1 p b
2 t d
3 k g
4 tS S
5 m n
6 Z n~
7 r( r l
8 f s x
9 a e i o u
```

Ahora bien, el primero de los grupos de texto analizados que aparece en la tabla 6.11, se trata del modelo de lenguaje que viene por defecto en el CIEMPIESS. Al ser este el conjunto de texto más grande, se encontraron la mayor cantidad de PFS, con respecto a los demás conjuntos de texto. De hecho, se encontró que un 37.22 % de todas las palabras analizadas pertenecían a un grupo PFS mayor o igual a dos elementos.

Sin embargo, aunque esta proporción de PFS es bastante importante, hay que recordar que aquí, las condiciones son las más ideales, es decir, que en una situación real, no habrá tantas palabras que sean equiprobables, sin mencionar que muchos de los grupos PFS encontrados son del tipo: “preventiva-preventivo” , “provechoso-provechosa”, “preparado-preparada”, es decir, que son palabras que sólo son diferentes por su género masculino o femenino.

Tabla 6.12: Conteo de PFS considerando a las vocales distintas

Objeto de medición	M.L.	Train	Test
Palabras analizadas	49,085	12,100	1,176
Grupos PFS con 2 o más elementos	1,400	370	11
Número total de PFS	3,055	784	22
Porcentaje de PFS	6.22 %	6.47 %	1.87 %

Lo que ocurriría en un sistema real con estas palabras, es que fácilmente pueden ser separadas por medio del modelo de lenguaje, si tan solo se toma en cuenta su probabilidad, ya que estarían en trigramas bien diferenciados por artículos como "el" o "la", como por ejemplo en: "**la** luz preventiva", "**el** mantenimiento preventivo" o "**el** desayuno preparado", "**la** comida preparada", etc.

Las vocales tónicas son las únicas vocales que hasta ahora son diferenciadas, para lograr agrupar sólo PFS con la vocal tónica en la misma posición, como en las palabras: "prevención-propensión", "propondrá-provendrá", "propagadas-provocadas", etc. Esto intuitivamente parece volver a las palabras más parecidas aún.

La tabla 6.12 muestra los resultados de la detección de PFS equiprobables, pero ahora con categorías fonéticas que consideran a todas las vocales diferentes.

Estos resultados prueban que al volver más restrictivos a las categorías fonéticas, las PFS se reducen drásticamente en el sistema.

6.4.2 Conteo de PFS considerando su probabilidad

En esta sección se implementó la técnica de "cortes al modelo de lenguaje" presentada en la sección 5.2.3, para detectar PFS con base en su probabilidad. Se hacen cortes equidistantes al modelo de lenguaje del sistema para analizar las posibles PFS que se podrían encontrar, y sin tener preferencia por ninguna en particular.

El número de ocurrencias está directamente relacionado con la probabilidad de las palabras en un texto, así que se utilizó esta medida para agrupar a las PFS, es decir, que se fijó un $\epsilon = 10$ como la distancia máxima posible entre dos PFS.

Otra razón por la que se decidió utilizar la frecuencia de la palabra como medida de su probabilidad, fue la posibilidad de aplicar la idea de los "cortes al modelo de lenguaje", no sólo al modelo de lenguaje, sino también a los archivos de transcripción tanto del conjunto de pruebas, como del de entrenamiento. Esto se especifica debido a que el único conjunto textual que cuenta con una medida de probabilidad de sus palabras es el modelo de lenguaje y no los archivos de transcripción, por tanto, la frecuencia de palabra resulta una excelente medida para unificar criterios para todos

Tabla 6.13: Conteo de PFS considerando su probabilidad y a sus vocales iguales

Objeto de medición	M.L.	Train	Test
Palabras analizadas	49,085	12,100	1,176
Grupos PFS con 2 o más elementos	5,857	1,309	48
Número total de PFS	13,247	2,905	104
Porcentaje de PFS	26.98 %	24.00 %	8.84 %

Tabla 6.14: Conteo de PFS considerando su probabilidad y a sus vocales distintas

Objeto de medición	M.L.	Train	Test
Palabras analizadas	49,085	12,100	1,176
Grupos PFS con 2 o más elementos	886	274	7
Número total de PFS	1,865	564	14
Porcentaje de PFS	3.79 %	4.66 %	1.19 %

los conjuntos de texto analizados.

Ahora bien, la tabla 6.13 muestra un conteo de PFS en los mismos tres conjuntos de texto de la sección anterior, y también considerando las vocales no tónicas como iguales, pero tomando en cuenta la probabilidad de cada PFS. Esta tabla arroja números muy parecidos a los registrados en la tabla 6.11, quizá debido a que, de nuevo, la lista de categorías fonéticas es muy poco restrictiva al considerar a las vocales no tónicas, como iguales.

Los resultados más restrictivos en cuanto a conteo de PFS se refiere, son los mostrados en la tabla 6.14, que muestra como el porcentaje de PFS se reduce a menos del 10 % en todos los conjuntos de texto analizados. El resultado mostrado en esta y en la sección 6.4.1, es que las PFS están ocurriendo en una proporción menor al 10 %.

Sin embargo, si se quiere estudiar el fenómeno del desplazamiento acentual, y se crean categorías fonéticas poco restrictivas en los que se consideren todas las vocales como iguales, o dicho de otro modo, que todas las vocales sean eliminadas del análisis para permitir agrupar palabras con desplazamiento acentual, se obtiene una densidad de PFS del 41.99 % en el modelo de lenguaje, considerando su probabilidad.

La idea que llevó de considerar a las palabras con desplazamiento acentual como PFS, y que luego llevó a la idea de marcarlas en las transcripciones, para finalmente medir su efecto en un sistema ASR, es una de las contribuciones más importantes de esta tesis.

6.5 Evaluación de la influencia de vocales tónicas en reconocimiento de voz.

En esta sección se presenta uno de los experimentos más importantes de esta tesis, puesto que refleja de manera práctica, la teoría desarrollada en el capítulo 5, que considera a las palabras cuyas transcripciones fonéticas sólo se diferencian entre sí por la posición de su vocal tónica, como palabras fonéticamente similares.

Hay que recordar que experimentalmente en la sección 6.4.1 se determina básicamente que las PFS son, de hecho, escasas en condiciones normales, pero que de manera particular, las PFS con diferente posición en la vocal tónica son algo más numerosas que otro tipo de PFS.

De esta manera, y partiendo del hecho que la solución dada en el capítulo 5 al problema de las PFS, en general es construir un mejor modelo de lenguaje que separe por probabilidad a PFS con significados muy distintos, se presenta aquí un experimento en donde se construyen 2 diferentes modelos de lenguaje, uno de ellos considerando vocales tónicas, y el otro no. De este modo se sabría si hay evidencia de que la solución al problema de las PFS, es por medio del modelo de lenguaje.

Adicionalmente, se crean también 4 diferentes diccionarios de pronunciación para determinar qué nivel de granularidad en la transcripción fonética es el óptimo para utilizarse con el corpus CIEMPIESS. Dos de los diccionarios están en el nivel T29, y los otros dos en el nivel T66, ambos de Mexbet. En ambos niveles un diccionario considera vocales tónicas y el otro no.

Ahora bien, el experimento consiste en trazar 4 diferentes curvas de aprendizaje, dosificando poco a poco la cantidad de audio con el que se entrena el sistema. El experimento es llevado a cabo por medio del sistema CMU-Sphinx3 y debe realizarse trazando un punto en la gráfica que represente el WER, con cada hora de audio añadida. Es decir, se entrena con una hora de audio y se hace una medida del WER, luego se entrena con dos horas y se hace lo mismo, y así sucesivamente hasta agotar todo el audio de entrenamiento del CIEMPIESS.

En total, este experimento debe realizarse 4 veces, entrenando un sistema en T29 con tónica, luego uno en T29 sin tónicas, etcétera. Al final, se interponen las gráficas y se ve cuál presenta el WER más bajo. El audio que es reconocido en estas pruebas es siempre el mismo para los 4 experimentos, y se trata del conjunto de pruebas del CIEMPIESS, cuyos audios y transcripciones no están incluidos en modo alguno ni en el conjunto de entrenamiento, ni el modelo de lenguaje.

En el caso particular del CIEMPIESS, resulta que más o menos cada mil grabaciones representan el equivalente a una hora de audio. De acuerdo a esto, se estimó el número de senones de acuerdo a la tabla 6.15 que se obtuvo de la sección de “preguntas

Tabla 6.15: Número de senones/número de grabaciones.

No. de Grabaciones	No. de Senones
1000-3000	500
4000-6000	1000
6000-8000	2500
8000-10000	4000
10000-17000	5000

frecuentes" de la página de Sphinx[6.8].

La figura 6.1 muestra cuatro curvas de aprendizaje diferentes representando las siguientes condiciones de reconocimiento:

1. T29 TONICS: Tanto en modelo de lenguaje como el diccionario de pronunciación consideran la vocal tónica de cada palabra, y el nivel de granularidad de las transcripciones es Mexbet T29.
2. T29 NO TONICS: Ni el modelo de lenguaje ni el diccionario de pronunciación consideran la vocal tónica de ninguna palabra, y el nivel de granularidad de las transcripciones es Mexbet T29.
3. T66 TONICS: Tanto en modelo de lenguaje como el diccionario de pronunciación consideran la vocal tónica de cada palabra, y el nivel de granularidad de las transcripciones es Mexbet T66.
4. T66 NO TONICS: Ni el modelo de lenguaje ni el diccionario de pronunciación consideran la vocal tónica de ninguna palabra, y el nivel de granularidad de las transcripciones es Mexbet T66.

Como puede observarse en la figura 6.1, las mejores condiciones de entrenamiento para hacer reconocimiento son las llamadas "T29 TONICS". Esto comprueba la hipótesis del mejor desempeño de un ASR marcando las vocales tónicas. Sin embargo, la diferencia entre estas y las "T29 NO TONICS" es muy pequeña (sólo 1.7 %) y quizá no sea estadísticamente significativo. La "T29 TONICS" también resultó ser la curva más estable de todas. Al final, el WER de cada condición fue:

- T29 TONICS = 44.0 % (WER)

[6.8] Ver: http://www.speech.cs.cmu.edu/sphinxman/FAQ.html#2

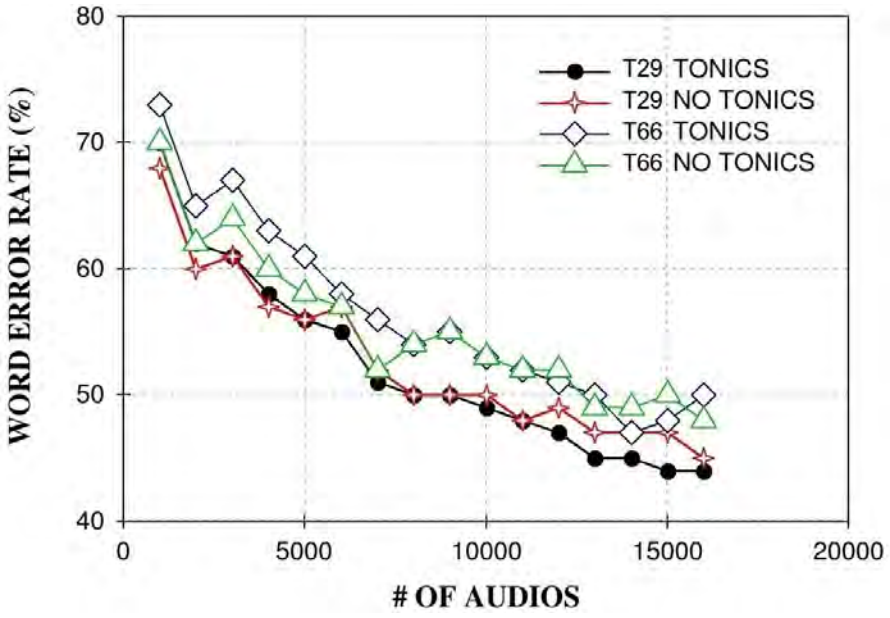

Figura 6.1: Curvas de aprendizaje para diferentes condiciones de entrenamiento.

- T29 NO TONICS = 45.7 % (WER)
- T66 TONICS = 50.5 % (WER)
- T66 NO TONICS = 48.0 % (WER)

También se observa que todas las curvas de aprendizaje son consistentes con la idea de que "entre más se entrena, más precisión en el reconocimiento se obtiene" y también se aprecia que las curvas no son asintóticas. Esto significa que aún se necesita agregar más horas de entrenamiento para tener un mejor desempeño, sobre todo en el nivel T66.

Finalmente, todos estos resultados encierran la conclusión de que el nivel T29 es el mejor de los dos niveles posibles, en este experimento. Como ya se mencionó, el nivel T66 carecía de representatividad de algunos fonemas en el corpus CIEMPIESS.

El nivel T29, sin considerar vocales tónicas, resulta ser el caso en el que menos fonemas deben ser aprendidos, y sin embargo, este caso no fue el que arrojó el WER más bajo. Por tanto, la interpretación es que el modelo de lenguaje es el que hace la diferencia entre el nivel T29 con tónicas y el T29 sin tónicas. De alguna manera, el sistema es más preciso mapeando fonemas a palabras, cuando estas contienen la información de la vocal tónica.

Capítulo 7

Conclusiones

En este trabajo se entrega una definición de palabras fonéticamente similares dentro del contexto del reconocimiento automático de voz, se presenta un algoritmo para detectarlas y se muestra un conjunto de métodos para minimizar su efecto en sistemas ASR.

Se entrega también un conjunto de aportaciones prácticas, como la construcción del corpus CIEMPIESS, que ya ha sido publicado en el Linguistic Data Consortium (LDC) (Hernández-Mena & Herrera-Camacho, 2015b). Toda la documentación sobre este recurso puede encontrarse en el apéndice B.

Como precedente al corpus CIEMPIESS, se entrega también el corpus CHM150, que demostró ser demasiado pequeño para las pruebas requeridas en esta tesis, pero que puede ser útil para otras aplicaciones del área de voz, como el reconocimiento de hablante. Al igual que el corpus CIEMPIESS, el corpus CHM150 ha sido publicado en LDC (Hernández-Mena & Herrera-Camacho, 2016), de donde puede ser descargado gratuitamente.

En este trabajo también se retoma el alfabeto fonético Mexbet, que fue creado especialmente para el español hablado en México, y se amplía a dos nuevos niveles de granularidad, independientes de los tres niveles: T22, T44 y T54, que ya habían sido definidos con anterioridad. Se trata de los niveles T29 y T66. Las tablas que muestran los símbolos de los dos nuevos niveles de Mexbet, junto con información sobre su punto y su modo de articulación se encuentran en el apéndice H, mientras que la tabla de equivalencias entre el alfabeto Mexbet y el alfabeto fonético internacional (AFI) se hallan en el apéndice I.

Con el alfabeto Mexbet, fue posible diseñar también diversos sistemas de fonetización de palabras en español como las herramientas en línea que se accesan desde el sitio `http://www.ciempiess.org/tools`, y que se usan para transcribir palabras en español, directamente en nivel T29 o T66 de Mexbet.

Además de esto, bajo el mismo principio con el que fueron creadas las herramientas

en línea, se creó una librería en lenguaje Python 2.7 para permitir que los usuarios puedan incorporar funciones de transcripción de Mexbet a su código propietario. Se trata de la **"librería fonetica2"** cuyo manual de uso se encuentra en el apéndice G.

El corpus CIEMPIESS fue probado en cuatro diferentes sistemas estándar de reconocimiento de voz (Sphinx, HTK, Julius y Kaldi), demostrando así su versatilidad y efectividad. Al mismo tiempo, se demuestra el correcto funcionamiento de estos cuatro sistemas cuando son alimentados con audios en español hablando en México. Los experimentos con HTK y Kaldi para el español hablado en México son inéditos.

Por lo tanto, otra contribución muy importante de esta tesis es la adaptación de los cuatro sistemas de reconocimiento mencionados al español hablado en el centro de México, utilizando un corpus de tamaño mayor a los que existen hasta el momento, en este dialecto particular. Con estos experimentos, las investigaciones nacionales posteriores, tendrán una base sólida para realizar nuevos diseños en reconocedores.

Sin embargo, uno de los problemas del corpus CIEMPIESS es la representatividad de fonemas en el nivel T66 de Mexbet. De manera empírica, se sabe que entre más símbolos existan dentro del alfabeto fonético del sistema ASR, mayor cantidad de audio se necesita para que ese alfabeto sea efectivo. Por lo tanto, si se desea crear buenos modelos acústicos en el nivel T66 de Mexbet, es necesario recopilar más audio. Se asume como hipótesis, que el nivel T66 debería tener mejor desempeño que el nivel T29, con un corpus mayor.

En el caso de las herramientas de transcripción fonética, el hecho de que en estas fuera considerado el uso de la vocal tónica que fue marcada en las transcripciones del CIEMPIESS y que fue requerida para los algoritmos de fonetización, división silábica y detección de PFS, permitió dar cuenta de la precisión y efectividad de estas herramientas, en diversos aspectos del reconocimiento de voz.

Tambien, derivado de los trabajos de esta tesis, las reglas de Mexbet están ahora condensadas en español en el apéndice F y divulgadas en inglés en el artículo (Hernández-Mena et al., 2014).

Otro de los puntos importantes en esta investigación fue la experimentación con sistemas ASR de código abierto. Al inicio se adoptó como sistema base al CMU-Sphinx3. Por tanto, se deja constancia de la manera de incorporar el corpus CIEMPIESS al CMU-Sphinx3, por medio de lo escrito en el apéndice A. Ahora bien, con respecto al sistema Sphinx, también se deja documentada la manera en que pueden hacerse modelos de lenguaje compatibles con este, mediante el software llamado "statistical language modelling toolkit", o mediante la herramienta en línea conocida como "LM-TOOL". Todo esto en el apéndice C.

Respecto al sistema HTK, se diseñaron dos sistemas de reconocimiento de voz creados para ser usados con el corpus CIEMPIESS y con Mexbet. Se trata del sistema "HTK2SPHINX-CONVERTER"(Hernández-Mena & Herrera-Camacho, 2015a) creado para hacer reconocimiento mediante una gramática (ver apéndice D), y el sistema

"HTK-BENCHMARK", que fue creado para hacer reconocimiento mediante un modelo de lenguaje de n-gramas (ver apéndice E).

Se deja constancia en el capítulo 6 de un experimento hecho con el novedoso sistema Kaldi, que desplazará a Sphinx como el sistema de reconocimiento de voz imperante. Los estudios realizados a estos dos sistemas y al HTK, verifican que el sistema Kaldi ha optimizado su código respecto a los otros dos; en particular la introducción de "Finite State Transducers" (FST) en el sistema Kaldi, mejora el desempeño del modelo de lenguaje.

Otra contribución importante de este trabajo es la inclusión de palabras con desplazamiento acentual en la definición de PFS, dado su efecto en el rendimiento de un sistema ASR. Esto permitió en los experimentos, encontrar una mayor cantidad de PFS. Por lo tanto, también se concluye que una de las mejores maneras para minimizar el impacto negativo de las PFS en sistemas ASR, es construyendo un buen modelo de lenguaje, que sea rico y un diccionario de pronunciación que incluya las vocales tónicas de todas sus entradas.

No obstante, un experimento importante a futuro es determinar el conjunto óptimo de categorías fonéticas para un cierto sistema ASR, en condiciones distintas de prueba respecto al corpus de entrenamiento y al paradigma de clasificación.

Finalmente, el experimento que integra todas las contribuciones mencionadas anteriormente, es aquel en el que se creó un diccionario adicional con palabras sin la vocal tónica indicada. Se le aplicó al sistema Sphinx3 la base de datos CIEMPIESS con el diccionario normal (con vocales tónicas indicadas) y el señalado anteriormente. El experimento muestra una mejora del 1.7 % de precisión cuando se considera este tipo de PFS (palabras con sílaba tónica indicada), lo que se considera la mayor aportación de este trabajo para el desempeño de los sistemas ASR. Cabe señalar que esta es una formulación novedosa para el reconocimiento de voz continua.

Apéndice A

Tutorial de Sphinx 3

El sistema CMU-Sphinx 3 (o simplemente Sphinx para abreviar) es un sistema de reconocimiento automático de voz basado en modelos ocultos de Markov, escrito en C/C++, y que fue desarrollado por la universidad Carnegie Mellon en la década de los noventas. Este sistema permite hacer reconocimiento robusto de voz continua, independiente del hablante, que puede manejar grandes vocabularios y que es capaz de funcionar tanto en modo "batch" (reconocimiento por lotes o fuera de línea) o en modo "live" (reconocimiento en vivo o en tiempo real).

En esta sección se presenta un pequeño tutorial que explica al lector como obtener, instalar y configurar Sphinx para hacer reconocimiento de voz en español de México, en modo "batch" y en ambiente Linux. Este tutorial esta basado parcialmente en otro distribuido libremente por los desarrolladores de Sphinx y que puede consultarse en la siguiente dirección web:

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

A.1 Requerimientos del sistema

Es necesario tener previamente instalado el siguiente software en el sistema operativo donde se pretende ejecutar Sphinx3:

- Perl 5 o mayor
- Un compilador de C. El compilador **gcc** es la opción sugerida aquí.

A.2 Descarga del sistema

Para que el sistema Sphinx funcione con todas sus capacidades, requiere de trabajar con ciertas librerías de funciones y un software de entrenamiento. A continuación se enlistan los paquetes que deberán descargarse como parte de la instalación de Sphinx y se muestra la liga para su descarga directa por internet.

- Sphinx 3. http://cmusphinx.org/download/nightly/sphinx3.nightly.tar.gz
- SphinxBase. http://cmusphinx.org/download/nightly/sphinxbase.nightly.tar.gz
- SphinxTrain. http://cmusphinx.org/download/nightly/SphinxTrain.nightly.tar.gz

Adicionalmente se pide descargar un paquete especial con un corpus en idioma inglés llamado "AN4" y que fue creado por la Carnegie Mellon para propósitos didácticos. Este paquete contiene los audios del corpus y todos los archivos necesarios para configurar Sphinx de tal modo que se puedan realizar experimentos de entrenamiento y reconocimiento con ese mismo corpus.

- Corpus AN4. http://www.speech.cs.cmu.edu/databases/an4/an4_sphere.tar.gz

Posteriormente se explicará como reconfigurar Sphinx para sustituir este corpus en inglés por otro que este en español de México.

A.3 Instalación de Sphinx 3

Se pide que todos los paquetes descargados en el apartado anterior se pongan en una sola carpeta llamada **tutorial**, ubicada en cualquier parte del sistema operativo. Es decir que en este momento, la carpeta **tutorial** debe contener los siguientes elementos:

```
- tutorial
    * an4_sphere.tar.gz
    * sphinx3.nightly.tar.gz
    * sphinxbase.nightly.tar.gz
    * SphinxTrain.nightly.tar.gz
```

Ahora bien, a partir de este momento se darán las instrucciones para instalar Sphinx 3 y todos sus elementos por medio de la terminal de Linux. Por lo tanto, lo primero que hay que hacer es posicionarse en la carpeta **tutorial** de esta manera:

```
$ cd tutorial
```

A.3.1 Preparando los archivos para este tutorial

El paquete AN4 contiene archivos expresamente hechos para enseñar a los usuarios cómo debe trabajarse Sphinx, pero primero es preciso descomprimir el paquete AN4:

```
$ gunzip -c an4_sphere.tar.gz | tar xf -
```

Después se descomprime también el software llamado SphinxTrain:

```
$ gunzip -c SphinxTrain.nightly.tar.gz | tar xf -
```

En este punto, el contenido de la carpeta **tutorial** debe verse más o menos así:

```
- tutorial
   * an4
   * an4_sphere.tar.gz
   * sphinx3.nightly.tar.gz
   * sphinxbase.nightly.tar.gz
   * SphinxTrain
   * SphinxTrain.nightly.tar.gz
```

Si todo esta en orden, es preciso compilar SphinxTrain:

```
$ cd SphinxTrain
$ configure
$ make
```

Y finalmente, se le ordena a SphinxTrain que copie los archivos necesarios para este tutorial en la carpeta **an4** de la siguiente manera:

```
$ perl scripts_pl/setup_tutorial.pl an4
$ cd ..
```

A.3.2 Preparando Sphinx 3

Una vez que el tutorial esta listo, hay que descomprimir SphinxBase y Sphinx 3:

```
$ gunzip -c sphinxbase.nightly.tar.gz | tar xf -
$ gunzip -c sphinx3.nightly.tar.gz | tar xf -
```

Para luego compilarlos e instalarlos:

```
$ cd sphinxbase
$ ./autogen.sh
$ ./configure
$ make
$ cd..
$ cd sphinx3
$ ./configure --prefix=‘pwd‘/build --with-sphinxbase=‘pwd‘/../sphinxbase
$ make
$ make install
$ cd ..
```

En este punto debemos estar situados en la raíz de la carpeta **tutorial**, la cual debe contener los siguientes elementos:

```
- tutorial
    * an4
    * an4_sphere.tar.gz
    * SphinxTrain
    * SphinxTrain.nightly.tar.gz
    * sphinx3
    * sphinx3.nightly.tar.gz
    * sphinxbase
    * sphinxbase.nightly.tar.gz
```

Finalmente, es preciso pedir a cierto script dentro de la carpeta **sphinx3** que copie algunos archivos necesarios para este tutorial a la carpeta **an4**:

```
$ cd sphinx3
```

```
$ perl scripts/setup_tutorial.pl an4
$ cd ..
```

A.3.3 Ejecutar una ronda de entrenamiento

Antes de poder efectuar cualquier experimento de reconocimiento, es necesario entrenar el sistema con los datos del corpus que en este caso es el corpus AN4. Para lograr esto se ingresa a la carpeta **an4**:

```
$ cd an4
```

Ahora bien, debido a que el corpus AN4 contiene archivos de audio en formato NIST, es necesario tomarlos para calcular los vectores MFCC de cada uno de ellos. Para lograrlo se hace uso de un script llamado "make_feats.pl" que se invoca de la siguiente manera:

```
$ perl scripts_pl/make_feats.pl  -ctl etc/an4_train.fileids
```

Una vez hecho esto, puede comenzar la ronda de entrenamiento simplemente tecleando el siguiente comando:

```
$ perl scripts_pl/RunAll.pl
```

Dependiendo de la computadora en donde se esté ejecutando, este proceso podría tardar varios minutos en finalizar.

A.3.4 Ejecutar una ronda de reconocimiento

Una vez finalizada la ronda de entrenamiento, es posible ejecutar ahora una ronda de reconocimiento de los 127 archivos de audio que el corpus AN4 tiene dedicados específicamente para ello.

Pero antes, el mismo problema que había en la ronda de entrenamiento ocurre aquí; como los audios están en formato NIST, es preciso calcular sus vectores MFCC de la siguiente manera:

```
$ perl scripts_pl/make_feats.pl  -ctl etc/an4_test.fileids
```

Y finalmente, ejecutar la ronda de reconocimiento así:

```
$ perl scripts_pl/decode/slave.pl
```

Al final de este proceso (que dependiendo de la computadora en el que se haga podría durar varios minutos), se verá en la terminal el siguiente mensaje, informando el error de palabra o "Word Error Rate" (WER) y el error de oración o "Sentence Error Rate" (SER) obtenido en la ronda de reconocimiento:

```
SENTENCE ERROR: 56.154% (73/130)   WORD ERROR RATE: 16.429% (127/773)
```

Ahora bien, si se ha alcanzado este punto exitosamente, el diagnóstico es que Sphinx esta funcionando correctamente dentro del sistema operativo y por lo tanto, será posible seguir sin ningún problema la siguiente parte de este tutorial en donde se explica detalladamente como configurar Sphinx para mejorar su desempeño y para adaptarle un corpus en español de México.

A.4 Reconociendo español de México con Sphinx 3

En este apartado se pide descargar el corpus CIEMPIESS, que es un corpus en español de México de 17 horas de duración, extraído de Radio-IUS, una estación de radio de la Facultad de Derecho de la UNAM. El corpus CIEMPIESS viene presentado en un formato que lo hace ideal para adaptarlo fácilmente al sistema Sphinx.

El corpus CIEMPIESS se distribuye libremente mediante una licencia "Creative Commons Atribución-CompartirIgual 4.0 Internacional" y puede descargarse de forma gratuita en la siguiente dirección electrónica:

http://www.ciempiess.org/downloads

A.4.1 Crear un nuevo experimento

Antes de empezar a modificar los archivos internos de Sphinx, es preciso generar un nuevo experimento. Esto es, una carpeta análoga a la carpeta **an4**, pero con otro nombre. Para ello, estando dentro de la carpeta **an4** aplicamos el comando:

```
$ perl scripts_pl/copy_setup.pl -task TESIS
```

En este caso "TESIS" es el nombre del proyecto. Por lo tanto, este comando generará una carpeta llamada **TESIS** con los mismos archivos que la carpeta **an4** tenia justo antes de ejecutar la ronda de entrenamiento descrita en este tutorial, con la diferencia de que los archivos que en la carpeta **an4** tenían el prefijo "an4", se copiarán con el prefijo "TESIS" a la carpeta **TESIS**.

Este mismo procedimiento de crear un nuevo experimento de Sphinx resultará útil toda vez que se quiera realizar una nueva ronda de entrenamiento y reconocimiento sin alterar un proyecto previo. Ahora bien, al final de este tutorial se pretende que se haya ejecutado una ronda de entrenamiento y una ronda de reconocimiento con el corpus CIEMPIESS en la carpeta **TESIS**. Esto quiere decir que si este mismo comando se ejecuta, pero ahora desde la carpeta **TESIS**, es decir, algo así:

```
$ cd TESIS
$ perl scripts_pl/copy_setup.pl -task TESIS2
```

Ahora en la carpeta **TESIS2** se copiará con el corpus CIEMPIESS, en vez de copiarse con el corpus AN4. Lo cual puede resultar muy cómodo para los experimentadores que en un momento dado, sólo estén interesados en hacer reconocimiento con el corpus CIEMPIESS.

A.4.2 Configurar el archivo "sphinx_train.cfg"

Dentro de la carpeta **TESIS** existe una una carpeta llamada **etc**. Esta carpeta es muy importante por que contiene todos los archivos de configuración de usuario. La mayoría de estos archivos comenzarán como el prefijo "TESIS". Por ejemplo, "TESIS.dic", "TESIS.phone", "TESIS_train.transcription", etc. A excepción de los archivos "feat.params", "sphinx_decode.cfg" y "sphinx_train.cfg".

En esta sección se explica cuáles son las líneas del archivo "sphinx_train.cfg" que se deben modificar para mejorar el rendimiento de Sphinx dependiendo del equipo en donde se ejecute y para adaptar el corpus CIEMPIESS.

Como su nombre lo indica, el archivo "sphinx_train.cfg" esta relacionado con la configuración de la fase de entrenamiento de Sphinx y el archivo "sphinx_decode.cfg" (que se explica en la siguiente sección) esta relacionado con la fase de reconocimiento. Por otra parte el archivo "feat.params" contiene los parámetros que se desea que sean tomados en cuenta a la hora de convertir los archivos de audio en archivos de vectores MFCC. En la terminología de Sphinx, al módulo que efectua tal conversión se le conoce con el nombre de "front-end".

Dicho lo anterior, se muestran a continuación las líneas específicas que del archivo "sphinx_train.cfg" deben modificarse, mostrando los nuevos valores que deben llevar.

En primer lugar, los archivos de audio del corpus CIEMPIESS están en formato SPHERE de NIST[A.1] y tienen una extensión ".sph", por lo tanto:

```
$CFG_WAVFILE_EXTENSION = 'sph';
```

[A.1] NIST es el acrónimo para "National Institute of Standards and Technology"

Tabla A.1: Número de senones por horas de entrenamiento

Entrenamiento en Horas	No. de Senones
1-3	500-1000
4-6	1000-2500
6-8	2500-4000
8-10	4000-5000
10-30	5000-5500

```
$CFG_WAVFILE_TYPE = 'nist';
```

Para el corpus CIEMPIESS, el número de estados ligados o senones, se calcula con base en la tabla A.1, que proporcionan los propios desarrolladores de Sphinx.

En el entendido de que el corpus CIEMPIESS consta de 17 horas para entrenamiento, el número de estados ligados recomendado en este tutorial es de 5000:

```
$CFG_N_TIED_STATES = 5000;
```

Ahora bien, para aprovechar al máximo el número de núcleos de la computadora en dónde Sphinx esta siendo ejecutado, se tienen que especificar el número total de núcleos con los que cuenta(n) el/los procesador(es) local(es). Por ejemplo, para dos procesadores con cuatro núcleos cada uno, el valor que se pondría es 8:

```
# How many parts to run Forward-Backward estimation in
$CFG_NPART = 8;
```

Finalmente, para indicarle a Sphinx que debe paralelizar sus tareas con base en el número de núcleos especificado arriba se debe asegurar que las siguientes líneas se vean justo así:

```
# Queue::POSIX for multiple CPUs on a local machine
# Queue::PBS to use a PBS/TORQUE queue
$CFG_QUEUE_TYPE = "Queue::POSIX";
```

Todo lo demás en el archivo "sphinx_train.cfg" permanece inalterado.

A.4.3 Configurar el archivo "sphinx_decode.cfg"

Como ya se ha dicho, en la misma carpeta **etc** de la carpeta **TESIS** esta el archivo llamado "sphinx_decode.cfg", el cual se encarga de configurar la fase de reconocimiento de Sphinx.

La única modificación que sufre para optimizar su desempeño es que debe especificarse el número total de núcleos con los que cuenta el/los procesador(es) local(es). Siguiendo el ejemplo de la sección anterior, para dos procesadores de cuatro núcleos cada uno, debe especificarse el siguiente valor:

```
$DEC_CFG_NPART = 8; #  Define how many pieces to split decode in
```

Todo lo demás en el archivo "sphinx_decode.cfg" permanece inalterado.

A.4.4 Incorporar los archivos del CIEMPIESS al proyecto

En esta sección se explica como incorporar físicamente los archivos necesarios para que Sphinx pueda trabajar con el corpus CIEMPIESS, en vez de con el corpus AN4. Este proceso consiste únicamente en borrar ciertos archivos y carpetas en la carpeta **TESIS** y sustituirlos por otros archivos y carpetas del corpus CIEMPIESS.

Dentro de la carpeta **TESIS** se encuentra una carpeta llamada **wav**. Luego, es preciso borrar totalmente el contenido de la carpeta **wav** hasta que quede con cero elementos.

Ahora bien, el corpus CIEMPIESS se encuentra contenido en una carpeta llamada **CIEMPIESS_SPH**. Dentro de ella se pueden hallar dos carpetas llamadas **train** y **test** que deben ser copiadas dentro de la carpeta **wav** mencionada arriba. Estas dos carpetas **train** y **test** contienen los archivos de audio del corpus CIEMPIESS y como sus nombres indican, están clasificados como archivos para el entrenamiento (train) y archivos para efectuar pruebas de reconocimiento (test) en modo "batch". Es importante mencionar aquí que los archivos en ambas carpetas son todos diferentes, para evitar reconocer los mismos archivos con los que se entrena.

Luego de esto, hay que inspeccionar la carpeta **etc** que se encuentra dentro de la carpeta **TESIS**. En ella se encontrarán los siguientes archivos:

1. TESIS.dic
2. TESIS.filler
3. TESIS.phone
4. TESIS.ug.lm

5. TESIS.ug.lm.DMP
6. TESIS_test.fileids
7. TESIS_test.transcription
8. TESIS_train.fileids
9. TESIS_train.transcription
10. feat.params
11. sphinx_decode.cfg
12. sphinx_train.cfg

Después, se debe comprobar que dentro de la carpeta **CIEMPIESS_SPH** existe una carpeta **sphinx_experiments**, y que dentro de ella esta otra carpeta llamada **T22_TONIC** con los siguientes archivos:

1. T22CT_CIEMPIESS.dic
2. T22CT_CIEMPIESS.filler
3. T22CT_CIEMPIESS.phone
4. T22CT_CIEMPIESS.ug.lm
5. T22CT_CIEMPIESS.ug.lm.DMP
6. T22CT_CIEMPIESS_test.fileids
7. T22CT_CIEMPIESS_test.transcription
8. T22CT_CIEMPIESS_train.fileids
9. T22CT_CIEMPIESS_train.transcription
10. feat.params

Salta a la vista que los archivos en ambas carpetas son casi los mismos, excepto que tienen diferentes prefijos en los nombres de archivo. Siendo así que en la carpeta **etc** los archivos comienzan por el prefijo "TESIS" y los archivos en la carpeta **T22_TONIC** comienzan con el prefijo "T22CT_CIEMPIESS".

Por lo tanto, lo que hay que hacer es borrar todos los archivos en la carpeta **etc**, a excepción de los archivos "sphinx_decode.cfg" y "sphinx_train.cfg". Luego de esto, hay que copiar todos los archivos de la carpeta **T22_TONIC** a la carpeta **etc** y cambiarles el prefijo "T22CT_CIEMPIESS" por el de "TESIS" a los nombres de archivo.

Si este proceso se efectua correctamente, significa que ya se tiene todo lo necesario para que Sphinx pueda aceptar sin problemas al corpus CIEMPIESS. Lo que resta ahora es repetir dos de los primeros pasos de este tutorial que están bajo los títulos de:

- Ejecutar una Ronda de Entrenamiento
- Ejecutar una Ronda de Reconocimiento

Al final de estos procesos, se obtiene un sistema Sphinx entrenado para reconocer español de México en vez de reconocer inglés.

Apéndice B

Características del corpus CIEMPIESS

En esta sección se explica detalladamente la arquitectura del corpus CIEMPIESS, esto es, la organización de todos los archivos que lo conforman y su utilidad.

El corpus CIEMPIESS fue creado en el Laboratorio de Tecnologías de Lenguaje (LTL) de la Facultad de Ingeniería (FI) de la Universidad Nacional Autónoma de México (UNAM) entre los años de 2012 y 2014 por Carlos Daniel Hernández Mena, estudiante de doctorado del Programa de Maestría y Doctorado en Ingeniería de la UNAM, supervisado por el Doctor José Abel Herrera Camacho. CIEMPIESS es el acrónimo para:

"Corpus de Investigación en Español de México del Posgrado de Ingeniería Eléctrica y Servicio Social"

El CIEMPIESS es un corpus oral extraído de la radio que fue originalmente diseñado para crear modelos acústicos para uso en reconocimiento automático de voz. Se compone de grabaciones de conversaciones espontáneas entre el/los moderador(es) del programa y sus invitados.

Estas grabaciones fueron descargadas en formato MP3 de la página "PODCAST UNAM"[B.1] y fueron originalmente creadas por "RADIO-IUS"[B.2], que es una estación de radio que pertenece a la Facultad de Derecho de la UNAM.

El corpus CIEMPIESS es un producto de software libre y esta bajo un licencia "Creative Commons Atribución-CorpartirIgual 4.0 Internacional". Puede ser descargado gratuitamente de:

http://www.ciempiess.org/downloads

[B.1] http://podcast.unam.mx/

[B.2] http://www.derecho.unam.mx/cultura-juridica/radio.php

Y también del Linguistic Data Consortium en la siguiente dirección:

https://catalog.ldc.upenn.edu/LDC2015S07

Para consultar datos estadísticos útiles medidos directamente del Corpus CIEMPIESS se puede ingresar a la siguiente dirección:

http://www.ciempiess.org/CIEMPIESS_Statistics.html

Es importante mencionar que el corpus CIEMPIESS fue diseñado para utilizarse con el sistema de reconocimiento automático de voz CMU Sphinx 3 y eso explica en gran medida la organización de sus archivos y la forma de nombrarlos. Sin embargo, es posible utilizar el corpus CIEMPIESS con cualquier otro sistema, haciendo pequeñas modificaciones a sus archivos.

B.1 Organización de los archivos del corpus CIEMPIESS

Al descargar y descomprimir el corpus CIEMPIESS se tendrá una carpeta llamada **CIEMPIESS_SPH**, la cual contiene los siguientes archivos y carpetas:

- Carpeta **transcriptions**
- Carpeta **train**
- Carpeta **test**
- Carpeta **textgrids**
- Carpeta **sphinx_experiments**
- Archivo **LICENSE.txt**
- Archivo **README.txt**

A continuación se presenta una explicación detallada de la utilidad de cada carpeta listada.

B.1.1 Carpeta "transcriptions"

En esta carpeta se encuentran los siguientes archivos que trabajan en pares:

- CIEMPIESS_FULL_TRAIN.transcription

- CIEMPIESS_FULL_TRAIN.fileids

- CIEMPIESS_test.transcription
- CIEMPIESS_test.fileids

- CIEMPIESS_train.transcription
- CIEMPIESS_train.fileids

Primero que nada, para entender los archivos "FULL_TRAIN" se requiere una explicación anecdótica: Cuando una primera versión del corpus CIEMPIESS fue terminada en Diciembre de 2013, contenía un total de 16,717 archivos de audio con sus respectivas transcripciones. Estos audios "primigenios" pueden identificarse fácilmente por que tienen claves como estas:

```
0173M_09ALX_22OCT12
0177M_09ALX_22OCT12
0020M_11ALX_10DIC12
0001F_12ALX_17DIC12
0009F_12ALX_17DIC12
```

En donde la "F" indica que el archivo contiene una voz de mujer (**F**emenina) y la "M" indica que se trata de un archivo de audio con voz de hombre (**M**asculina).

Después de analizar esta primera versión no liberada del corpus CIEMPIESS, se decidió que se seleccionarían dos conjuntos de audios, uno para la fase de entrenamiento (train) y otro para una fase de pruebas de reconocimiento (test). Por lo tanto, se seleccionaron 700 de estos audios "primigenios" para el conjunto de pruebas y se le añadieron otros 300 archivos de audio provenientes de diversos corpus ya existentes en el laboratorio. Estos audios añadidos pueden identificarse fácilmente por que tienen claves como:

```
F09MAY1844_0036
F09MAY1844_0038
AB01_1
AB01_4
OSC_001
OSC_003
```

Por lo tanto, los archivos que en su nombre contengan el sufijo "_train" son los que pertenecen al conjunto de entrenamiento y los archivos con el sufijo "_test" pertenecen al conjunto de pruebas.

Ahora bien, los archivos con la extensión ".transcription" contienen las transcripciones de los archivos de audio y pueden ser ligadas a ellos por medio de las claves que aparecen entre paréntesis. Los archivos con la extensión ".fileids" son los archivos de rutas y contienen la ruta para ubicar cada uno de los archivos de audio dentro de las carpetas **train** y **test** del corpus CIEMPIESS.

Finalmente, los archivos "FULL_TRAIN" contienen las transcripciones y las rutas, sólo de los 16,717 archivos "primigenios" sin diferenciarlos como archivos de entrenamiento o de pruebas.

B.1.2 Carpeta "train"

En la carpeta **train** se encuentran las siguientes subcarpetas:

- ALX_TRAIN
- ANG_TRAIN
- MAB_TRAIN

Sucede que todos los audios "primigenios" del CIEMPIESS fueron extraídos por tres diferentes trabajadores: Alejandro (ALX), Ángel (ANG) y Mabel (MAB). Y esa es la razón por la que estas subcarpetas tienen estos nombres.

Dentro de cada una de estas subcarpetas se encuentran archivos de audio necesarios para la etapa de entrenamiento. Todos estos archivos tienen un nombre de archivo codificado de la siguiente manera:

```
Clave de archivo: 0001M_01ALX_17DIC12
```

En donde:

- **0001**: Se trata del archivo número 0001 dentro de la carpeta "01".
- **M_**: Género del hablante: "M" para Masculino y "F" para Femenino.
- **01ALX_**: Esta es la carpeta número 01 que fue procesada por el trabajador "ALX".
- **17DIC12**: Fecha en que la carpeta "01" fue creada.

B.1.3 Carpeta "test"

Como ya se había mencionado, el conjunto de audios de prueba (test) proviene de diferentes corpus. Por lo tanto, la carpeta **test** se organiza en subcarpetas que delatan el origen de los audios que contienen. Estas carpetas se describen a continuación:

- **ciempiess**: Aquí se encuentran los 700 audios "primigenios" que fueron extraídos de la primera versión del corpus CIEMPIESS. Todos estos audios tienen codificado su nombre de archivo de la misma manera que se explicó arriba.
- **description**: Esta carpeta contiene 200 grabaciones espontáneas de personas describiendo imágenes y contestando preguntas sobre temas diversos.
- **fm**: Esta carpeta contiene 17 grabaciones extraídas de la radio FM. Las estaciones elegidas para estos audios son diferentes a las estaciones de donde se extrajo el corpus CIEMPIESS.
- **read**: Esta carpeta contiene grabaciones de una persona leyendo frases. El hablante en estas grabaciones es una persona del sexo masculino, de entre 25 y 30 años de edad que ha vivido toda su vida en la ciudad de México.

B.1.4 Carpeta "textgrids"

En la carpeta **textgrids** se encuentran los siguientes archivos y subcarpetas:

- Carpeta **full_train**
- Archivo **CIEMPIESS_FULL_TRAIN.label_transcriptions**
- Archivo **CIEMPIESS_FULL_TRAIN.label_fileids**

Ahora bien, una de las razones por las que la construcción del corpus CIEMPIESS tomó tanto tiempo, es por que cuenta con "etiquetas de tiempo" que indican el inicio y el fin de cada palabra de cada archivo de audio "primigenio", contenido en el corpus CIEMPIESS.

Estas "etiquetas de tiempo" fueron creadas con un software libre llamado PRAAT[B.3]. En PRAAT se generan estos archivos de texto llamados "etiquetas de tiempo" con extensión ".TextGrid". Es por lo tanto de aquí de donde la carpeta **textgrids** toma su nombre.

Esto quiere decir que existen un total de 16,717 "etiquetas de tiempo", una por cada archivo de audio "primigenio" en el corpus CIEMPIESS.

[B.3] Más información en www.praat.org

Finalmente, un detalle muy importante para hacer notar, es el hecho de que el archivo "CIEMPIESS_FULL_TRAIN.label_transcriptions" contenido en esta misma carpeta **textgrids** fue creado automáticamente, con ayuda de un script en Python, lo cual quiere decir, que es un reflejo exacto de todas las "etiquetas de tiempo".

Este hecho hace que el archivo "CIEMPIESS_FULL_TRAIN.label_transcriptions" sea muy parecido al archivo "CIEMPIESS_FULL_TRAIN.transcription", contenido en la carpeta **transcriptions**. De hecho, en un principio estos dos archivos eran exactamente iguales, sin embargo, ahora han dejado de serlo debido a que este último archivo ha sufrido correcciones ortográficas, mientras que el primero ha permanecido inalterado. La razón es que si se modifica una transcripción en el archivo "CIEMPIESS_FULL_TRAIN.label_transcriptions", el cambio tendría que verse reflejado en la correspondiente "etiqueta de tiempo" y viceversa. Desgraciadamente, el proyecto CIEMPIESS no ha contado aún con la gente necesaria para hacer estas correcciones que serían particularmente laboriosas.

B.1.5 Carpeta "sphinx_experiments"

En la carpeta **sphinx_experiments** se encuentran las siguientes subcarpetas:

- T22_NOTONIC
- T22_TONIC
- T50_NOTONIC
- T50_TONIC

Todas estas subcarpetas contienen todos los archivos de configuración de usuario necesarios para efectuar experimentos de reconocimiento de voz con el software CMU-Sphinx 3.

Las carpetas con el sufijo "TONIC" contienen archivos en donde se toma en cuenta la vocal tónica de todas las palabras contenidas tanto en los archivos de transcripción como en el diccionario de pronunciación. Esto implica que los archivos con el sufijo "NOTONIC", no tomen en cuenta las vocales tónicas en ninguna palabra.

Las carpetas con el prefijo "T22" tiene un diccionario de pronunciación en alfabeto Mexbet nivel T29, y las carpetas con el prefijo "T50" tienen un diccionario de pronunciación en alfabeto Mexbet nivel T66.

De manera general, todas las carpetas contienen los siguientes tipos de archivos:

1. Archivo **feat.params**: Contiene parámetros útiles para configurar el proceso de creación de archivos de atributos MFCC.

2. Archivo **.dic**: Diccionario de pronunciación.
3. Archivo **.filler**: Diccionario de relleno.
4. Archivo **.phone**: Lista de fonemas del alfabeto fonético utilizado.
5. Archivo **.ug.lm** : Modelo de lenguaje versión texto en formato ARPA.
6. Archivo **.ug.lm.DMP**: Modelo de lenguaje versión binaria en formato ARPA.
7. Archivo **_test.fileids**: Archivo de rutas de los archivos de audio del conjunto de pruebas.
8. Archivo **_test.transcription**: Archivo de transcripción de los archivos de audio del conjunto de pruebas.
9. Archivo **_train.fileids**: Archivo de rutas de los archivos de audio del conjunto de entrenamiento.
10. Archivo **_train.transcription**: Archivo de transcripción de los archivos de audio del conjunto de entrenamiento.

Es preciso hacer notar que en los archivos con sufijo "TONIC" se marca la vocal tónica y otras grafías especiales con un doble caracter en mayúsculas. Por ejemplo: AAbre, bloquEEen, enIIgmas, OObras, ajUUsco, etc. La razón es que ni Sphinx, ni la LMTOOL, ni el CMU SLMTK (ver apéndice C) distinguen entre mayúsculas y minúsculas. Esto es un problema por que palabras como "Opera" y "opEra" serían interpretadas por estas herramientas como la misma palabra. Por lo tanto, para entender las transcripciones del corpus CIEMPIESS es preciso hacer unas ligeras modificaciones a sus palabras, del tipo: "buscar y reemplazar". Estas modificaciones implican cambiar:

- AA por A
- EE por E
- II por I
- OO por O
- UU por U
- KKSS por KS
- JJ por J
- SS por SS
- NN por N

En donde KS, J, S son caracteres especiales para manejar los diferentes contextos de la grafía “x” y N para manejar la grafía “ñ”, que no corresponde con ninguno de los 128 caracteres ASCII.

Apéndice C

Herramientas para modelado de lenguaje

En esta sección se muestran dos herramientas para crear modelos de lenguaje útiles para el sistema Sphinx. Se trata del software llamado "LMTOOL" que es una herramienta en línea y del software "CMU SLM Toolkit" que es una herramienta para ejecución en sistemas tipo UNIX/Linux. Ambas herramientas fueron creadas por la Universidad Carnegie Mellon y son distribuidas bajo una licencia libre y de código abierto.

C.1 LMTOOL

La LMTOOL es una herramienta en línea que sirve para crear archivos de modelo de lenguaje compatibles con el sistema Sphinx y puede manejar corpus pequeños de unas cuantas docenas o unos pocos cientos de oraciones. Para acceder a ella es preciso ingresar a esta dirección web:

http://www.speech.cs.cmu.edu/tools/lmtool.html

Aquí se encuentra un botón que permite cargar un archivo de tipo texto con la característica de tener una oración por cada renglón.

Este archivo se sube a la herramienta en línea y poco tiempo después la página devuelve un link para descargar una carpeta comprimida con los siguientes archivos:

- Archivo **.corpus**: No es más que una copia del archivo que se le proporcionó a la LMTOOL (archivo de entrada) por medio del botón de carga.
- Archivo **.dic**: Este es un diccionario de pronunciación que contiene una lista de todas las palabras (sin repetir) extraídas del archivo de entrada con su respectiva pronunciación en idioma inglés. Esta lista esta orden alfabético.

- Archivo **.lm**: Archivo ASCII que contiene el modelo de lenguaje en un formato estándar conocido como **ARPA**. Para utilizar este archivo en el sistema Sphinx se le pone adicionalmente a la extensión **.lm**, la extensión **.arpa** de manera que queda un archivo con la extensión **.lm.arpa**.
- Archivo **.sent**: Contiene las oraciones del archivo **.corpus**, pero delimitadas por las etiquetas `<s>` y `</s>`.
- Archivo **.sent.arpabo**: Es una copia exacta del archivo **.lm**.
- Archivo **.token**: Contiene una lista de las palabras del corpus indicando su número de ocurrencias.
- Archivo **.vocab**: Contiene una lista de las palabras del corpus sin repetición, pero sin ninguna otra indicación.

Como puede verse, el único archivo generado por la LMTOOL que servirá para configurar al sistema Sphinx para el español de México es el archivo **.lm** que debe convertirse a un formato binario. Este proceso se explica al final del presente apéndice.

C.2 CMU statistical language modeling toolkit v2

Esta es una herramienta que sirve para construir modelos de lenguaje para corpus grandes ya que para corpus pequeños se tiene a la lmtool.

Para descargar la CMU SLM Toolkit se debe ingresar a la siguiente dirección web:

http://www.speech.cs.cmu.edu/SLM/toolkit.html

La documentación para esta herramienta se encuentra en:

http://mi.eng.cam.ac.uk/~prc14/toolkit_documentation.html

Una vez descargada, se descomprime una carpeta llamada **CMU_Cam_Toolkit_v2**. Dentro de ella se encuentran varias carpetas, un archivo de texto llamado **README** y un script llamado **endian.sh**. El archivo README contiene las instrucciones de instalación y el script sirve para determinar si la computadora en la que se instalará este software es de tipo "Little Endian" o de tipo "Big Endian". Las máquinas con Linux son de tipo "Little Endian", por lo tanto, el archivo README dice que debe modificarse el "makefile" dentro de la carpeta **src**.

Después de modificar este "makefile" la instalación en Linux es realmente muy sencilla, solo hay que teclear lo siguiente desde la terminal:

```
$ cd src
$ make install
```

Lo más importante después de ejecutar estos comandos es que se crean dentro de la carpeta **bin** un total de doce archivos ejecutables, los cuales son:

1. binlm2arpa
2. evallm
3. idngram2lm
4. idngram2stats
5. interpolate
6. mergeidngram
7. ngram2mgram
8. text2idngram
9. text2wfreq
10. text2wngram
11. wfreq2vocab
12. wngram2idngram

El diagrama C.1 muestra el uso típico de esta herramienta para crear un modelo de lenguaje. Los rectángulos representan algunos de los ejecutables generados en la carpeta **bin** en el paso anterior.

Para propósitos de esta explicación se colocará la carpeta "CMU_Cam_Toolkit_v2" en una ruta no muy profunda:

```
/home/carlos/CMU_Cam_Toolkit_v2
```

y se creará una carpeta llamada "Tesis_LM" en la ruta:

```
/home/carlos/Tesis_LM
```

[C.1] Imagen extraída de la misma documentación de la "CMU SLM Toolkit"

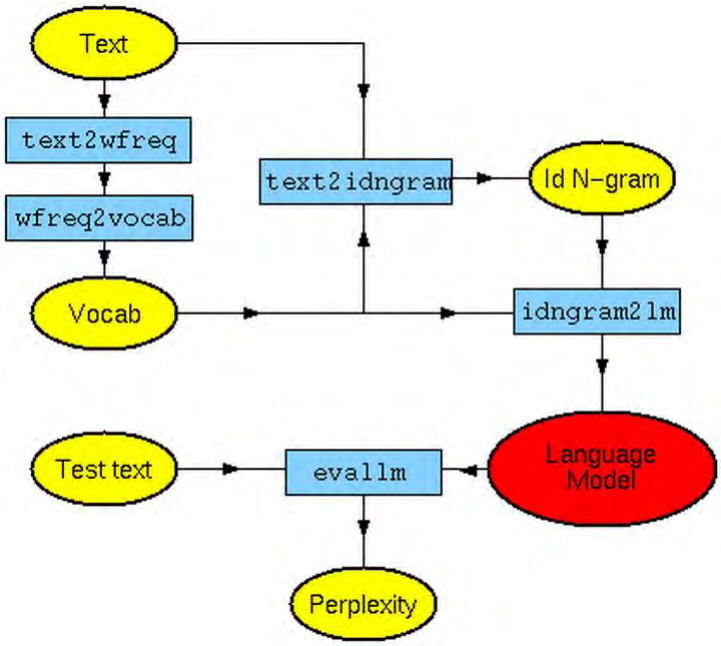

Figura C.1: Creación de un modelo de lenguaje con la "CMU SLM Toolkit".[C.1]

en donde se depositará el archivo de tipo texto que en el diagrama C.1 se indica como "text" y que esta en un óvalo amarillo. Este archivo puede ser el mismo archivo de entrada que se mande a la LMTOOL y debe tener las misma característica de contener una oración por cada renglón.

A continuación se sigue el diagrama C.1 paso por paso, tomando en consideración que nos encontramos posicionados en la ruta:

```
/home/carlos/CMU_Cam_Toolkit_v2/bin
```

C.2.1 Ejecutable "text2wfreq"

Parámetros:

```
text2wfreq [ -hash 1000000 ]
           [ -verbosity 2 ]
           < .text > .wfreq
```

La sintaxis exacta utilizada para este ejemplo es:

```
$ ./text2wfreq </home/carlos/Tesis_LM/Tesis_train.text> /home/
carlos/Tesis_LM/Tesis_train.wfreq
```

Y se crea el archivo **Tesis_train.wfreq** que muestra una lista de palabras del corpus sin repetir con su número de ocurrencias.

C.2.2 Ejecutable "wfreq2vocab"

Parámetros:

```
wfreq2vocab [ -top 20000 | -gt 10]
            [ -records 1000000 ]
            [ -verbosity 2]
            < .wfreq > .vocab
```

Sintaxis:

```
$ ./wfreq2vocab </home/carlos/Tesis_LM/Tesis_train.wfreq> /home/
carlos/Tesis_LM/Tesis_train.vocab
```

Y se crea el archivo **Tesis_train.vocab** que muestra una lista de palabras del corpus sin repetir y sin mostrar el número de ocurrencias.

C.2.3 Ejecutable "text2idngram"

Parámetros:

```
text2idngram -vocab .vocab
          [ -buffer 100 ]
          [ -temp /usr/tmp/ ]
```

```
          [ -files 20 ]
          [ -gzip | -compress ]
          [ -n 3 ]
          [ -write_ascii ]
          [ -fof_size 10 ]
          [ -verbosity 2 ]
          < .text > .idngram
```

Sintaxis:

```
$ ./text2idngram -vocab /home/carlos/Tesis_LM/Tesis_train.vocab -temp
/home/carlos/Tesis_LM/ -write_ascii </home/carlos/Tesis_LM/
Tesis_train.text> /home/carlos/Tesis_LM/Tesis_train.idngram
```

Y se crea el archivo **Tesis_train.idngram** que muestra cuatro columnas de números enteros.

C.2.4 Ejecutable "idngram2lm"

Parámetros:

```
idngram2lm -idngram .idngram
           -vocab .vocab
           -arpa .arpa | -binary .binlm
         [ -context .ccs ]
         [ -calc_mem | -buffer 100 | -spec_num y ... z ]
         [ -vocab_type 1 ]
         [ -oov_fraction 0.5 ]
         [ -linear | -absolute | -good_turing | -witten_bell ]
```

```
[ -disc_ranges 1 7 7 ]
[ -cutoffs 0 ... 0 ]
[ -min_unicount 0 ]
[ -zeroton_fraction 1.0 ]
[ -ascii_input | -bin_input ]
[ -n 3 ]
[ -verbosity 2 ]
[ -four_byte_counts ]
[ -two_byte_bo_weights
[ -min_bo_weight -3.2 ] [ -max_bo_weight 2.5 ]
[ -out_of_range_bo_weights 10000 ] ]
```

Sintaxis:

```
$ ./idngram2lm {idngram /home/carlos/Tesis_LM/Tesis_train.idngram
-vocab /home/carlos/Tesis_LM/Tesis_train.vocab
-arpa /home/carlos/Tesis_LM/Tesis_train.arpa -ascii_input
```

Y se crea el archivo **Tesis_train.arpa** que es de hecho el modelo de lenguaje deseado.

C.2.5 Ejecutable "evallm"

Respetando el flujo del diagrama C.1 solo falta aplicar el ejecutable "evallm" para evaluar el modelo de lenguaje y calcular su perplejidad (perplexity).

Sintaxis:

```
$ ./evallm -arpa /home/carlos/Tesis_LM/Tesis_train.arpa
```

Después de teclear este comando en la terminal se entra a un modo interactivo en donde el usuario debe elegir uno de dos comandos:

- Comando **perplexity**: Sirve para calcular la perplejidad y la entropía
- Comando **validate**: Sirve para verificar que las probabilidades en el modelo de lenguaje realmente sumen 1.

Parámetros del comando **perplexity**:

```
perplexity -text .text
         [ -probs .fprobs ]
         [ -oovs .oov_file ]
         [ -annotate .annotation_file ]
         [ -backoff_from_unk_inc | -backoff_from_unk_exc ]
         [ -backoff_from_ccs_inc | -backoff_from_ccs_exc ]
         [ -backoff_from_list .fblist ]
         [ -include_unks ]
```

Sintaxis:

```
evallm: perplexity -text /home/carlos/Tesis_LM/Tesis_train.text
```

Lo que arroja los siguientes resultados:

- Perplejidad = 13.8
- Entropía = 3.79 bits

Para salir del modo interactivo del ejecutable "evallm" se teclea:

```
Ctrl + C
```

C.3 Cómo crear un “binary dump file”

Una vez que se ha creado un modelo de lenguaje en un archivo de texto y en formato **ARPA**, ya sea por medio de la LMTOOL (archivo **.lm**) o de la herramienta CMU SLM Toolkit (archivo **Tesis_train.arpa**) es necesario convertirlo a un formato binario para que sea leído por Sphinx a mucho mayor velocidad que si estuviera en versión textual.

En la terminología de Sphinx este archivo se conoce como “Binary Dump File” o simplemente “Dump File”. Para crearlo es necesario descargar un programa llamado “lm3g2dmp” de la siguiente dirección web:

http://www.speech.cs.cmu.edu/sphinx/download/nightly/lm3g2dmp.nightly.tar.gz

Al descomprimir el archivo descargado se crea la carpeta llamada “lm3g2dmp”. En ella es preciso hacer una compilación de archivos para crear el ejecutable “lm3g2dmp” de la siguiente manera:

```
$ cd lm3g2dmp
$ make
```

Una vez creado el ejecutable “lm3g2dmp” se puede utilizar de la siguiente manera para crear el “Dump File”:

```
$ cd lm3g2dmp
$ ./lm3g2dmp /home/carlos/Tesis_LM/Tesis_train.arpa /home/carlos/
Tesis_LM/
```

Estos comandos crearán un archivo llamado “Tesis_train.arpa.DMP” en la carpeta “Tesis_LM”.

Para más información se pueden consultar los pasos 10 y 12 del siguiente tutorial en línea:

http://ronaldramdhan.wordpress.com/2010/03/10/sphinx-decoder-tutorial/

Apéndice D

Tutorial para el software HTK2SPHINX-CONVERTER

HTK2SPHINX-CONVERTER es un software escrito en Python 2.7 que permite utilizar el sistema de reconocimiento de voz HTK[D.1] con los mismos archivos de entrada y casi de la misma manera que se usa el software de reconocimiento de voz CMU-Sphinx 3.

HTK2SPHINX-CONVERTER puede también hacer "reconocimiento en vivo" con ayuda del sistema de reconocimiento de voz Julius.

Las dos diferencias fundamentales entre el HTK2SPHINX-CONVERTER y el CMU-Sphinx3 es que el primero es un reconocedor de voz basado en gramáticas y el segundo utiliza modelo de lenguaje.

D.1 Requerimientos del sistema

Aunque este sistema sólo ha sido probado en ambiente Linux, puede ser adaptado a otros sistemas operativos debido a la versatilidad que ofrece Python. Por lo tanto, lo que es necesario tener instalado para ejecutarlo es lo siguiente:

- Python 2.7. Ver https://www.python.org/downloads/
- Perl 5 o posterior. Ver https://www.perl.org/get.html
- HTK 3.4.1 o posterior. Ver http://htk.eng.cam.ac.uk/
- Julius 4.3.1 o posterior. Ver http://julius.sourceforge.jp/en_index.php

[D.1] Para más información sobre HTK vea (Young et al., 2006)

D.2 Descarga del sistema

Para descargar HTK2SPHINX-CONVERTER es preciso ingresar a la dirección:

http://www.ciempiess.org/

Y elegir la pestaña "DESCARGAS".

HTK2SPHINX-CONVERTER se distribuye libremente bajo una licencia GNU GPL V3.

D.3 Arquitectura del sistema

La carpeta llama "TASK" contiene los siguientes archivos y directorios:

- **Carpeta scripts_py**: Contiene todos los scripts de Python necesarios para manipular todo el sistema.
- **Carpeta wav**: Contiene los archivos de audio para las etapas de entrenamiento y prueba.
- **Carpeta feat**: Contiene los archivos de atributos extraídos directamente de los archivos de audio. Esta carpeta es generada automáticamente por el script "scripts_py/make_feats.py".
- **Carpeta model_parameters**: Contiene los archivos generados por HTK para ejecutar la etapa de entrenamiento del sistema.
- **Carpeta logdir**: Contiene los archivos de registro de mensajes del sistema o archivos "log" que conciernen a algunos procesos de la fase de entrenamiento.
- **Carpeta result**: Contiene un archivo con los resultados del reconocimiento en modo "batch" o de reconocimiento por lotes.
- **Carpeta etc**: Contiene los archivos de entrada necesarios para configurar todo el sistema. Estos archivos son compatibles con el sistema CMU-Sphinx 3 y pueden cambiarse por el usuario para efectuar experimentos con cualquier otro corpus.

D.4 Ejecución el sistema

Al descargar el HTK2SPHINX-CONVERTER ya se tienen los archivos necesarios para llevar a cabo un pequeño experimento de reconocimiento de voz que incluya: entrenamiento, reconocimiento por lotes ("batch decode") y reconocimiento en vivo ("live decode").

Nota: Las siguientes instrucciones son para un sistema Linux. Para ejecutar estas instrucciones en un sistema con Microsoft Windows se deben sustituir las "/" por "\" en las rutas de los scripts y archivos.

D.4.1 Configuración de inicio

En la carpeta "etc" se encuentran los siguientes archivos de configuración:

- htk_decode.cfg
- htk_train.cfg

En cada uno de ellos hay que identificar la linea:

```
<CFG_BASE_DIR> = "/home/carlos/VARIOS_PROGRAMAS/HTK/TASK"
```

Y cambiar el directorio escrito ahí por el actual.

D.4.2 Ejecutar una ronda de entrenamiento

Ejecutar la siguiente secuencia de comandos en la terminal:

```
$ cd TASK
$ python scripts_py/make_feats.py etc/TASK_train.fileids
$ python scripts_py/RunAll.py
```

D.4.3 Ejecutar una ronda de prueba en modo "batch"

Ejecutar la siguiente secuencia de comandos en la terminal:

```
$ cd TASK
$ python scripts_py/make_feats.py etc/TASK_test.fileids
$ python scripts_py/decode/slave.py
```

D.4.4 Ejecutar una ronda de prueba en modo "live"

Conecte el micrófono a su computadora.

Ejecutar la siguiente secuencia de comandos en la terminal:

```
$ cd TASK
$ python scripts_py/livedecode.py
```

Diga oraciones siguiendo la gramática en el archivo "etc/TASK.grammar"

Cancele con las teclas:

```
CTRL + C
```

Nota: Muchos de los archivos que se requieren para esta tarea de reconocimiento en vivo no se generan de forma automática. Por lo tanto, si se quiere hacer reconocimiento en vivo con una gramática diferente se tendría que consultar el siguiente manual en línea:

http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial/data-prep/step-1

En este manual deben identificarse los pasos para generar los archivos:

- scripts_py/live_decode/sample.grammar
- scripts_py/live_decode/sample.voca

Estos archivos contendrán la gramática especificada en el archivo "etc/TASK.grammar".

D.4.5 Como configurar un nuevo experimento

El script scripts_py/new_experiment.py necesario para generar un nuevo folder "TASK" en una ruta distinta y con un nombre distinto **no ha sido implementado aún en esta versión de HTK2SPHINX-CONVERTER**.

Así que se tiene que copiar manualmente el directorio "TASK" a otra ubicación en el sistema para poder ejecutar nuevos experimentos.

Si se desea cambiar el nombre a la carpeta "TASK" se tienen que modificar manualmente los nombres de los archivos en la carpeta etc y editar algunas líneas de los archivos de configuración "htk_train.cfg" y "htk_decode.cfg".

Apéndice E

Tutorial para el software HTK-BENCHMARK

El HTK-BENCHMARK es un software escrito en Python 2.7 que permite utilizar el sistema de reconocimiento de voz HTK[E.1] con los mismos archivos de entrada y casi de la misma manera en que se usa el software de reconocimiento de voz CMU-Sphinx3.

EL HTK-BENCHMARK no implementa "reconocimiento en vivo" como lo hace el HTK2SPHINX-CONVERTER (Ver Apéndice D), sin embargo, al igual que el CMU-Sphinx3 hace reconocimiento con un modelo de lenguaje basado en 3-gramas y en formato ARPA. De hecho, los modelos de lenguaje para el CMU-Sphinx3 y para el HTK-BENCHMARK son compatibles e intercambiables[E.2].

Las únicas dos diferencias que existen entre los archivos de entrada del CMU-Sphinx3 y del HTK-BENCHMARK son que el segundo requiere un diccionario de pronunciación con extensión ".dict.hdecode" que es exactamente igual al diccionario regular de extensión ".dic", sólo que contiene también las entradas del diccionario de relleno (que tiene extensión ".filler").

La otra diferencia es que aunque ambos sistemas utilizan modelos de lenguaje basados en trigramas y en formato ARPA, el HTK-BENCHMARK requiere que este archivo ARPA este comprimido por medio de **gunzip**, mientras que el CMU-Sphinx3 requiere que este en un

[E.1] Para más información sobre HTK vea (Young et al., 2006)

[E.2] El archivo "feat.params" del CMU-Sphinx3 es completamente diferente al del HTK-BENCHMARK, pero en ambos sistemas pueden utilizarse sin problemas con sus valores por defecto, y casi nunca es necesario tocarlos.

formato binario.

E.1 Requerimientos del sistema

El HTK-BENCHMARK fue diseñado para ser utilizado en ambiente Linux. A continuación se muestran los requerimientos para su uso:

- Python 2.7. Ver https://www.python.org/downloads/
- Perl 5 o posterior. Ver https://www.perl.org/get.html
- HTK 3.4.1 o posterior. Ver http://htk.eng.cam.ac.uk/
- HDecode 3.4.1 o posterior. Ver http://htk.eng.cam.ac.uk/extensions/index.shtml
- SoX v14.3.2 o posterior. Ver http://sox.sourceforge.net/
- gzip 1.4 o superior. Ver http://www.gzip.org/
- awk. Ver http://www.gnu.org/software/gawk/manual/gawk.html

E.2 Descarga del sistema

Para descargar el HTK-BENCHMARK es preciso ingresar a la dirección:

http://www.ciempiess.org/

Y elegir la pestaña "DESCARGAS".

El HTK-BENCHMARK se distribuye libremente bajo una licencia GNU GPL V3.

E.3 Arquitectura del sistema

Después de ejecutar tanto la etapa de entrenamiento como la de reconocimiento, la carpeta de proyecto (que por defecto se llama "TASK") contendrá los siguientes archivos y directorios:

- **Carpeta scripts_py**: Contiene los scripts de Python necesarios para manipular todo el sistema.

- **Carpeta wav**: Contiene los archivos de audio para las etapas de entrenamiento y prueba.
- **Carpeta feat**: Contiene los archivos de atributos (vectores de coeficientes MFCC) extraídos directamente de los archivos de audio. Esta carpeta es generada automáticamente por el script "scripts_py/make_feats.py".
- **Carpeta model_parameters**: Contiene los archivos generados por HTK para ejecutar la etapa de entrenamiento del sistema.
- **Carpeta logdir**: Contiene los archivos de registro de mensajes del sistema o archivos "log" que conciernen a algunos procesos de la fase de entrenamiento.
- **Carpeta result**: Contiene un archivo con los resultados del reconocimiento en modo "batch" o de reconocimiento por lotes.
- **Carpeta etc**: Contiene los archivos de entrada necesarios para configurar todo el sistema. Estos archivos son compatibles con el sistema CMU-Sphinx 3 y pueden cambiarse por el usuario para efectuar experimentos con cualquier otro corpus de entrada.

Nota: Los archivos en la carpeta "scripts_py/seed_files" son el núcleo de todo el sistema. Si no se sabe como modificarlos, es mejor no tocarlos, ya que si alguno es modificado incorrectamente, podría provocar la falla del sistema completo.

E.4 Ejecución el sistema

Al bajar el HTK-BENCHMARK ya se tienen los archivos necesarios para llevar a cabo un pequeño experimento de reconocimiento de voz que incluya: entrenamiento y reconocimiento por lotes ("batch decode").

El nombre de proyecto por defecto es "TASK", que también es el nombre por defecto de la carpeta donde se encuentra el HTK-BENCHMARK.

Nota: Las siguientes instrucciones son sólo para sistemas Linux. El HTK-BENCHMARK no fue optimizado para funcionar en ambiente Windows, sin embargo, es posible que pudiera funcionar ahí si se modifica el código fuente.

E.4.1 Archivos de configuración en la carpeta "etc"

Como se mencionó anteriormente, la carpeta "etc" contiene los archivos de configuración para que el usuario pueda controlar el sistema completo. El nombre de casi todos estos archivos depende del nombre del proyecto que por defecto es "TASK". Los archivos son:

- **feat.params**: Especifica las características de los archivos de audio de entrada y las características de los vectores de coeficientes MFCC o vectores de atributos. Este archivo es utilizado por el script "scripts_py/make_feats.py".
- **htk_decode.cfg**: Archivo de configuración de la etapa de reconocimiento.
- **htk_train.cfg**: Archivo de configuración de la etapa de entrenamiento.
- **<PROJECT_NAME>.dic**: Diccionario de pronunciación para la etapa de entrenamiento.
- **<PROJECT_NAME>.dict.hdecode**: Diccionario de pronunciación para la etapa de reconocimiento.
- **<PROJECT_NAME>.filler**: Diccionario de relleno. Este diccionario registra ciertos sonidos en la grabación que no son habla (ej. risas, tosidos, estornudos, etc.).
- **<PROJECT_NAME>.phone**: Lista de fonemas
- **<PROJECT_NAME>_test.fileids**: Archivo de rutas del conjunto de pruebas de reconocimiento.
- **<PROJECT_NAME>_test.transcription**: Archivo de transcription del conjunto de pruebas de reconocimiento.
- **<PROJECT_NAME>.tg.lm.gz**: Modelo de lenguaje en formato ARPA requerido por la herramienta de HTK llamada HDecode. Hay que notar que este archivo es una versión comprimida en zip del archivo <PROJECT_NAME>.tg.lm realizada por el comando **gzip**.
- **<PROJECT_NAME>.tg.lm**: Modelo de lenguaje en formato ARPA sin comprimir, a diferencia de como lo esta el archivo <PROJECT_NAME>.tg.lm.gz, este archivo sólo se incluye para que el usuario pueda consultar el modelo de lenguaje, pero en realidad, el sistema no lo necesita.

- **<PROJECT_NAME>_train.fileids**: Archivo de rutas del conjunto de entrenamiento.
- **<PROJECT_NAME>_train.transcription**: Archivo de transcripción del conjunto de entrenamiento.

E.4.2 Configuración de inicio

En la carpeta "etc" se encuentran los siguientes archivos de configuración:

- htk_decode.cfg
- htk_train.cfg

Estos archivos son sumamente importantes ya que influyen en el comportamiento de todo el sistema.

Lo que hace que estos archivos sean tan importantes es que contienen rutas que el sistema necesita, como la ruta hacia donde se guardan los archivos con coeficientes MFCCs, la ruta hacia el modelo acústico, la ruta hacia la carpeta de archivos de audio de entrada, etc. Estos dos archivos también contienen el nombre del proyecto actual que es muy importante para identificar los archivos en la carpeta "etc".

Tanto el archivo htk_decode.cfg como el archivo htk_train.cfg tienen en común las siguientes variables de configuración:

- **<CFG_DB_NAME>**: Nombre del proyecto.
- **<CFG_BASE_DIR>**: Ruta al directorio actual del proyecto.
- **<CFG_WAVFILES_DIR>**: Ruta a la carpeta "wav".
- **<CFG_FEATFILES_DIR>**: Ruta a la carpeta "feat".

Nota: Siempre que la carpeta del proyecto es movida hacia alguna otra ubicación del sistema operativo, es posible modificar los archivos htk_decode.cfg y htk_train.cfg para que sus rutas apunten hacia esa nueva ubicación por medio del script "scripts_py/setup_paths.py". Sin embargo, si no se desea utilizar este script, las rutas pueden ser modificadas a mano sin mayor problema, sobre todo si deben apuntar a ubicaciones especiales dentro del sistema operativo.

E.4.3 Ejecutar configuración inicial

Lo primero que se tiene que hacer después de que el HTK-BENCHMARK es descargado y descomprimido en la ruta deseada, se tiene que ingresar a la carpeta TASK por medio de la terminal de Linux utilizado el siguiente comando:

```
$ cd HTK-BENCHMARK/TASK
```

Luego, se ejecuta una configuración automática de inicio para que todas rutas del sistema apunten al lugar correcto, que queda determinado por medio de la ruta del proyecto actual. Esto se hace mediante la ejecución del script "scripts_py/setup_paths.py"

```
$ python scripts_py/setup_paths.py
```

En este punto es preciso acalarar que:

1. Todos los scripts deben de ejecutarse desde la carpeta del proyecto, que en este ejemplo tiene el nombre por defecto: "TASK".
2. Se requiere que el proyecto y la carpeta del proyecto tengan el mismo nombre.
3. Si manualmente se mueve la carpeta de proyecto a otra ubicación, es preciso ajustar las rutas en los archivos htk_decode.cfg y htk_train.cfg ya sea manualmente, o por medio del script "scripts_py/setup_paths.py".

E.4.4 Ejecutar un diagnóstico de los archivos en las carpetas "wav" y "etc"

En este paso se verifica que todos los archivos de configuración en la carpeta "etc" y los archivos de audio en la carpeta "wav" existan y puedan ser accesados.

Esto se logra ejecutando los scripts "scripts_py/00.verify/verify_train.py" para verificar el conjunto de entrenamiento y "scripts_py/00.verify/verify_test.py" para verificar el conjunto de pruebas, así:

```
$ python scripts_py/00.verify/verify_train.py
$ python scripts_py/00.verify/verify_test.py
```

Estos scripts avisarán al usuario si hay errores en alguno de los archivos de configuración de la carpeta "etc" o si hay algún problema con los archivos de audio en la carpeta "wav", pero además, estos scripts darán información valiosa sobre los conjuntos de pruebas y de entrenamiento.

Si no hay errores, ya se puede hacer una ronda de entrenamiento.

E.4.5 Ejecutar una ronda de entrenamiento

Primero deben crearse los archivos con vectores de coeficientes MFCC a partir de los archivos de audio en la carpeta "wav", tecleando lo siguiente:

```
$python scripts_py/make_feats.py etc/TASK_train.fileids etc/htk_train.cfg
```

Y después, es posible ejecutar la ronda de entrenamiento por medio del comando:

```
$ python scripts_py/RunAll.py
```

E.4.6 Ejecutar una ronda de reconocimiento en modo "Batch"

Aquí también se necesita crear los archivos con vectores de coeficientes MFCC tecleando:

```
$ python scripts_py/make_feats.py etc/TASK_test.fileids etc/htk_decode.cfg
```

Para ejecutar la ronda de prueba se teclea:

```
$ python scripts_py/decode/slave.py
```

E.4.7 Como generar un nuevo experimento

Si se desea crear un nuevo proyecto, con un nuevo nombre (por ejemplo "TASK2"), en una nueva ruta (por ejemplo "/home/carlos/Escritorio/"), lo único que hay que hacer es utilizar el

script “scripts_py/new_project.py” de la siguiente manera:

```
$ python scripts_py/new_project.py /home/carlos/Escritorio/ TASK2
```

Esto creará un directorio llamado “TASK2” en la ruta “/home/carlos/Escritorio/” con todos los archivos necesarios para ejecutar todos los pasos anteriores.

También hay que notar que el nombre de la mayoría de los archivos en la carpeta “etc” de este nuevo proyecto comenzarán con “TASK2”.

Finalmente, si se desea hacer experimentos con un corpus de entrada diferente, obviamente deberán modificarse manualmente los archivos en la carpeta “etc” y en el directorio “wav” para añadir el nuevo corpus al sistema.

Apéndice F

Reglas de transcripción fonética de Mexbet

1. **[a_j]: Ante consonante palatal; En diptongo decreciente con /i/ ej. “ayEr”, “aIre” - / a . Z e_7 r(/ , / a_7 i . r(e / - [a_j . Z E_7 r(_\] , [a_j_7 i(. r(E] .**
2. **[a_2]: En diptongo decreciente con /u/; Ante /o/; En silaba cerrada trabada por /l/; Ante el fonema consonántico /x/ ej. “Aunque”, “paOla”, “altUra”, “ajEno” - / a_7 u n . k e / , / p a . o_7 . l a / , / a l . t u_7 . r(a / , / a . x e_7 . n o / - [a_2_7 u(N . k_j e] , [p a_2 . o_7 . l a] , [a_2 l_[. t U_7 . r(a] , [a_2 . x e_7 . n o] .**
3. **/ a /: En todos los demás contextos ej. “cAsa” - / k a_7 . s a / - [k a_7 . s a] .**
4. / e /: Libre, en sílaba abierta; En silaba cerrada trabada por /m, n, s, d/ ej. “pErro”, “pensAr” - / p e_7 . r o / , / p e n . s a_7 r(/ - [p E_7 . r O] , [p e n . s a_7 r(_\] .
5. [E]: En sílaba cerrada, trabada por consonante (excepto /m, n, s, d/); En contacto con /r/ o /r(/; En diptongo decreciente con /i/; Ante /x/ (sin condición silábica) ej. “selvAtico”, “mercAdo”, “peinAr”, “lejAno” - / s e l . b a_7 . t i . k o / , / m e r(. k a_7 . d o / , / p e i . n a_7 r(/ , / l e . x a_7 . n o / - [s E l . V a_7 . t i . k o] , [m E r(_\. k a_7 . D o] , [p E i(. n a_7 r(_\] , [l E . x a_7 . n o] .
6. **/ o /: Libre, en sílaba abierta ej. “comEr” - / k o . m e_7 r(/ - [k o . m E_7 r(_\] .**

7. **[O]: En sílaba cerrada, trabada por consonante sin excepción; En contacto con /r/ o /r(/; En diptongo decreciente con /i/; Ante /x/ (sin condición silábica) ej. "montAr", "zOrra", "herOico", "mojAr" - / m o n . t a_7 r(/ , / s o_7 . r a / , / e . r(o_7 i . k o / , / m o . x a_7 r(/ - [m O n_[. t a_7 r(_\] , [s O_7 . r a] , [E . r(O_7 i(. k o] , [m O . x a_7 r(_\] .**
8. / i /: Libre, en sílaba abierta ej. "chiflAr" - / tS i . f l a_7 r(/ - [tS i . f l_0 a_7 r(_\] .
9. [I]: En sílaba cerrada; En contacto con /r/ o /r(/; Ante /x/ (sin condición silábica) ej. "silbAr", "rIco", "quijAda" - / s i l . b a_7 r(/ , / r i_7 . k o / , / k i . x a_7 . d a / - [s I l . V a_7 r(_\] , [r I_7 . k o] , [k_j I . x a_7 . D a] .
10. [j]: En posición inicial de diptongo ej. "diArio" - / d i a_7 . r(i o / - [d j a_7 . r(j o] .
11. [i( ]: En posición final de diptongo ej. "sEis" - / s e_7 i s / - [s E_7 i(s] .
12. **/ u /: Libre, en sílaba abierta ej. "pulIr" - / p u . l i_7 r(/ - [p u . l I_7 r(_\] .**
13. **[U]: En sílaba cerrada; En contacto con /r/ o /r(/; Ante /x/ (sin condición silábica) ej. "funcionAr", "puritAno", "sujEto" - / f u n . s i o . n a_7 r(/ , / p u . r(i . t a_7 . n o / , / s u . x e_7 . t o / - [f U n . s j o . n a_7 r(_\] , [p U . r(I . t a_7 . n o] , [s U . x e_7 . t o] .**
14. **[w]: En posición inicial de diptongo ej. "cuEnto" - / k u e_7 n . t o / , [k w e_7 n_[. t o] .**
15. **[u( ]: En posición final de diptongo ej. "reusAr" - / r e u . s a_7 r(/ - [r E u(. s a_7 r(_\] .**
16. / p /: Inicial de sílaba, mantiene sus rasgos originales ej. "repAso" - / r e . p a_7 . s o / - [r E . p a_7 . s o] .
17. **/ t /: Inicial de sílaba, mantiene sus rasgos originales ej. "tetEra" - / t e . t e_7 . r(a / - [t e . t E_7 . r(a] .**
18. [k_j]: Se palataliza ante vocales anteriores (palatales) ej. "troquelAr" - / t r(o . k e . l a_7 r(/ - [t r(_0 O . k_j e . l a_7 r(_\] .
19. / k /: Inicial de sílaba, mantiene sus rasgos originales ej. "cAza" - / k a_7 . s a / - [k a_7 . s a] .

20. **/ b /: Inicial absoluto; Después de pausa; Después de nasal ej. “vAca”, “sOmbra” - / b a_7 . k a / , / s o_7 m . b r(a / - [b a_7 . k a] , [s O_7 m . b r(a] .**

21. **[V]: Se presenta fricatizado (aproximante) en todos los demás contextos ej. “cabEza” - / k a . b e_7 . s a / - [k a . V e_7 . s a] .**

22. / d /: Inicial absoluto; Después de pausa; Después de nasal y de /l/ ej. “dOna”, “cOnde”, “cElda” - / d o_7 . n a / , / k o_7 n . d e / , / s e_7 l . d a / - [d o_7 . n a] , [k O_7 n_[. d e] , [s E_7 l_[. d a] .

23. [D]: Se presenta fricatizado (aproximante) en todos los demás contextos ej. “ademAs” - / a . d e . m a_7 s / - [a . D e . m a_7 s] .

24. **/ g /: Inicial absoluto; Después de pausa; Después de nasal ej. “gAto”, “angOsto” - / g a_7 . t o / , / a n . g o_7 s . t o / - [g a_7 . t o] , [a N . g O_7 s_[. t o] .**

25. **[G]: Se presenta fricativo (aproximante) en todos los demás contextos ej. “pegAr” - / p e . g a_7 r(/ - [p e . G a_7 r(_\] .**

26. / tS /: Se mantiene bastante estable (principalmente en el centro del país) ej. “mochIla” - / m o . tS i_7 . l a / - [m o . tS i_7 . l a] .

27. **/ f /: Este fonema se mantiene estable ej. “refOrma” - / r e . f o_7 r(. m a / - [r E . f O_7 r(_\. m a] .**

28. [z]: Sonorizado, ante consonantes sonoras ej. “asbEsto” - / a s . b e_7 s . t o / - [a z . V e_7 s_[. t o] .

29. [s_[ ]: Dentalizado, ante consonante dental sorda ej. “asterIsco” - / a s . t e . r(i_7 s . k o / - [a s_[. t E . r(I_7 s . k o] .

30. [z_[ ]: Sonorizado y dentalizado, ante consonante dental sonora ej. “dEsde” - / d e_7 s . d e / - [d e_7 z_[. D e] .

31. / s /: En todos los demás contextos ej. “recEta” - / r e . s e_7 . t a / - [r E . s e_7 . t a] .

32. **/ x /: Este fonema se mantiene bastante estable (sobre todo en el centro del país) ej. “mejOr” - / m e . x o_7 r(/ - [m E . x O_7 r(_\] .**

33. [dZ]: Modifica su modo de articulación a africado en posición inicial absoluta; Después de pausa; Después de nasal y de lateral ej. “yUnque”, “inyectAr”, “ulyAna” - / Z u_7 n . k e / , / i n . Z e k . t a_7 r(/ , / u l . Z a_7 . n a / - [dZ U_7 N . k_j e] , [I n_j . dZ E k . t a_7 r(_\] , [U l_j . dZ a_7 . n a] .

34. / Z /: En todos los demás contextos ej. “payAso” - / p a . Z a_7 . s o / - [p a_j . Z a_7 . s o] .

35. **/ m /: En todos los contextos, principalmente inicial de sílaba ej. “medievAl” - / m e . d i e . b a_7 l / , [m e . D j e . V a_2_7 l] .**

36. **[m_n]: La unión de nasales alveolar y bilabial combina la articulación. No se implementó por que modifica la división silábica ej. “enmEdio” - “ e n . m é . d i o “ - / e n . m e_7 . d i o / - [e . m_n e_7 . D j o] .**

37. [m]: Ante bilabial, se labializa ej. “inviErno” - / i n . b i e_7 r(. n o / - [I m . b j E_7 r(_\. n o] .

38. [M]: Ante labiodental, se labiodentaliza ej. “infOrme” - / i n . f o_7 r(. m e / - [I M . f O_7 r(_\. m e] .

39. [n_[ ]: Ante dental se dentaliza ej. “anteriOr” - / a n . t e . r(i o_7 r(/ - [a n_[. t E . r(j O_7 r(_\] .

40. / n /: Inicio de sílaba, sin condición ej. “tIna” - / t i_7 . n a / - [t i_7 . n a] .

41. [n_j]: Ante palatal se palataliza ej. “inyecciOn” - / i n . Z e k . s i o_7 n / - [I n_j . dZ E k . s j O_7 n] .

42. [N]: Ante velar se velariza ej. “hangAr” - / a n . g a_7 r(/ - [a N . g a_7 r(_\] .

43. **/ n∼ /: En todos los contextos ej. “nIño” - / n i_7 . n∼ o / - [n i_7 . n∼ o] .**

44. [l_[ ]: Dentalizado, ante dental ej. “Alto” - / a_7 l . t o / - [a_2_7 l_[. t o] .

45. [l_j]: Palatalizado, ante palatal ej. “salchIcha” - / s a l . tS i_7 . tS a / - [s a_2 l_j . tS i_7 . tS a] .

46. [l_0]: Ensordecido, después de /p, k, f/ ej. “plAnta” - / p l a_7 n . t a / - [p l_0 a_7 n_[. t a] .

47. / l /: En todos los demás contextos ej. “lOco” - / l o_7 . k o / - [l o_7 . k o] .

48. **[r(_0]: Ensordecido, tras de /p, t, k, f/ ej. “crEma” - / k r(e_7 . m a / - [k r(_0 E_7 . m a] .**

49. **[r(_\]: Se relaja en posición final de sílaba o palabra ej. “mermAr” - / m e r(. m a_7 r(/ - [m E r(_\. m a_7 r(_\] .**

50. **/ r(/: En todos los demás contextos ej. “arEna” - / a . r(e_7 . n a / - [a . r(E_7 . n a] .**

51. / r /: Inicial de palabra y de sílaba; Cuando está precedida por /s, n, l/ ej. “honrAdo” - / o n . r a_7 . d o / - [O n . r a_7 . D o] .

52. **/ S /: En todos los contextos ej. “xicotEncatl” - / S i . k o . t e_7 n . k a . tl / - [S i . k o . t e_7 N . k a_j . tl] .**

53. / tl /: Al final de palabra ej. “popocatEpetl” - / p o . p o . k a . t e_7 . p e . tl / - [p o . p o . k a . t e_7 . p e . tl] .

Apéndice G

Tutorial de la librería *fonetica2*

La librería *fonetica2* es una colección de diversas funciones escritas en Python 2.7 destinadas a transcribir palabras en español tanto fonética como fonológicamente en alfabeto Mexbet.

En esta sección se describen todas las funciones disponibles hasta ahora y se explica cómo incorporarlas a un programa de usuario escrito en Python.

Nota: Se considera que los argumentos de entrada de todas las funciones de la librería *fonetica2* serán palabras en español y que no se ingresarán extranjerismos (ej. "capuccino", "ballet"), abreviaturas (ej. "PRD", "SNI"), palabras inventadas (ej. "osabarne", "decete") o nombres propios (ej. "mayelly", "jeannete"). El uso de estas palabras en cualquiera de las funciones de la librería podría producir resultados erróneos.

G.1 Requerimientos del sistema

Gracias a la versatilidad que ofrece Python, la librería *fonetica2* puede ser utilizada en cualquier sistema operativo. El único software que debe estar instalado previamente en el sistema para poder hacer uso de ella es:

- Python 2.7

G.2 Descarga de la librería *fonetica2*

Para descargar la librería *fonetica2* es preciso ingresar a la dirección:

http://www.ciempiess.org/

Y elegir la pestaña "DESCARGAS".

La librería *fonetica2* se distribuye libremente bajo una licencia GNU GPL V3.

G.3 Contenido de la carpeta "fonetica2_library"

En la carpeta "fonetica2_library" se pueden encontrar seis scripts diferentes:

- **div_sil_Ejemplo.py**: Script para probar la función div_sil()
- **T29_Ejemplo.py**: Script para probar la función T29()
- **T66_Ejemplo.py**: Script para probar la función T66()
- **TT_Ejemplo.py**: Script para probar la función TT()
- **TT_INV_Ejemplo.py**: Script para probar la función TT_INV()
- **vocal_tonica_Ejemplo.py**: Script para probar la función vocal_tonica()

Y finalmente encuentra una carpeta llamada "fonetica2" en donde se encuentran todas las funciones disponibles en la librería y que se describen en el siguiente apartado.

G.4 Descripción de las funciones

Todas las funciones de la librería *fonetica2* están contenidas dentro de la carpeta llamada "fonetica2" que debe tratarse como una sola unidad.

Las funciones disponibles hasta el momento son:

- **vocal_tonica()**: Devuelve la misma palabra de entrada pero con su vocal tónica indicada en mayúscula (ej. cAsa, pErro, gAto, etc.).
- **div_sil()**: Devuelve la palabra de entrada dividida en sílabas.
- **TT()**: "TT" es el acrónimo para "Transformación Textual". Esta función produce transformaciones reversibles en la palabra de entrada que son necesarias para prepararla para una correcta transcripción fonética o fonológica.

- **TT_INV()**: Es la función inversa de la función TT().
- **T29()**: Produce una transcripción fonológica en Mexbet T29 de la palabra de entrada. Esta transcripción presenta también división silábica representada por puntos.
- **T66()**: Produce una transcripción fonológica en Mexbet T66 de la palabra de entrada. Esta transcripción presenta también división silábica representada por puntos.

Nota: Se asume que las palabras de entrada estarán en minúsculas, pero podrían llevar indicada la vocal tónica con una mayúscula. De no llevar la vocal tónica indicada, las funciones T29() y T66() no la deducirían.

G.5 Instrucciones para ejecutar los scripts de prueba

Cualquier script con el sufijo "_Ejemplo.py" debe colocarse al mismo nivel que la carpeta llamada "fonetica2". Una vez hecho esto se puede ejecutar dicho script en la terminal así:

```
$ python <script_name>_Ejemplo.py
```

Este comando producirá algún resultado en la terminal dependiendo del script que se invoque. Por ejemplo, si se desea ejecutar el script llamado "div_sil_Ejemplo.py" se teclea:

```
$ cd fonetica2
$ python div_sil_Ejemplo.py
```

Lo cual produce:

```
pa.la.bra
ac.ción
pe.rro
```

```
ex.ce.len.te
e.xa.men
pe.ña
```

G.6 Instrucciones para incorporar funciones a un código en Python

Si se revisa el código fuente de cualquiera de los scripts de prueba, se verá que contienen líneas como estas:

```
#Modulo para trabajar con el sistema operativo
import sys

#Añadir el path donde esta la carpeta "fonetica2"
sys.path.append(".")

#Modulo creado por mi donde viene la funcion div_sil()
from fonetica2.div_sil import div_sil
```

Y también se verá que contienen líneas en donde se llama a alguna de las funciones de la librería *fonetica2*. Por ejemplo:

```
print div_sil("palabra")
```

Basado en esto, se deduce que es muy fácil incluir funciones de la librería *fonetica2* en un código en Python escrito por un usuario. Sólo hay que copiar estas líneas al código del usuario en cuestión. El único requisito es que la carpeta llamada "fonetica2" este el mismo nivel que el script de este usuario.

Apéndice H

Fonemas y alófonos del español mexicano

H.1 Sistema fonológico del español mexicano en alfabeto Mexbet T29

Consonantes	**Labial**	**Labiodental**	**Dental**	**Alveolar**	**Palatal**	**Velar**
Oclusivo sordo	p		t			k
Oclusivo sonoro	b		d			g
Africado sordo					tS	
Fricativo sordo		f		s	S	x
Fricativo sonoro					Z	
Nasal	m			n	n~	
Lateral				l	tl	
Vibrante				r(r		
Vocales				**Anterior**	**Media**	**Posterior**
Cerrada				i		u
Media				e		o
Abierta					a	
Vocales tónicas				**Anterior**	**Media**	**Posterior**
Cerrada				i_7		u_7
Media				e_7		o_7
Abierta					a_7	

H.2 Alófonos del español mexicano en distribución complementaria en alfabeto Mexbet T66

Consonantes	**Labial**	**Labiodental**	**Dental**	**Alveolar**	**Palatal**	**Velar**
Oclusivos Sordos	p		t		k_j	k
Oclusivos Sonoros	b		d			g
Africados Sordos					tS	
Africados Sonoros					dZ	
Fricativos Sordos		f	s_[	s	S	x
Fricativos Sonoros o Sonorizados	V		D z_[	z	Z	G
Nasales	m	M	n_[	n	n_j n~	N
Laterales Sonoras			l_[	l	l_j tl	
Laterales Ensordecidos				l_0		
Vibrantes				r(r		
Vibrantes Ensordecidos y Debilitados				r(_0 r(_\		
Vocales				**Anteriores**	**Medias**	**Posteriores**
Semiconsonantes				j		w
				i(		u(
Cerradas				i		u
				I		U
Medias				e		o
				E		O
Abiertas				a_j	a	a_2
Vocales tónicas				**Anteriores**	**Medias**	**Posteriores**
Semiconsonantes				j_7		w_7
				i(_7		u(_7
Cerradas				i_7		u_7
				I_7		U_7
Medias				e_7		o_7
				E_7		O_7
Abiertas				a_j_7	a_7	a_2_7

Apéndice I

Equivalencias entre Mexbet, AFI y LaTeX

Descripción	**AFI**	**Mexbet**	**Código Latex del símbolo AFI**
Bilabial oclusivo sordo	p	p	p
Dental oclusivo sordo	t	t	t
Velar oclusivo sordo	k	k	k
Palatalizado oclusivo sordo	k^{j}	k_j	k^{j}
Bilabial oclusivo sonoro	b	b	b
Bilabial fricativo sonoro	$\underset{\sim}{\beta}$	V	\textlowering{\textbeta}
Dental oclusivo sonoro	d	d	d
Dental fricativo sonoro	$\underset{\sim}{\eth}$	D	\textlowering{\textipa{\;D}}
Velar oclusivo sonoro	g	g	g
Velar fricativo sonoro	$\underset{\sim}{\gamma}$	G	\textlowering{\textbabygamma}
Palatal africado sordo	$\widehat{tʃ}$	tS	\textroundcap{t\textesh}
Labiodental fricativo sordo	f	f	f
Alveolar fricativo sordo	s	s	s
Alveolar fricativo sonorizado	$\underset{\smile}{s}$	z	\textsubwedge{s}
Dentalizado fricativo sordo	$\underset{\sqsubset}{s}$	s_[	\textsubbridge{s}
Dentalizado fricativo sonorizado	$\underset{\sqsubset}{\underset{\smile}{s}}$	z_[	\textsubwedge{\hphantom{l}}s \textsubbridge{\hphantom{l}}
Palatal fricativo sordo	ʃ	S	\textesh
Velar fricativo sordo	x	x	x
Palatal fricativo sonoro	ʝ	Z	\textctj
Palatal africado sonoro	$\widehat{dʒ}$	dZ	\textroundcap{d\textyogh}
Bilabial nasal sonoro	m	m	m
Bilabial-Alveolar nasal sonoro	$\overset{n}{m}$	m_n	$\overset{n}{m}$
Alveolar nasal sonoro	n	n	n
Labio-Dentalizado nasal sonoro	ɱ	M	\textltailm

Dentalizado nasal sonoro	n̪	n_[	\textsubbridge{n}
Palatalizado nasal sonoro	nʲ	n_j	n^{j}
Velarizado nasal sonoro	nˠ	N	n\textsuperscript {\textbabygamma}
Palatal nasal sonoro	ɲ	n~	\textltailn
Alveolar lateral sonoro	l	l	l
Dentalizado lateral sonoro	l̪	l_[	\textsubbridge{l}
Palatalizado lateral sonoro	lʲ	l_j	l^{j}
Alveolar lateral ensordecido	l̥	l_0	\textsubring{l}
Palatal lateral sonoro	t͡l	tl	\textroundcap{tl}
Alveolar vibrante simple sonoro	ɾ	r(	\textfishhookr
Alveolar vibrante simple ensordecido	ɾ̥	r(_0	\textsubring{\textfishhookr}
Alveolar vibrante simple relajado	ɹ	r(_\	\textturnr
Alveolar vibrante múltiple	r	r	r
Semiconsonante palatal	j	j	j
Semivocal palatal	i̯	i(	\textsubarch{i}
Vocal cerrada palatal	i	i	i
Vocal semiabierta palatal	i̞	I	\textlowering{i}
Vocal media palatal	e	e	e
Vocal semiabierta palatal	e̞	E	\textlowering{e}
Vocal abierta palatalizada	a⁺	a_j	a^{+}
Vocal central abierta	a	a	a
Vocal abierta velarizada	a̱	a_2	\textsubbar{a}
Vocal media velar	o	o	o
Vocal semiabierta velar	o̞	O	\textlowering{o}
Semivocal velar	u̯	u(	\textsubarch{u}
Vocal cerrada velar	u	u	u
Vocal semiabierta velar	u̞	U	\textlowering{u}
Semiconsonante velar	w	w	w

Nota: Para agregar símbolos del Alfabeto Fonético Internacional (AFI) a un código en LaTeX se necesita utilizar la librería "tipa" de este modo:

\usepackage{tipa}

Apéndice J

Frecuencia de fonemas y alófonos del español mexicano

Fonema	Alófono	Zipf y Rogers 1939	Navarro 1946	Alarcos 1950	Quilis 1981	Rojo 1991	Llisterri y Mariño 1993	Pérez 2003	Cuétara 2004
/p/	[p]	2.92	3.04	2.10	2.77	2.66	2.60	2.58	2.43
/t/	[t]	4.46	4.82	4.60	4.53	4.48	4.63	4.92	4.49
/k/	[k]	3.84	4.23	3.80	3.98	3.98	4.04	3.94	1.05
	[kʲ]	-	-	-	-	-	-	-	2.43
/b/	[b]	3.26	2.54	2.50	2.37	2.66	0.45	1.92	0.44
	[β]	-	-	-	-	-	2.47	-	1.89
/B/		-	-	0.10	0.03	-	-	-	-
/d/	[d]	5.06	5.00	4.00	4.24	4.79	0.76	4.84	1.04
	[ð]	-	-	-	-	-	3.20	-	3.25
/D/		-	-	0.25	0.31	-	-	-	-
/g/	[g]	1.02	1.04	1.00	0.94	0.95	0.11	0.94	0.21
	[ˠ]	-	-	-	-	-	0.79	-	0.91
/G/		-	-	0.25	0.28	-	-	-	-
/ʧ/	[t͡ʃ]	0.30	0.30	0.40	0.37	0.28	0.40	0.32	0.32
/f/	[f]	0.72	0.72	1.00	0.55	0.68	0.51	0.75	0.71
/θ/	[θ]	1.74	2.23	1.70	1.45	1.68	1.53	-	-
/s/	[s]	8.12	8.50	8.00	8.32	7.58	6.95	9.61	5.81
	[s̪]	-	-	-	-	-	1.33	-	1.42
	[s̻]	-	-	-	-	-	-	-	1.19
/x/	[x]	0.58	0.51	0.70	0.57	0.73	0.63	0.74	0.62
/ʒ/	[ʒ]	2.40	0.40	0.40	0.41	0.22	0.19	0.69	0.56
	[d͡ʒ]	-	-	-	-	-	-	-	0.29
/m/	[m]	2.98	2.40	2.50	3.06	3.09	3.63	2.62	3.06
/n/	[n]	5.94	2.94	2.70	2.78	6.99	7.02	7.78	5.02
	[n̪]	-	-	-	-	-	-	-	1.22
	[nˠ]	-	-	-	-	-	0.46	-	0.67
/ɲ/	[ɲ]	0.36	0.36	0.20	0.25	0.19	0.27	0.24	0.24
/N/		-	4.69	3.70	4.86	-	-	-	-
/l/	[l]	5.20	5.46	4.70	4.23	5.08	4.25	5.05	4.81
/ʎ/	[ʎ]	0.60	0.60	0.50	0.38	0.38	0.54	-	-
/ɾ/	[ɾ]	5.90	2.40	2.50	3.26	5.67	4.25	6.19	5.36
/r/	[r]	1.04	0.80	0.60	0.43	0.79	0.40	0.64	0.65
/R/		-	3.51	4.50	1.93	-	-	-	-
/i/	[i]	4.20	4.76	8.60	7.38	7.5	4.29	7.46	4.51
	[j]	-	-	-	-	-	2.60	-	2.67
/e/	[e]	12.20	11.75	12.60	14.67	13.51	13.73	14.13	11.13
	[e̞]	-	-	-	-	-	-	-	4.44
/a/	[a]	14.06	13.00	13.70	12.19	13.40	13.43	12.31	11.89
	[a+]	-	-	-	-	-	-	-	0.97
	[a̟]	-	-	-	-	-	-	-	1.19
/o/	[o]	9.32	8.90	10.30	9.98	9.57	10.37	9.28	7.42
	[o̞]	-	-	-	-	-	-	-	2.47
/u/	[u]	1.76	1.92	2.10	3.33	3.16	1.98	3.05	1.93
	[w]	-	-	-	-	-	1.35	-	1.29

Frecuencia de aparición de los fonemas y de los alófonos del español según varios autores

CUÉTARA PRIEDE, JAVIER. 2004. *Fonética de la ciudad de México. Aportaciones desde las tecnologías del habla*, tesis de maestría inédita, México: UNAM, p. 88.

Referencias

Adda-Decker, M. & Lamel, L. (2005). Do speech recognizers prefer female speakers? *INTERSPEECH*, pages 2205–2208. 4

Andruski, J. E., Blumstein, S. E., & Burton, M. (1994). The effect of subphonetic differences on lexical access. *Cognition*, 52(3):163–187. 42, 43

Anguita, J., Peillon, S., Hernando, J., Bramoullé, A., & Lannion, F. (2004). Word confusability prediction in automatic speech recognition. *INTERSPEECH*. 86, 91

Bahl, L., de Souza, P., Gopalakrishnan, P., Kanevsky, D., & Nahamoo, D. (1990). Constructing groups of acoustically confusable words. *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 85–88. 86, 110

Bahl, L. R., De Gennaro, S. V., Gopalakrishnan, P., & Mercer, R. L. (1993). A fast approximate acoustic match for large vocabulary speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 1(1):59–67. 104

Barnard, E., Davel, M. H., & van Heerden, C. J. (2009). ASR corpus design for resource-scarce languages. *INTERSPEECH*, pages 2847–2850. 62, 78

Bashford, J. A., Warren, R. M., & Lenz, P. W. (2006). Polling the effective neighborhoods of spoken words with the verbal transformation effect. *The Journal of the Acoustical Society of America*, 119(4):EL55–EL59. 91

Basseville, M. (1989). Distance measures for signal processing and pattern recognition. *Signal processing*, 18(4):349–369. 88

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press. 15

Branting, L. K. (2003). A comparative evaluation of name-matching algorithms. *Proceedings of the 9th international conference on Artificial intelligence and law*, pages 224–232. 90, 111

Buchwald, A., Felty, R. A., & Pisoni, D. B. (2008). Neighbors as competitors: Phonological analysis of spoken word recognition errors. *Journal of the Acoustical Society of America*, 123(5):3327–3327. 2, 3, 87

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167. 16

Cabeza-Galan, A. (2000). Fundamentos básicos de las telecomunicaciones. *Servicios de formación de telefónica de España S.A.U.* 11

Casacuberta, F., García, R., Llisterri, J., Nadeu, C., Pardo, J., & Rubio, A. (1992). Desarrollo de corpus para investigación en tecnologías del habla (Albayzin). *Procesamiento del lenguaje natural*, 12. 59

Celdrán, E. M. (1998). Análisis espectrográfico de los sonidos del habla. *Editorial Ariel.* 12

Celdrán, E. M. & Planas, A. M. F. (2007). Manual de fonética española: articulaciones y sonidos de español. *Editorial Ariel.* 35, 36, 37, 38, 39

Chan, A., Gouvea, E., Singh, R., Ravishankar, M., & Rosenfeld, R. (2007). (Third Draft) The Hieroglyphs: Building Speech Applications Using CMU Sphinx and Related Resources . *Carnegie Mellon University.* 28

Chen, J.-Y., Olsen, P. A., & Hershey, J. R. (2007). Word confusability-measuring hidden Markov model similarity. *INTERSPEECH*, pages 2089–2092. 86, 88

Christian, P. (1998). Soundex-can it be improved? *Computers in Genealogy*, 6:215–221. 90

Cristianini, N. & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. *Cambridge University Press.* 16

Cuetara-Priede, J. (2004). Fonética de la ciudad de México. Aportaciones desde las tecnologías del habla . *Universidad Nacional Autónoma de México.* in Spanish. 20, 44, 81, 82, 83, 121

Dalston, R. M. (1975). Acoustic characteristics of English/w, r, l/spoken correctly by young children and adults. *Journal of the Acoustical Society of America*, 57(2):462–469. 4

Davis, S. B. & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366. 86

de Córdoba, R., Guarasa, J. M., Ferreiros, J., Montero, J. M., & Pardo, J. M. (2002). State clustering improvements for continuous HMMs in a Spanish large vocabulary recognition system. *INTERSPEECH*. 45

Deng, L., Lenning, M., & Mermelstein, P. (1989). Use of vowel duration information in a large vocabulary word recognizer. *Journal of the Acoustical Society of America*, 86(2):540–548. 4

Du, J., Liu, P., Jiang, H., Soong, F. K., & Wang, R.-H. (2007). A New Minimum Divergence Approach to Discriminative Training. *Acoustics, Speech and Signal Processing (ICASSP)*, 4:677–680. 88

Eide, E. (2001). Distinctive features for use in an automatic speech recognition system. *INTERSPEECH*, pages 1613–1616. 42

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16. 112

Erler, K. & Freeman, G. H. (1996). An HMM-based speech recognizer using overlapping articulatory features. *Journal of the Acoustical Society of America*, 100(4):2500–2513. 43

Federico, M., Giordani, D., & Coletti, P. (2000). Development and Evaluation of an Italian Broadcast News Corpus. *LREC*. 78

Ferretti, M., Maltese, G., & Scarci, S. (1989). Language model and acoustic model information in probabilistic speech recognition. *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 707–710. 105

Fischer-Jorgensen, E. (1952). The phonetic basis for identification of phonemic elements. *The Journal of the Acoustical Society of America*, 24(6):611–617. 31

Flores, J. D. (2004). Método de Multiplicadores de Lagrange: Una Versión Animada. *University of South Dakota*. 16

Gadd, T. (1990). PHONIX: The algorithm. *Program*, 24(4):363–366. 88

Gales, M. J., Ragni, A., AlDamarki, H., & Gautier, C. (2009). Support vector machines for noise robust ASR. *Automatic Speech Recognition and Understanding (ASRU)*, pages 205–210. 16

Gálvez, C. (2006). Identificación de nombres personales por medio de sistemas de codificación fonética. *Encontros Bibli: Revista Eletrônica de Biblioteconomia e Ciência da Informação*, 2(22):105–116. 111

Garofolo, J. et al. (1993). TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. *Linguistic Data Consortium*. 44

Garofolo, J., Fiscus, J., & Fisher, W. (1997). Design and preparation of the 1996 Hub-4 broadcast news benchmark test corpora. *Proc. 1997 DARPA Speech Recognition Workshop*. 78

Garrido, J. M., Escudero, D., Aguilar, L., Cardeñoso, V., Rodero, E., De-La-Mota, C., González, C., Vivaracho, C., Rustullet, S., Larrea, O., et al. (2013). Glissando: a corpus for multidisciplinary prosodic studies in Spanish and Catalan. *Language resources and evaluation*, 47(4):945–971. 75

Garrido-Galindo, M. I. & Herrera-Camacho, J. A. (2011). Nuevos métodos para el diseño de un corpus de vocablos confusos del español hablado de la Ciudad de México. *Non published material*. 96

Gil Fernández, J. (2007). Fonética para profesores de español: de la teoría a la práctica. *Arco/Libros*, page 614. 3, 87

Goldinger, S. D., Luce, P. A., & Pisoni, D. B. (1989). Priming lexical neighbors of spoken words: Effects of competition and inhibition. *Journal of memory and language*, 28(5):501–518. 2

Goldrick, M., Vaughn, C., & Murphy, A. (2013). The effects of lexical neighbors on stop consonant articulation. *The Journal of the Acoustical Society of America*, 134(2):EL172–EL177. 86

Goldwater, S. & Jurafsky, D. (2010). Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, 52(3):181–200. 3, 87

Gonzales-Cam, C. (2008). Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos. *Biblios*, (32):1–8. 111

González-Sigüenza, B. (2008). BATVOX: sistema automático de reconocimiento de locutor. *Estudios de fonética experimental*, 17:303–316. 11

Gotor, A. (2001). *Precepción del Lenguaje*. Departamento de Psicología Básica. 37

Graff, D. (2002). An overview of Broadcast News corpora. *Speech Communication*, 37(1):15–26. 77

Haas, J., Gallwitz, F., Horndasch, A., Huber, R., & Warnke, V. (2005). Telephone-based speech dialog systems. *Pattern Recognition*, pages 125–132. 42

Hagan, M. T. & Demuth, H. B. (1996). *Neural network Design*. PWS Publishing Company. 15

Hayamizu, S., Itahashi, S., Kobayashi, T., & Takezawa, T. (1993). Design and creation of speech and text corpora of dialogue. *IEICE TRANSACTIONS on Information and Systems*, 76(1):17–22. 77

Hernández-Mena, C. D. & Herrera-Camacho, A. (2012). Diseño y creación de un pequeño corpus oral de habla espontánea en español del centro de México para el desarrollo de sistemas de reconocimiento automático de voz. *ROC&C' 2012*. 81

Hernández-Mena, C. D. & Herrera-Camacho, A. (2015a). Creating a Grammar-Based Speech Recognition Parser for Mexican Spanish Using HTK, Compatible with CMU Sphinx-III System. *International Journal of Electronics and Electrical Engineering*, 3(3):220–224. 29, 46, 136

Hernández-Mena, C. D. & Herrera-Camacho, J. A. (2014). CIEMPIESS: A New Open-Sourced Mexican Spanish Radio Corpus. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 371–375. 44, 48, 84, 120

Hernández-Mena, C. D. & Herrera-Camacho, J. A. (2015b). CIEMPIESS LDC2015S07. *Linguistic Data Consortium*. 135

Hernández-Mena, C. D. & Herrera-Camacho, J. A. (2016). CHM150 LDC2016S04. *Linguistic Data Consortium*. 135

Hernández-Mena, C. D., Martínez-Gómez, N.-N., & Herrera-Camacho, A. (2014). A Set of Phonetic and Phonological Rules for Mexican Spanish Revisited, Updated, Enhanced and Implemented . *Research in Computing Science*, 83:61–71. 82, 136

Hertz, J. A., Krogh, A. S., & Palmer, R. G. (1991). *Introduction to the theory of neural network computation*, volume 1. Westview Press. 15

Hieronymus, J. (1994). ASCII phonetic symbols for the world's languages: Worldbet . *T&T Bell Laboratories*. 43

Huang, X., Acero, A., & Hon, H.-W. (2001). Spoken Language Processing. *Prentice Hall*. 9

Huang, X., Alleva, F., Hon, H. W., Hwang, M. Y., Lee, K. F., & Rosenfeld, R. (1993). The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 7(2):137–148. 28

Huerta, J. M. (2000). Speech recognition in mobile environments. *Carnegie Mellon University*. 65

Huerta, J. M. & Stern, R. M. (1998). Speech recognition from GSM codec parameters. *ICSLP*. 65

Huggins-Daines, D., Kumar, D., Chan, A., Black, W. A., Ravishankar, M., & Rudnicky, A. I. (2006). Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. *Acoustics, Speech and Signal Processing (ICASSP)*, 1:185–188. 28

Jelinek, F. (1997). Statistical methods for speech recognition. *MIT Press*. 9

Jiang, H., Li, X., & Liu, C. (2006). Large margin hidden Markov models for speech recognition. *Audio, Speech and Language Processing, IEEE Transactions on*, 14(5):1584–1595. 88

Jurafsky, D. & Martin, J. H. (2000). Speech and language processing an introduction to natural language processing, computational linguistics, and speech recognition. *Prentice Hall*. 9, 68

Kim, H. S. & Un, C. K. (1988). Improving discriminability among acoustically similar words by modified distance metric. *Electronics Letters*, 24(3):161–163. 2

Kirchhoff, K. (2000). Integrating articulatory features into acoustic models for speech recognition. *Phonus 5*, pages 73–86. 42

Kirschning, I. (2001). Research and Development of Speech Technology & Applications for Mexican Spanish at the Tlatoa Group. *CHI'01 Extended Abstracts on Human Factors in Computing Systems*, pages 49–50. 43, 44

Kondrak, G. & Dorr, B. (2004). Identification of confusable drug names: A new approach and evaluation methodology. *Proceedings of the 20th international conference on Computational Linguistics*, page 952. 111

Lambert, B. L., Lin, S.-J., Chang, K.-Y., & Gandhi, S. K. (1999). Similarity as a risk factor in drug-name confusion errors: the look-alike (orthographic) and sound-alike (phonetic) model. *Medical care*, 37(12):1214–1225. 111

Lander, T. & Metzler, S. (1994). The CSLU Labeling Guide. *CSLU Oregon*. 43

Lane, H. & Tranel, B. (1971). The Lombard sign and the role of hearing in speech. *Journal of Speech, Language, and Hearing Research*, 14(4):677–709. 42

Lane, I. (2001). Multipurpose large vocabulary continuous speech recognition engine julius. *Kyoto University*. 29

Lee, K. F., Hon, H. W., & Reddy, R. (1990). An overview of the SPHINX speech recognition system. *Acoustics, Speech and Signal Processing*, 38(1):35–45. 1, 27

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, 10:707. 90

Llisterri, J. (2004). Las tecnologías del habla para el español. *Seminario Internacional Ciencia, tecnología y lengua española: la terminología científica en español*, pages 123–141. 59, 77

Llisterri, J. (2012). Fonética y Fonología. *Universitat Autònoma de Barcelona*. 3, 87

Llisterri, J. & Mariño, J. B. (1993). Spanish adaptation of SAMPA and automatic phonetic transcription. *Reporte técnico del ESPRIT PROJECT*, 6819. 43

López Escobedo, F. (2011). El análisis de las características dinámicas de la señal de habla como posible marca para la contratación e identificación forense de voz: un estudio para el español de México. *Universitat Pompeu Fabra*. 65

Luce, P. A. (1986). Neighborhoods of Words in the Mental Lexicon. 2

Markel, J. & Gray, A. (1976). *LInear Prediction of Speech*. Springer-Verlag. 12

Martín-Iglesias, D., Bernal-Chaves, J., Peláez-Moreno, C., Gallardo-Antolín, A., & Díaz-de María, F. (2005). A speech recognizer based on multiclass SVMs with HMM-guided segmentation. *Nonlinear Analyses and Algorithms for Speech Processing*, pages 257–266. 16

Martínez Celdrán, E. (1984). Fonética. *Teide*. 31, 32, 33, 34, 37, 42

Mcclelland, J. L. & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive psychology*, 18(1):1–86. 2

Metze, F., Waibel, A., Gutacher, Z., Beyerer, J., Gutacher, E., & HdR, L. L. (2005). Articulatory Features for Conversational Speech Recognition. *Universitat Karlsruhe*. 42

Mills, D. L., Prat, C., Zangl, R., Stager, C. L., Neville, H. J., & Werker, J. F. (2004). Language experience and the organization of brain activity to phonetically similar words: ERP evidence from 14-and 20-month-olds. *Journal of Cognitive Neuroscience*, 16(8):1452–1464. 86

Mohamed, A., Dahl, G. E., & Hinton, G. (2009). Deep belief networks for phone recognition. *NIPS Workshop on deep learning and for speech recognition and related applications*. 16

Morgan, J. (2006). West Point Heroico Spanish Speech LDC2006S37. *Linguistic Data Consortium*. 44

Mousty, P., Radeau, M., Peeremen, R., & Bertelson, P. (1996). The role of neighborhood relative frequency in spoken word recognition. *International Conference on Spoken Language Processing (ICSLP 96)*, 4:2498–2501. 2

Nakatani, L. H. & Dukes, K. D. (1977). Locus of segmental cues for word juncture. *Journal of the Acoustical Society of America*, 62(3):714–719. 4

Norris, D. (1994). Shortlist: A connectionist model of continuous speech recognition. *Cognition*, 52(3):189–234. 2

Odden, D. (1996). What is phonology? *Cambridge University Press*. 35, 39

Olguín-Espinoza, J. M., Mayorga-Ortiz, P., Hidalgo-Silva, H., Vizcarra-Corral, L., & Mendiola-Cárdenas, M. L. (2013). VoCMex: a voice corpus in Mexican Spanish for research in speaker recognition. *International Journal of Speech Technology*, 16(3):295–302. 44

Olsen, P. A. & Hershey, J. R. (2007). Bhattacharyya error and divergence using variational importance sampling. *INTERSPEECH*, pages 46–49. 88

Ostendorf, M., Price, P., & Shattuck-Hufnagel, S. (1995). The Boston University Radio News Corpus. *Electrical, Computer and Systems Engineering Department*. 78

Paul, D. B. & Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. 46

Peereman, R. (1997). Orthographic and phonological neighborhoods in naming: Not all neighbors are equally influential in orthographic space. *Journal of Memory and language*, 37(3):382–410. 86

Pérez-Pavón, E. P. (2006). Construcción de un reconocedor de voz utilizando Sphinx y el corpus DIMEx100 . *Facultad de Ingeniería de la UNAM*. 9

Philips, L. (1990). Hanging on the Metaphone. *Computer Language Magazine*, 7(12):39–44. 88

Philips, L. (2000). The double metaphone search algorithm. *C/C++ users journal*, 18(6):38–43. 89

Pineda, L. A., Castellanos, H., Priede, J. C., Galescu, L., Juarez, J., Llisterri, J., Prez-Pavn, P., & Villaseñor, L. (2009). The Corpus DIMEx100: Transcription and Evaluation . *Language Resources and Evaluation*. 44

Pineda, L. A., Pineda, L. V., Cuétara, J., Castellanos, H., & López, I. (2004). DIMEx100: A New Phonetic and Speech Corpus for Mexican Spanish. *Lecture Notes in Computer Science*, 3315:974–984. 44, 45

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. 30

Printz, H. & Olsen, P. A. (2002). Theory and practice of acoustic confusability. *Computer Speech and Language*, 16(1):131–164. 88

Quilis, A. (1993). Tratado de Fonología y Fonética Españolas. *Ed. Gredos*. 39, 40, 41, 49

Rabiner, L. R. & Juang, B.-H. (1993). Fundamentals of Speech Recognition. *PTR Prentice Hall*. 9, 11, 15

Reddy, A. M. & Rose, R. C. (2008). Towards domain independence in machine aided human translation. *INTERSPEECH*, pages 2358–2361. 90

Reyes-Cortés, J.-A. (2010). Reconocimiento continuo del español hablado en México. *Universidad Nacional Autónoma de México (UNAM)*. 9, 22

Reynolds, D. (2009). Gaussian Mixture Models. *Encyclopedia of Biometrics*, pages 659–663. 13

Riley, M. D. & Roe, D. B. (1998). Confusable word detection in speech recognition. *Google Patents*. US Patent 5,737,723. 86, 110

Ríos Mestre, A. (1993). La información lingüística en la transcripción fonética automática del español. *Procesamiento del lenguaje natural. N. 13 (febrero 1993); pp. 381-387*. 46, 48

Russell, R. & Odell, M. (1918). The Soundex Indexing System. *National Archives and Records Administration*. 88, 89

Salcedo, C. S. (2010). The phonological system of Spanish. *Revista de Lingüística y Lenguas Aplicadas*. 39, 48

Salverda, A. P., Dahan, D., Tanenhaus, M. K., Crosswhite, K., Masharov, M., & McDonough, J. (2007). Effects of prosodically modulated sub-phonetic variation on lexical competition. *Cognition*, 105(2):466–476. 2

Scharenborg, O. (2010). Modeling the use of durational information in human spoken-word recognition. *Journal of the Acoustical Society of America*, 127(6):3758–3770. 4

Schultz, T. & Wand, M. (2010). Modeling Coarticulation in EMG based Continuous Speech Recognition. *Speech Communication*, 52(4):341–353. 42

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. pages 437–440. 16

Seltzer, M. (1999). Sphinx III signal processing front end specification. *CMU Speech Group*. 24, 28

Serridge, B. (2000). ASCII Phonetic Symbols for Mexican Spanish. *Grupo Tlatoa ICT, CENTIA. Universidad de las Américas, Puebla, http://webserver. pue. udlap. mx/˜ sistemas/tlatoa/documentation*. 43

Sierra-Martínez, G. & Rosas-González, A. (2010). Una clasificación de corpus lingüísticos informatizados. *Análisis lingüísticos: enfoque sincrónico, diacrónico e interdisciplinario*. 59, 60, 68

Solera-Ureña, R., Padrell-Sendra, J., Martín-Iglesias, D., Gallardo-Antolín, A., Peláez-Moreno, C., & Díaz-de María, F. (2007). SVMs for automatic speech recognition: a survey. *Progress in nonlinear speech processing*, pages 190–216. 16

Stanier, A. (1990). How accurate is Soundex matching. *Computers in Genealogy*, 3(7):286–288. 90

Stüker, S., Schultz, T., Metze, F., & Waibel, A. (2003). Multilingual articulatory features. *Acoustics, Speech, and Signal Processing*, 1:144–147. 42

Sutton, S., Cole, R. A., De Villiers, J., Schalkwyk, J., Vermeulen, P. J., Macon, M. W., Yan, Y., Kaiser, E. C., Rundle, B., Shobaki, K., et al. (1998). Universal speech tools: the CSLU toolkit. *ICSLP*, 98:3221–3224. 44

Taft, R. (1970). Special Report no. 1. *New York State Identification and Intelligence Systems (NYSIIIS)*. 88

Tomás, N.-T. (1915). Alfabeto fonético. *Revista de Filología Española*, 2:374–376. 43

Tomás, N.-T. (1966). El alfabeto fonético de la Revista de Filología Española. *Anuario de Letras*, 6:5–10. 43

Torruella, J. & Llisterri, J. (1999). Diseño de corpus textuales y orales. *Filología e informática. Nuevas tecnologías en los estudios filológicos*, pages 45–77. 59, 60, 77

Tsiakoulis, P. & Potamianos, A. (2010). On the effect of fundamental frequency on amplitude and frequency modulation patterns in speech resonances. *INTERSPEECH*, pages 649–652. 65

Twaddell, W. F. (1952). Phonemes and allophones in speech analysis. *The Journal of the Acoustical Society of America*, 24(6):607–611. 31

Ukkonen, E. (1992). Approximate string-matching with q-grams and maximal matches. *Theoretical computer science*, 92(1):191–211. 89

Uraga, E. (1999). Modelado Fonético para un Sistema de Reconocimiento de Voz Continua en Español. *Master's thesis. Instituto Tecnológico y de Estudios Superiores de Monterrey-Campus Morelos. México.* 44

Uraga, E. & Gamboa, C. (2004). VOXMEX Speech Database: Design of a Phonetically Balanced Corpus. *LREC.* 44, 73

Uraga, E. & Pineda, L. A. (2000). A set of phonological rules for Mexican Spanish. *México: Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas.* 44

Uraga, E. & Pineda, L. A. (2002). Automatic Generation of Pronunciation Lexicons for Spanish. *CICLing*, 2276:330–338. 44

Varela, A., Cuayáhuitl, H., & Nolazco-Flores, J. A. (2003). Creating a Mexican Spanish Version of the CMU Sphinx-III Speech Recognition System. *Springer*, 2905:251–258. 44

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., & Woelfel, J. (2004). Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems.* 28

Wang, H. M., Chen, B., Kuo, J. W., & Cheng, S. S. (2005). MATBN: A Mandarin Chinese broadcast news corpus. *Int. Journal of Computational Linguistics and Chinese Language Processing*, 10(2):219–236. 78

Wells, J. C. (1997). SAMPA computer readable phonetic alphabet. *Handbook of Standards and Resources for Spoken Language Systems.* 43

Wood, S. (2005). Beginners guide to Praat. *Lund University.* 12

Wu, S. & Manber, U. (1992). Fast text searching: allowing errors. *Communications of the ACM*, 35(10):83–91. 88

Young, S. J., Evermann, G., Gales, M. J. F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., & Woodland, P. (2006). The HTK Book (for HTK Version 3.4). *Cambridge University Engineering Department.* 14, 29, 44, 169, 175

Yu, D. & Deng, L. (2015). Automatic Speech Recognition, a deep learning approach. 16

Zobel, J. & Dart, P. (1996). Phonetic string matching: Lessons from information retrieval. *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 166–172. 89, 90, 99