UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

"CONVERSOR DE VOZ A TEXTO PARA EL IDIOMA QUECHUA USANDO LA HERRAMIENTA DE RECONOCIMIENTO DE VOZ KALDI Y UNA RED NEURONAL PROFUNDA"

Para optar al título profesional de:

INGENIERO INFORMATICO Y DE SISTEMAS

Presentado por:

BR. AIMITUMA SUYO, FRANKLIN

BR. CHURATA URTADO, RUTH MERY

Bajo la asesoría de:

MG. JULIO CESAR CARBAJAL LUNA

Cusco – Perú

2019

DEDICATORIA

Dedico este trabajo a mi madre Saturnina, a mi padre Aurelio que en paz descanse, a mis hermanos por el apoyo incondicional en todo este proceso de formación profesional y a mis sobrinos porque me inspiran a seguir adelante en este mundo maravilloso de la tecnología.

Franklin Aimituma Suyo

Dedico este trabajo a mis padres Felicitas y Antonio, a mi hermano Yulinio y toda mi familia, por su sacrificio, por demostrarme su cariño y apoyo incondicional.

Ruth Mery Churata Urtado

AGRADECIMIENTOS

Agradecemos la propuesta de tesis y la asesoría constante al Mg. Julio Cesar Carbajal que siempre estuvo al tanto de cada detalle durante todo el proceso de desarrollo de esta tesis.

También agradecemos a todas las amistades, familiares y toda las personas quechua hablantes que fueron parte de las sesiones de grabación para desarrollar el conjunto de grabaciones de voz, sin ellos, nada de este trabajo sería posible.

RESUMEN

El conjunto de variaciones en la pronunciación (acentos, velocidad, entonación) que son consecuencia de las variaciones en género, edad y localidad de los locutores, afectan en gran medida en la precisión de un conversor de voz a texto. Es por ello que, en esta tesis se describe la construcción de un conversor de voz a texto de habla continua con un gran vocabulario (*LVCSR-Large Vocabulary continuos Speech Recognition*) e independiente del locutor, para el idioma Quechua en su variación dialéctica Cusco-Qollao, basado en la herramienta Kaldi y la arquitectura de una Red Neuronal Profunda como clasificador de fonemas dentro del modelo acústico, para lo cual fue necesario la construcción del corpus de voz balanceada en género, a partir de grabaciones hechas a frases inmersas en distintos fuentes textuales, llegando a obtener un total de 18 horas de audio en Quechua. De igual forma, se realizó la construcción de los distintos recursos de voz (Diccionario fonético, fonemas y grandes colecciones de texto) necesarios para la construcción del modelo acústico y de lenguaje. Una vez construido todos los recursos de voz, se continua con el proceso de entrenamiento del modelo acústico basado en un modelo de Red Neuronal Profunda y el modelo Oculto de Markov (*Deep Neural Network (DNN)-Hidden Markov Model (HMM)*), del mismo modo, el modelo de lenguaje es basado en un modelo de *3-grams*. Finalmente, una vez concluido el proceso de entrenamiento, se realiza el proceso de prueba o reconocimiento basado en un conjunto de experimentos con el fin de obtener valores óptimos para los parámetros de la arquitectura DNN, es así que se llegó a obtener una precisión de 59.20%, con la tasa de aprendizaje igual a 0.002, numero de nodos internos igual a 512 y el número de capas internas igual a 3 como parte de los parámetros de la arquitectura DNN dentro del modelo acústico, lo cual es bastante aceptable en comparación a investigaciones con una cantidad de recursos de voz similares.

Palabras Claves: Sistema de reconocimiento de habla continúa de gran vocabulario, modelos ocultos de Markov, redes neuronales profundas, Kaldi, Quechua.

ABSTRACT

The set of variations in pronunciation (accents, speed, intonation) that are a consequence of the variations in gender, age and location of the speakers, greatly affect the accuracy of a voice to text converter. That is why, in this thesis, the building of a voice to text converter of continuous speech with a large vocabulary (LVCSR-Large Vocabulary continuous Speech Recognition) and independent of the speaker is described for the Quechua language in its dialectical variation Cusco- Qollao, based on the Kaldi tool and the architecture of a Deep Neural Network as a phoneme classifier within the acoustic model, for which the building of the gender-balanced voice corpus was necessary, from recordings made to phrases immersed in different textual sources, reaching a total of 18 hours of audio in Quechua. Similarly, the construction of the different voice resources (phonetic dictionary, phonemes and large collections of text) necessary for building of the acoustic and language model was carried out. Once all voice resources have been built, the training process of the acoustic model based on a Deep Neural Network and the Hidden Markok Model (Deep Neural Network (DNN) -Hidden Markov Model (HMM)). Similarly, the language model is based on a 3-gram model. Finally, once the training process is completed, the test or recognition process is carried out based on a set of experiments in order to obtain optimal values for the DNN architecture parameters, so that an accuracy of 59.20 was achieved %, with the learning rate equal to 0.002, number of internal nodes equal to 512 and the number of internal layers equal to 3 as part of the DNN architecture parameters within the acoustic model, which is quite acceptable compared to research with a similar amount of voice resources.

Key Words: Large Vocabulary Continuous Speech Recognition, Hidden Markov Models, Deep Neural Networks, Kaldi, Quechua.

LISTA DE ABREVIATURAS

ASR: *Automatic Speech Recognition* (Reconocimiento automático de voz)

CMVN: *Cepstral Mean and Variance Normalization* (Normalización de la Media Cepstral y de Varianza)

DCT: *Discrete Cosine Transformate* (Transformada discreta de coseno)

DFT: *Discrete Fourier Transformate* (Transformada discreta de Fourier)

DNN: *Deep Neural Networks* (Redes Neuronales Profundas)

DTW: *Dynamic Time Warping* (Deformación temporal dinámica)

FFT: *Fast Fourier Transformate* (Transformada rápida de Fourier)

FST: *Finite State Transducer* (Autómata de estado finito)

GMM: *Gaussian Mixture Models* (Modelo Mixto Gaussiano)

GUI: *Graphical User Interface* (Interfaz Gráfica de Usuario)

HMM: *Hidden Markov Model* (Modelos Ocultos de Markov)

IPA: *International Phonetic Alfabet* (Alfabeto Fonético Internacional)

LDA: *Linear Discriminat Analysis* (Análisis discriminativo lineal)

LVCSR: *Large Vocabulary Continuous Speech Recognition* (Reconocimiento de voz continuo y de gran vocabulario)

MFCC: *Mel Frequency Cepstral Coefficient* (Coeficientes Cepstrales en la frecuencia de Mel)

MLLR: *Maximun Linear Likelihood Regretion* (Regresión lineal de máxima verosimilitud)

WACC: *Word Accuracy* (Tasa de precisión por palabra)

OOV: *Out-Of-Vocabulary* (Fuera de vocabulario)

SAMPA: *Speech Assessment Methods Phonetic Alphabet* (Métodos de Evaluación de voz del Alfabeto fonético)

SAT: *Speaker Adaptive Training* (Entrenamiento adaptado al locutor)

SRILM: *SRI Language Modeling* (Modelamiento de lenguaje SRI)

WER: *Word Error Rate* (Tasa de error de palabra)

WFST: *Weighted Finite State Transducer* (Autómata de estado finito ponderado)

INDICE GENERAL

DEDICATORIA ii
AGRADECIMIENTOS iii
RESUMEN iv
LISTA DE ABREVIATURAS vi
INDICE DE FIGURAS x
INDICE DE TABLAS xi
INTRODUCCION xii
CAPÍTULO I 1
ASPECTOS GENERALES 1
1.1. Planteamiento del problema 1
1.1.1. Descripción del problema 1
1.1.2. Formulación del problema 1
1.2. Antecedentes 2
1.3. Objetivos 3
1.3.1. Objetivo general 3
1.3.2. Objetivos específicos 3
1.4. Alcances 3
1.5. Limitaciones 4
1.6. Justificación 4
1.7. Metodología 4
1.8. Cronograma de actividades 6
CAPÍTULO II 7
MARCO TEÓRICO 7
2.1. La voz humana 7
2.1.1. Producción del habla 7
2.1.2. Percepción del habla 8
2.2. Descripción lingüística del quechua 9
2.2.1. Fonología y sonido 9
2.2.2. Esquema fonológico por el modo y punto de articulación 12
2.3. Sistema de reconocimiento automático de voz 15
2.3.1. Arquitectura de un Sistema de reconocimiento de voz 16
2.3.2. Extracción de características 17
2.3.3. El modelo acústico 21
2.3.4. Modelo de lenguaje 24

2.3.5. Fase de entrenamiento 24
2.3.6. Fase de reconocimiento 26
2.3.7. Evaluación 27
2.4. Redes neuronales profundas 28
2.4.1. Capas 29
2.4.2. Funciones de activación 29
2.4.3. Propagación hacia adelante (forward propagation) 30
2.4.4. Propagación hacia atrás (Back propagation) 31
2.5. Herramienta KALDI 31
2.5.1. Arquitectura de la herramienta KALDI 32
2.5.2. Componentes del sistema ASR que usa la herramienta kaldi 33
2.5.3. Estructuras de directorios Kaldi y bibliotecas requeridas 34
2.5.4. Decodificador de Kaldi 36
2.5.5. Programación basada en scripting (Shell) 38
CAPÍTULO III 39
DESARROLLO DEL SISTEMA 39
3.1. Construcción del Corpus de voz 39
3.1.1. Obtención de corpus de texto 40
3.1.2. Obtención de audios de voz 40
3.2. Preparación de la herramienta kaldi 43
3.3. Desarrollo del conversor de voz a texto 43
3.3.1. Preparación de archivos para el modelo acústico 44
3.3.2. Preparación de archivos para el modelo de lenguaje 46
3.3.3. Preparación del directorio de lenguaje 48
3.3.4. Preparación de archivos de configuración 51
3.3.5. Extracción de características 52
3.3.6. Entrenamiento del modelo de lenguaje 53
3.3.7. Entrenamiento y alineamiento del modelo acústico 55
3.4. Proceso de decodificación 58
3.4.1. Decodificación monophone 58
3.4.2. Decodificación triphone 58
3.4.3. Decodificación DNN 59
3.5. Desarrollo de interfaz del conversor de voz a texto 59
CAPITULO IV 61
ANALISIS E INTERPRETACION DE RESULTADOS 61

CONCLUSIONES 66
TRABAJOS A FUTURO 67
BIBLIOGRAFÍA 68
ANEXOS 71

INDICE DE FIGURAS

Figura 1: Metodología de desarrollo sistema ASR. 5
Figura 2: Sistema fonatorio humano. 8
Figura 3: Sistema auditivo humano. 8
Figura 4: Arquitectura de un sistema de reconocimiento de voz. 16
Figura 5: Modelo basado en GMM-HMM. 17
Figura 6: Modelo basado en DNN-HMM. 17
Figura 7: Proceso de extracción de características. 18
Figura 8: Proceso de entramado y enventanado. 18
Figura 9: Banco de filtros en la escala de Mel. 19
Figura 10: Arquitectura del modelo acústico. 22
Figura 11: Modelo oculto de Markov. 23
Figura 12: Ejemplo de ejecución en el modelo 3-grams. 24
Figura 13: Ejecución del algoritmo Viterbi. 27
Figura 14: Arquitectura del modelo DNN-HMM. 29
Figura 15: Arquitectura de una Red Neuronal Profunda. 30
Figura 16: La arquitectura de la herramienta Kaldi. 33
Figura 17: Visión general de la estructura de directorios de Kaldi. 36
Figura 18. Máquina de estados finitos ponderado simple (WFST). 37
Figura 19: Resultado de la construcción del corpus de voz. 43
Figura 20: Interfaz de usuario para el conversor de voz a texto. 60
Figura 21: Tasa de precisión para distintos tamaños de corpus de voz. 61
Figura 22: Tasa de precisión de las distintas configuraciones DNN. 64
Figura 23: Resumen distintos modelos acústicos. 65

INDICE DE TABLAS

Tabla 1: Cronograma de actividades.....6
Tabla 2: Vocales del idioma Quechua. 10
Tabla 3: Consonantes del idioma Quechua.....11
Tabla 4: Características de los locutores.....39
Tabla 5: Características del corpus de voz para el quechua.....42
Tabla 6: Resultados de precisión de los distintos modelos triphone con delta. 62
Tabla 7: Resultados de precisión de las distintas configuraciones en DNN. 63
Tabla 8: Resultado textual conversor de voz a texto. 78

INTRODUCCION

Dentro del área de Reconocimiento Automático de voz (*Automatic Speech Recognition - ASR*) que tiene por objetivo generar texto a partir de la voz humana, se presenta una mayor complejidad al tratar de reconocer el habla humano con todo el conjunto de variaciones que posee en la pronunciación tales como acentos, velocidad y entonación, esto debido a las variaciones en género, edad y localidad de los locutores. Además de ello, la construcción de sistemas de reconocimiento de voz a texto (conversores de voz a texto) necesitan de recursos de voz y de texto en una gran cantidad y son desarrolladas para un idioma en específico, debido a que cada idioma presenta una característica fonética y gramatical única.

En el presente proyecto se implementa un conversor de voz a texto de habla continua con un gran vocabulario e independiente del locutor para el idioma Quechua de la región Cusco, basado en la herramienta de reconocimiento de voz Kaldi y la arquitectura de una Red Neuronal Profunda como clasificador de fonemas dentro del modelo acústico. También se construye un corpus de voz balanceada a partir de la grabación de voz de las distintas personas quechua hablantes con variaciones de género, edad y pronunciación, así mismo se construye todos los recursos de voz (Diccionario fonético, fonemas y grandes colecciones de texto) necesarios para la construcción de los modelos acústico y de lenguaje dentro de un sistema ASR.

Finalmente, se desarrolló satisfactoriamente el conversor de voz a texto de habla continua con un gran vocabulario e independiente del locutor para el idioma Quechua de la región Cusco con una precisión de reconocimiento del habla de 59.20%.

El presente trabajo está organizado en los siguientes capítulos.

Capítulo I: Cubre los aspectos generales del problema.

Capítulo II: Proporciona los fundamentos teóricos.

Capítulo III: Desarrollo del conversor de voz a texto basado en la herramienta Kaldi y la arquitectura de una Red Neuronal Profunda dentro del modelo acústico.

Capítulo IV: Análisis e Interpretación de los distintos resultados obtenidos por parte del conversor de voz a texto.

CAPÍTULO I

ASPECTOS GENERALES

1.1. Planteamiento del problema

1.1.1. Descripción del problema

El habla es la principal forma de comunicación que usan las personas para expresar sus pensamientos y sentimientos unos a otros, dentro de esa comunicación se tiene un proceso de reconocimiento de habla, que es realizada casi automáticamente por el ser humano para reconocer el contenido textual de la otra parte. Por otro lado, si llevamos ese proceso de reconocimiento de habla a las maquinas, esto llega a obtener una gran complejidad al momento de reconocimiento debido al gran conjunto de variaciones fonéticas que existe en un idioma y esto dificulta enormemente la obtención de una precisión cercana al que tiene una persona al momento de una conversación.

El conjunto de variaciones en la pronunciación (acento, velocidad, entonación) como consecuencia de la variación geográfica, género y edad de los locutores afectan drásticamente en la precisión de un sistema de reconocimiento de voz capaz de reconocer el habla continua de gran vocabulario e independiente del locutor. Siendo necesario obtener modelos acústicos y de lenguaje mucho más robustos, capaces de soportar las distintas características acústicas en los locutores.

Por otro lado, la construcción de los modelos acústico y de lenguaje conlleva a la construcción de un gran conjunto de recursos de voz y de texto que se hace difícil cuando se trata de idiomas nativos como es caso del idioma Quechua del imperio Inka, debido a la poca cantidad de fuentes textuales y de audio con las que se cuenta en formato digital, a comparación de los idiomas más habladas del mundo (Ingles, Español) que tienen un montón de audios y textos digitalizados en las distintas plataformas de internet listas para ser procesadas.

1.1.2. Formulación del problema

¿El conversor de voz a texto basado en la herramienta Kaldi y una Red Neuronal Profunda, es capaz de reconocer el habla en el idioma Quechua?

1.2. Antecedentes

Antón, J., 2015, "*Desarrollo de un sistema de reconocimiento de habla natural independiente del locutor*", Universidad Autónoma de Madrid, España.

Conclusión: Diseña un sistema de reconocimiento de habla natural de gran vocabulario (LVCRS) para el idioma español utilizando modelos ocultos de Markov (HMM). Luego procede a mejorarlo aplicando diversas técnicas, como el incremento del número de gaussianas en cada estado dentro del HMM y la optimización del tamaño de ventana de análisis para cada archivo de entrada. Finalmente compara los distintos resultados obtenidos en la fase de prueba junto a otros sistemas ASR similares como de Google y Apple, logrando una eficiencia superior a ellos en algunos dominios.

Comentario: Este trabajo de investigación nos facilita la comprensión de un sistema ASR continua de gran vocabulario e independiente del locutor. Así mismo son de vital importancia las distintas optimizaciones que se realizan tanto en la fase de extracción de características como el modelo acústico junto a una Red Neuronal Profunda para una mejora sustancial en el conversor de voz a texto.

Hernández, C., Meza, I. & Herrera, J. 2017, "*Automatic speech recognizers for Mexican Spanish and its open resources*", Universidad Nacional Autónoma de México, México.

Conclusión: Presentan un corpus de voz llamado CIEMPIESS consistente en 17 horas de grabación y transcritas, la metodología y las herramientas usadas para la construcción y evaluación del diccionario y su transcripción fonética automática comparando con una transcripción manual. También incluyen un modelo de lenguaje la cual fue creada usando texto de los periódicos universitarios que están enfocados en lo académico.

Comentario: Esta investigación nos facilita el modelo para la construcción de un corpus de voz, con todos los recursos de voz necesarios tanto para el modelo acústico como para el modelo de lenguaje. También nos permite saber las diferencias existentes entre una transcripción automática y una transcripción manual dentro del modelo acústico.

Cardenas, R., Zevallos, R., Baquerizo, R. & Camacho, L., 2018, "*Siminchik: A Speech Corpus for Preservation of Southern Quechua*". Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan.

Conclusión: Construyen un corpus de voz para el idioma Quechua, obtenidas de programas radiales de la zona sur del Perú, logrando una cantidad de 97 horas de conversaciones espontaneas. El proceso de transcripción es realizada por personas nativas de acuerdo a una convención en la escritura, finalmente realizan experimentos con un modelo basado en GMM-HMM y explican los retos inherentes en los lenguajes ancestrales.

Comentario: Este trabajo nos muestra la complejidad del idioma quechua en su variación Collao, siendo la característica aglutinante del idioma quechua difícil de representar en el modelo acústico y de lenguaje, además de ello reconoce la gran dificultad en la pronunciación de ellas aun por las personas nativas de la zona sur del Perú dentro del corpus de voz.

Castro, G. & Pinares, A., 2015, "*Sistema de reconocimiento automático de voz (conversor voz a texto) para el Quechua hablado en el Cusco con CMU SPHINX4*", Universidad Nacional de San Antonio Abad del Cusco, Perú.

Conclusión: Desarrollan un conversor de voz a texto dependiente del locutor para el idioma Quechua que funciona sobre la plataforma SPHINX-4 con un corpus de voz relativamente pequeño y resalta las características para considerar un buen corpus de voz, la cantidad de audios y la calidad de ellos, la diversidad de los audios y resalta que es ahí donde se encuentra la gran dificultad en los sistemas ASR. Concluye que la construcción de un modelo de lenguaje, las transcripciones y los demás elementos para un sistema ASR tiene aún mucho que mejorar ya que se necesita una mayor cantidad de frases en el idioma Quechua, para que el modelo de lenguaje sea más acertado.

Comentario: Este trabajo nos proporciona una base para la implementación de un sistema ASR para el idioma quechua de la región Cusco, sin embargo, cuenta con muchas limitaciones, tales como: tamaño del corpus de voz, variabilidad de voces, independencia del locutor y los recursos necesarios para el modelo acústico y de lenguaje. Así como también queda obsoleta el uso del modelo GMM-HMM siendo desplazada hoy en día por el modelo DNN-HMM dentro del modelo acústico.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un conversor de voz a texto para el idioma Quechua basado en la herramienta de reconocimiento de voz Kaldi y la arquitectura de una Red Neuronal Profunda.

1.3.2. Objetivos específicos

- Construir el corpus de voz en el idioma Quechua.
- Construir los recursos de voz (Diccionario fonético, fonema y grandes colecciones de texto) necesarios para el modelo acústico y de lenguaje.
- Determinar una arquitectura de Red Neuronal Profunda óptima en la clasificación de fonemas.
- Desarrollar el conversor de voz a texto para el idioma Quechua utilizando la herramienta Kaldi y el modelo DNN-HMM con una precisión cercana al 60%.

1.4. Alcances

- Se construyó un corpus de voz para el idioma Quechua balanceado en genero a partir de grabaciones hechas a personas quechua hablantes de la región Cusco.
- Se construyó los distintos recursos de voz (Diccionario fonético, fonemas y grandes colecciones textuales) necesarios para el modelo acústico y de lenguaje.
- Se obtuvo la arquitectura de una Red Neuronal Profunda óptima para la cantidad de recursos existente.
- Se desarrolló el conversor de voz a texto basado en la herramienta Kaldi y el modelo DNN-HMM para el modelo acústico.

1.5. Limitaciones

- La edad de las personas que son parte del corpus de voz fluctuará entre los 19 y 40 años.
- Dada la variedad del dialecto quechua en el Perú, el trabajo se basará en la variación dialectal Cusco - Collao.
- El proceso de entrenamiento de los modelos dentro del conversor de voz se realizan en una CPU, más no en una GPU, dado que no se cuenta con la factibilidad económica para adquirirlos.

1.6. Justificación

En los últimos años, dentro de los sistemas de reconocimiento automático de voz, se realiza una búsqueda incesante de nuevos enfoques que sean capaces de lidiar con las distintas características acústicas propias de la variación de género, edad y localidad de los locutores, es así que durante mucho tiempo se hizo uso de un enfoque probabilístico como es el modelo GMM-HMM dentro del modelo acústico, para ser luego reemplazada por un modelo mucho más robusto como es el modelo DNN-HMM.

El nuevo enfoque de las Redes Neuronales Profundas y los Modelos Ocultos de Markov (DNN-HMM) es una de las líneas más novedosas en esta disciplina, que ha obtenido una mejora considerable en comparación al enfoque anterior ampliamente utilizado GMM-HMM dentro del modelo acústico. (Liu, 2017), esto debido que es un modelo con una distribución más compleja pero que generaliza mucho mejor con una cantidad pequeña de parámetros. Parte de esa mejora, dentro de los sistemas de reconocimiento de voz, es el uso del conjunto de herramientas llamado Kaldi que obtiene una mayor precisión en comparación a otras herramientas ampliamente utilizadas como: Sphinx y HTK. (Christian G., Patrick L., Rico P., et al., 2014).

La construcción del modelo DNN-HMM necesita de recursos de voz y texto en un idioma en específico y que estos se han desarrollado durante años para los idiomas más conocidos y hablados del mundo (Inglés, Español). Sin embargo, se ha dejado de lado el desarrollo de este tipo de sistemas para lenguajes nativos como es el caso del idioma Quechua de la región Cusco, a pesar de ser este parte de la riqueza cultural de la región Cusco, y que atrae una gran cantidad de turistas con la intención de conocer más sobre la cultura Inca, las costumbres y vivencias andinas.

1.7. Metodología

Dada la naturaleza del trabajo de investigación se utilizó los métodos de investigación explorativa y aplicativa. Explorativa ya que se hizo el proceso de búsqueda de procedimientos adecuados para la construcción del modelo acústico basado en DNN-HMM y aplicativa porque se utilizaron los conocimientos adquiridos para la creación de un prototipo capaz de demostrar los resultados del modelo acústico basado en DNN-HMM. (Fontelles, 2009).

El mecanismo para la construcción del conversor de voz a texto, se divide en 3 fases y se representan en la Figura 1.

Fase de obtención de recursos de voz

1. La construcción de un sistema ASR inicia con la construcción del corpus de voz (conjunto de grabaciones de voz y las respectivas transcripciones). Así mismo, se realiza la construcción de los recursos de voz tales como: Diccionario fonético, fonemas y grandes colecciones de texto, todas estas necesarias para la construcción de los modelos acústicos y de lenguaje.

Fase de Entrenamiento:

1. Luego se realiza la extracción de características al conjunto de grabaciones de voz (corpus de voz), obteniendo así un conjunto de vectores característicos.
2. A partir del conjunto de vectores característicos de la señal de voz y los distintos recursos de voz, se realiza el entrenamiento del modelo acústico basado en el modelo DNN-HMM y el modelo de lenguaje basado en 3-grams.

Fase de reconocimiento:

1. Dentro de la fase de reconocimiento, las personas realizan la pronunciación de alguna frase que luego es almacenada en forma de señal de voz.
2. A partir de dicha grabación de voz, se realiza el proceso de extracción de características para dicha señal de voz.
3. Finalmente se realiza el proceso de reconocimiento de voz a partir de los modelos acústico y de lenguaje obtenidos previamente en la fase de entrenamiento.

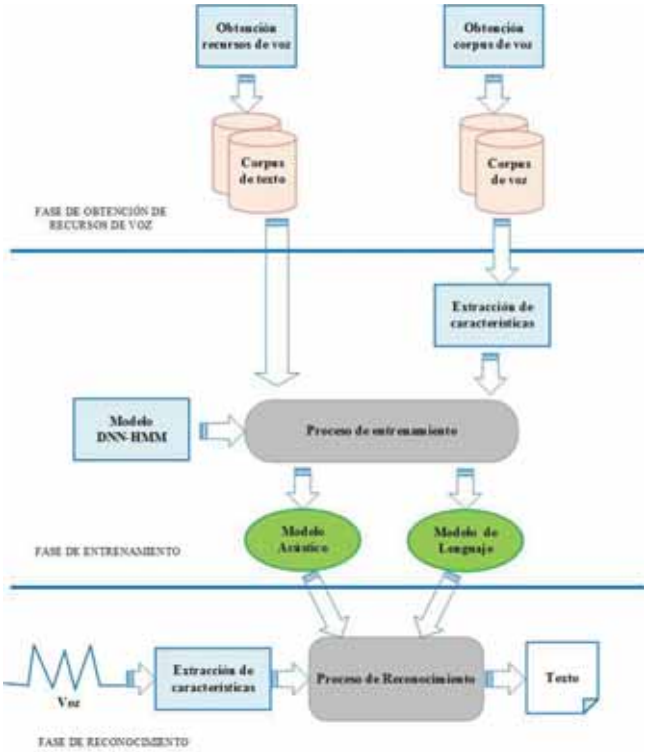


Figura 1: Metodología de desarrollo sistema ASR.

Fuente: Elaboración propia.

1.8. Cronograma de actividades

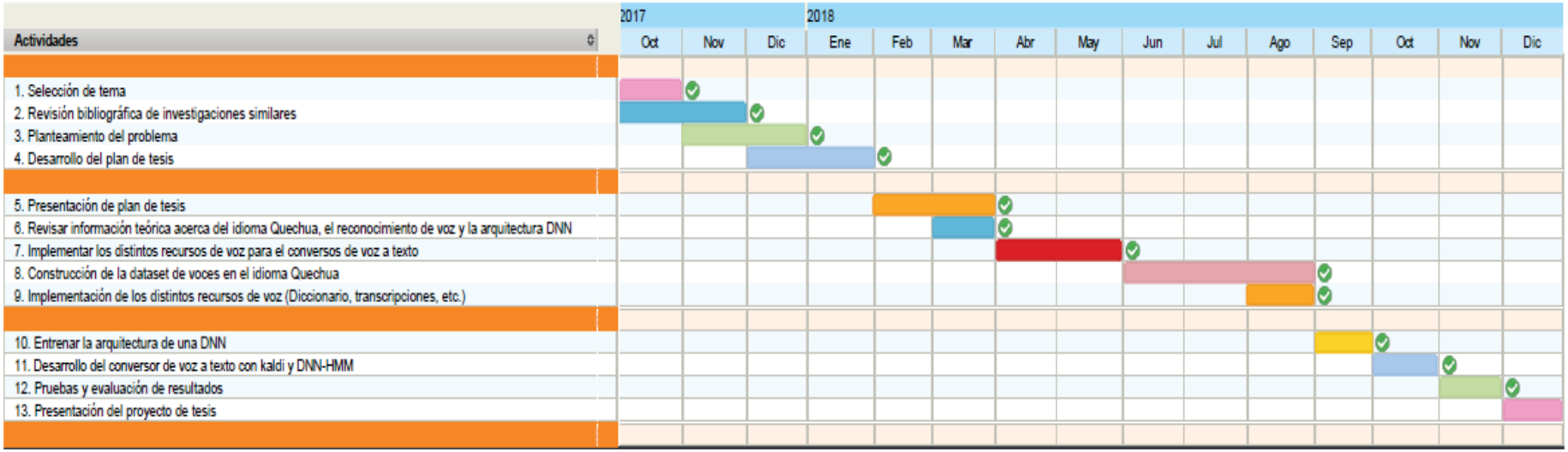


Tabla 1: Cronograma de actividades.

Fuente: Elaboración propia.

CAPÍTULO II

MARCO TEÓRICO

2.1. La voz humana

2.1.1. Producción del habla

La producción de sonidos del habla comienza en el cerebro. Después de la creación del mensaje y la estructura léxico gramatical en nuestra mente, necesitamos una representación de la secuencia de sonido y una serie de comandos que serán ejecutados por nuestros órganos del habla para producir el enunciado.

2.1.1.1. El sistema fonatorio

El sistema fonatorio, está constituido principalmente por la laringe, las cuerdas vocales, los músculos extrínsecos (suprahioideos e infrahioideos), el hueso hioides, el temporal y la mandíbula, los cuales mediante movimientos coordinados, permiten la producción de vibraciones en una frecuencia o tono específico para cada momento de la fonación.

En primer lugar, el aire ingresa a los pulmones a través del mecanismo de inhalación. Como el aire es expulsado de los pulmones con la fuerza muscular a través de los bronquios y la tráquea, este se convierte como la fuente de aire para el proceso fonatorio. Este flujo de aire está en pulsos cuasi periódicos, que luego se modulan en frecuencia al pasar por la cavidad de la garganta, la cavidad bucal y posiblemente la cavidad nasal.

Antes de la llegada del aire a la glotis, las cuerdas vocales se mantienen con la hendidura glótica cerrada (fase pre fonatorio) para evitar el paso del aire. La hendidura glótica es una especie de puerta que regula, mediante la apertura o cierre de las cuerdas vocales, todo lo que entra o sale de los pulmones y el estómago.

Una vez que el aire llega a la glotis, el aire se acumula debajo de las cuerdas generando presión, conocida como presión subglótica. Esta presión se acumula debajo de ellas ocasionando que se separen y permitan la salida del aire. Con la salida del aire, la presión disminuye permitiendo que las cuerdas cierren una vez más la hendidura glótica. Este fenómeno se va produciendo de forma repetida y rápida, produciéndose las vibraciones; como resultado de un continuo cierre y apertura de las cuerdas vocales durante el proceso de exhalación/fonación.

Durante este proceso de apertura y cierre de las cuerdas, que permite o evita el paso del aire, éstas modifican su longitud y tensión, para con ello variar la frecuencia o número de ondas producidas (tono). A mayor longitud y tensión, mayor será la frecuencia o número de ondas por segundo (tonos agudos) y por el contrario, a menor longitud y tensión, menor será también la frecuencia de ondas por segundo producidas (tonos graves). (Torres y Ferran, 2008).

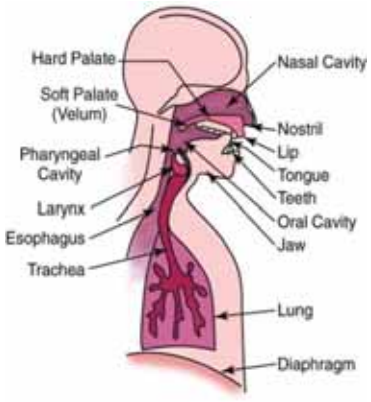


Figura 2: Sistema fonatorio humano.

Fuente: (Hammami, 2014).

2.1.2. Percepción del habla

El habla, como fenómeno acústico, se relaciona y coordina estrechamente con el sistema auditivo, que está especialmente equipado para recibir el contenido que produce la voz humana, dado que el habla es una secuencia de sonidos complejos que varían de continuo en intensidad y frecuencia.

2.1.2.1. El sistema auditivo

El sistema auditivo humano es bilateral. Ambos oídos están ubicados en los huesos temporales, que son de fuerte contextura y están localizados en la base del cráneo. La información acústica llega hasta nuestros oídos, y estos funcionan de manera coordinada con los movimientos de la cabeza.

Las vibraciones aéreas (sonido) son trasmitidas desde el tímpano al estribo a través de los otros huesecillos. La placa inferior del estribo vibra hacia dentro y hacia fuera de la ventana oval. Los cambios de presión en la perilinfa del canal vestibular tienen lugar a causa de dichas vibraciones, los que son también trasmitidos a la endolinfa del conducto coclear a través de la flexible membrana de *Reissner*. Los cambios de presión en el conducto coclear afectan al canal timpánico a través de la membrana basilar, en la que se encuentra el órgano de Corti. (Hammami, 2014).

Figura 3: Sistema auditivo humano.

Fuente: (Hammami, 2014).

El oído humano es un diminuto e ingenioso aparato preparado para recepcionar ondas sonoras y transformarlas en un código neural, cuya interpretación se realiza a nivel del cerebro. Para este fin el oído actúa como amplificador, filtro, atenuador y medidor de frecuencias, al mismo tiempo que funciona como un sistema de comunicación de varios canales.

Dentro de los confines de aproximadamente 16 cm3, nuestros oídos utilizan principios acústicos, mecánicos, electrónicos y de matemática elevada para llevar a cabo lo que hacen. Veamos una de las tantas cosas que pueden hacer los oídos. (Hammami, 2014).

- Los oídos captan desde el más leve susurro hasta el atronador estruendo de un avión de reacción; es decir, una sonoridad diez billones de veces mayor. En términos técnicos, esto es un campo auditivo de unos 130 decibeles (dB).
- Los oídos tienen una enorme capacidad selectiva. En un medio de una multiplicidad de señales acústicas podemos escuchar la voz de una sola persona, o detectar en una orquesta de cien músicos si un instrumento ha emitido una nota equivocada.
- Los oídos son capaces de captar y localizar la posición de la fuente de un sonido con una exactitud aproximada de un grado. Lo hacen percibiendo las minúsculas diferencias que hay en el tiempo de llegada del sonido a cada uno de los oídos y en la intensidad con que llega a ellos. La diferencia de tiempo puede ser de tan sólo diez millonésimas de segundo, pero los oídos pueden detectar y trasmitir ese mensaje al cerebro.
- En un adulto, los oídos tienen la capacidad de reconocer y distinguir unos 400 000 sonidos diferentes, los cuales están ligados al habla, a la música y a los sonidos que producen el hombre y la naturaleza. Los mecanismos del oído analizan automáticamente las ondas sonoras y las compara con las que están acumuladas en la memoria. Así es como se puede saber si cierta nota musical procede de un violín o de una flauta, o quién es la persona que está hablando por teléfono.

2.2. Descripción lingüística del quechua

El alfabeto Quechua utilizado en el presente trabajo está basado en el diccionario de la Academia Mayor de la lengua Quechua (2006), a continuación se muestra el alfabeto y su respectiva clasificación:

A-CH-E-F-H-I-J-K-L-LL-M-N-Ñ-O-P-Q-R-S-T-U-W-Y

La estructura del idioma quechua en los dominios fonéticos y fonológicos es mostrada a continuación.

2.2.1. Fonología y sonido

Su clasificación fonológica, a su vez, es la siguiente:

2.2.1.1. Vocales

Altas: anterior I; posterior U.

Medias: anterior E; posterior O.
Baja: central A.

2.2.1.2. Consonantes

a. **OCLUSIVAS Y AFRICADAS SIMPLES (5):** labial, P; palatal y africada, CH; alveolar, T; velar, K; postvelar, Q.
b. **OCLUSIVAS ASPIRADAS (5**): labial, PH; palatal, CHH; velar, KH; alveolar, TH; postvelar, QH.
c. **OCLUSIVAS GLOTALIZADAS O REFORZADAS (5):** labial, P'; palatal, CH'; alveolar, T'; velar, K'; postvelar, Q'.
d. **FRICATIVAS (3):** alveolar, S; palatal, SH; glotal, H.
e. **LATERALES (2):** alveolar, L; palatal LL.
f. **NASALES (3):** labial, M; alveolar o velar, N; palatal, Ñ.
g. **VIBRANTE SIMPLE (1):** alveolar, R.

El mencionado sistema de escritura, en su inventario fonológico, posee 31 fonemas segmentados, más un signo convencional de glotalización (') y un signo de aspiración (H).

2.2.1.3. Sistema fonético internacional

El número de letras empleados en el alfabeto Quechua depende mucho del dialecto Quechua. Para el presente trabajo se consideraron las vocales como se muestran en la Tabla 2 y las consonantes como se muestran en la Tabla 3.

GRAFEMA	IPA	SAMPA	KALDI	LONGITUD VOCAL	ALTURA VOCAL	FRONTALIDAD VOCAL	ARREDONDAMIENTO DE LOS LABIOS
A	A	A	A	Corta	Baja	Frontal	Sin redondear
E	ɛ	E	E	Corta	Media	Frontal	Sin redondear
I	I	I	I	Corta	Alta	Frontal	Sin redondear
O	ɔ	O	O	Corta	Media	Posterior	Redondeada
U	U	U	U	Larga	Alta	Posterior	Redondeada

Tabla 2: Vocales del idioma Quechua.

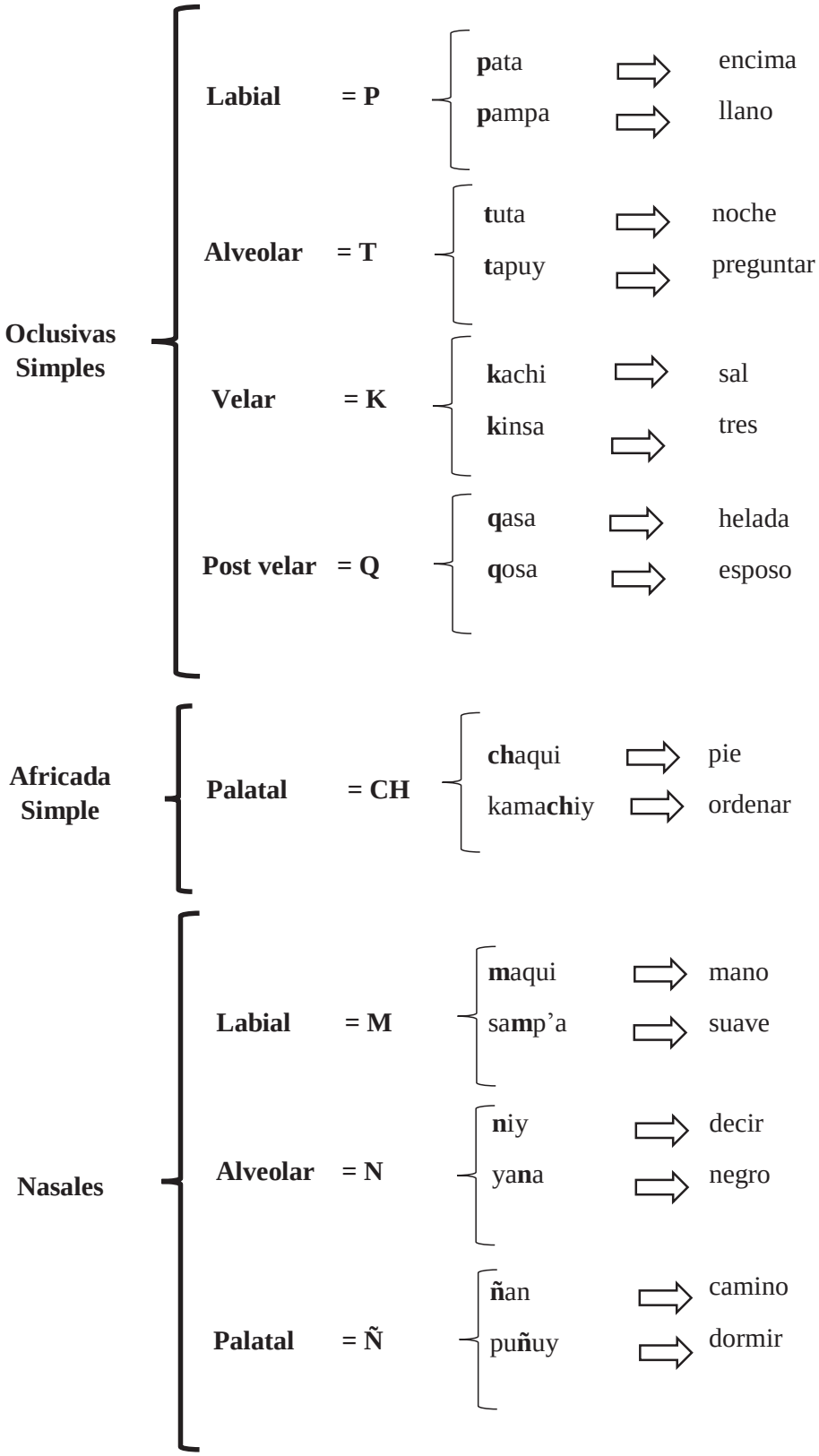
Fuente: Elaboración propia.

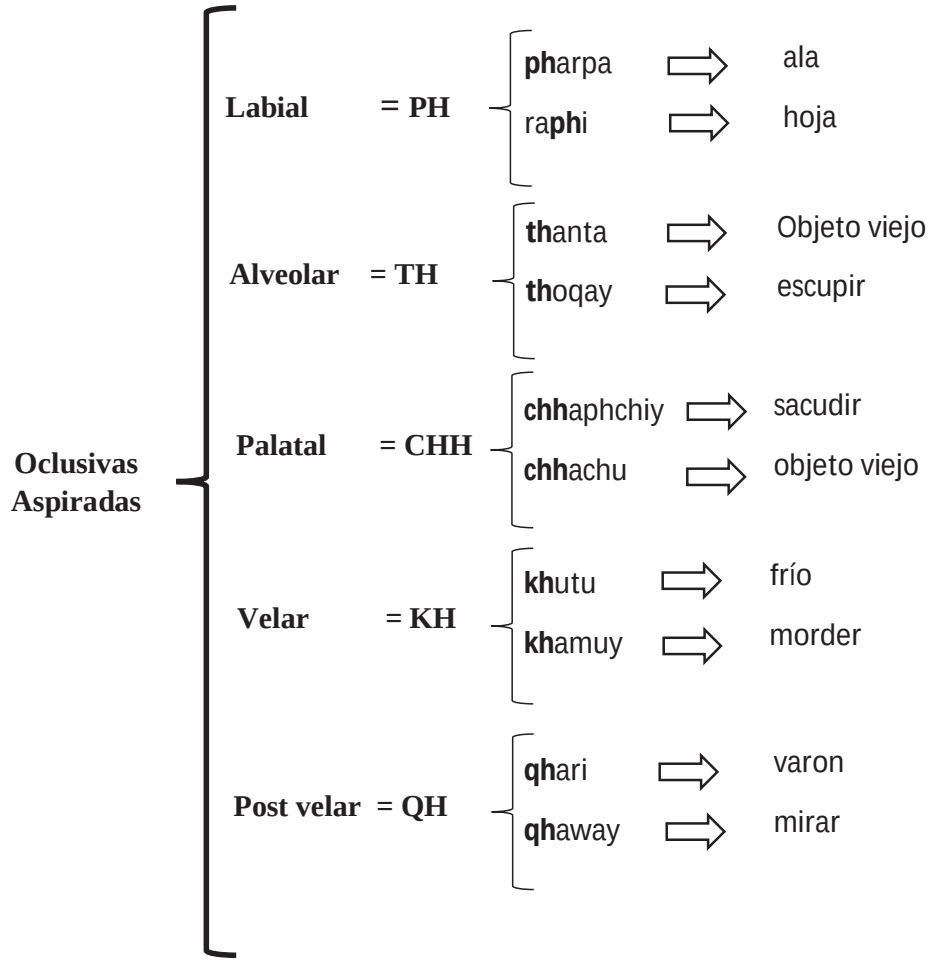
GRAFEMA	IPA	SAMPA	KALDI	TIPO DE CONSONANTES	LUGAR DE ARTICULACIÓN	VOCALIZADAS
CH	tʃ	tS	CH	Africada	Palatal	Sorda
CHH	tʃʰ	tS_h	CHH	Africada	Palatal	Sorda
CH’	tʃˀ	tS_>	CH’	Africada	Palatal	Sorda
H	h	h	H	Fricativa	Postvelar	Sorda
K	k	k	K	Oclusiva Simple	Velar	Sorda
KH	kʰ	k_h	KH	Oclusiva Aspirada	Velar	Sorda
K’	kˀ	k_>	K’	Oclusiva Glotalizada	Velar	Sorda
L	l	l	L	Oclusiva Lateral	Alveolar	Sonora
LL	ʎ	L	LH	Oclusiva Lateral	Palatal	Sonora
M	m	m	M	Oclusiva Nasal	Labial(Bilabial)	Sonora
N	n	n	N	Oclusiva Nasal	Alveolar	Sonora
Ñ	ɲ	J	NH	Oclusiva Nasal	Palatal	Sonora
P	p	p	P	Oclusiva Simple	Labial (Bilabial)	Sorda
PH	pʰ	p_h	PH	Oclusiva Aspirada	Labial (Bilabial)	Sorda
P'	pˀ	p_>	P’	Oclusiva Glotalizada	Labial (Bilabial)	Sorda
Q	q	q	Q	Oclusiva Simple	Postvelar	Sorda
QH	qʰ	q_h	QH	Oclusiva Aspirada	Postvelar	Sorda
Q'	qˀ	q_>	Q’	Oclusiva Glotalizada	Postvelar	Sorda
R	r	r	R	Oclusiva Vibrante	Alveolar	Sonora
S	s	s	S	Oclusiva Fricativa	Alveolar	Sorda
SH	ʃ	S	SH	Fricativa	Palatal	Sorda
T	t	t	T	Oclusiva Simple	Alveolar	Sorda
TH	tʰ	t_h	TH	Oclusiva Aspirada	Alveolar	Sorda
T'	tˀ	t_>	T’	Oclusiva Glotalizada	Alveolar	Sorda
W	w	w	W	Aproximante	Labio-Velar	Sonora
Y	j	j	Y	Aproximante	Palatal	Sonora

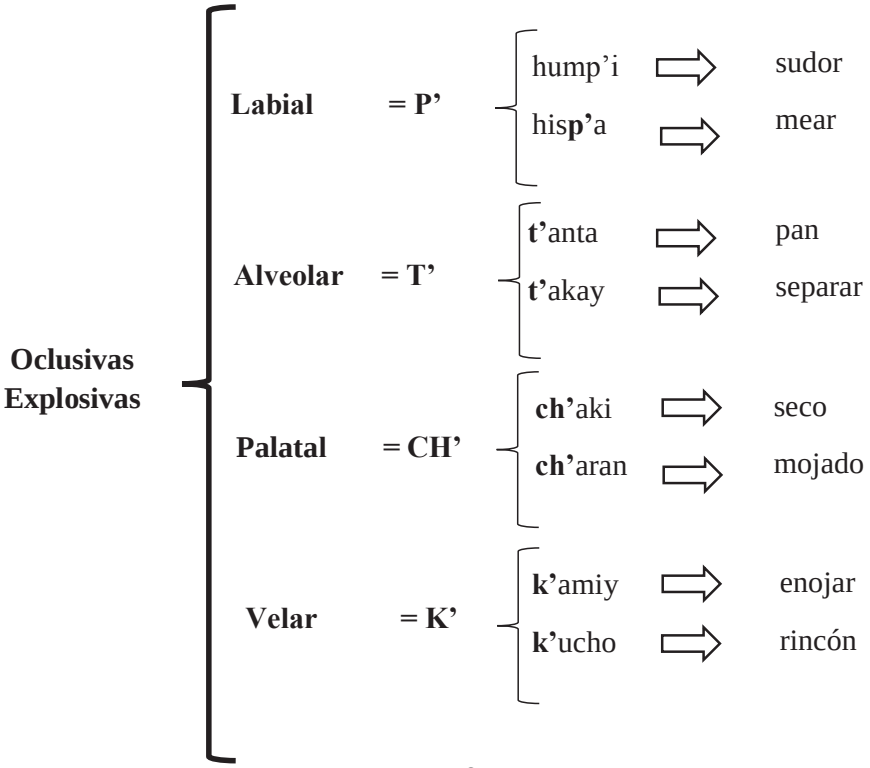
Tabla 3: Consonantes del idioma Quechua

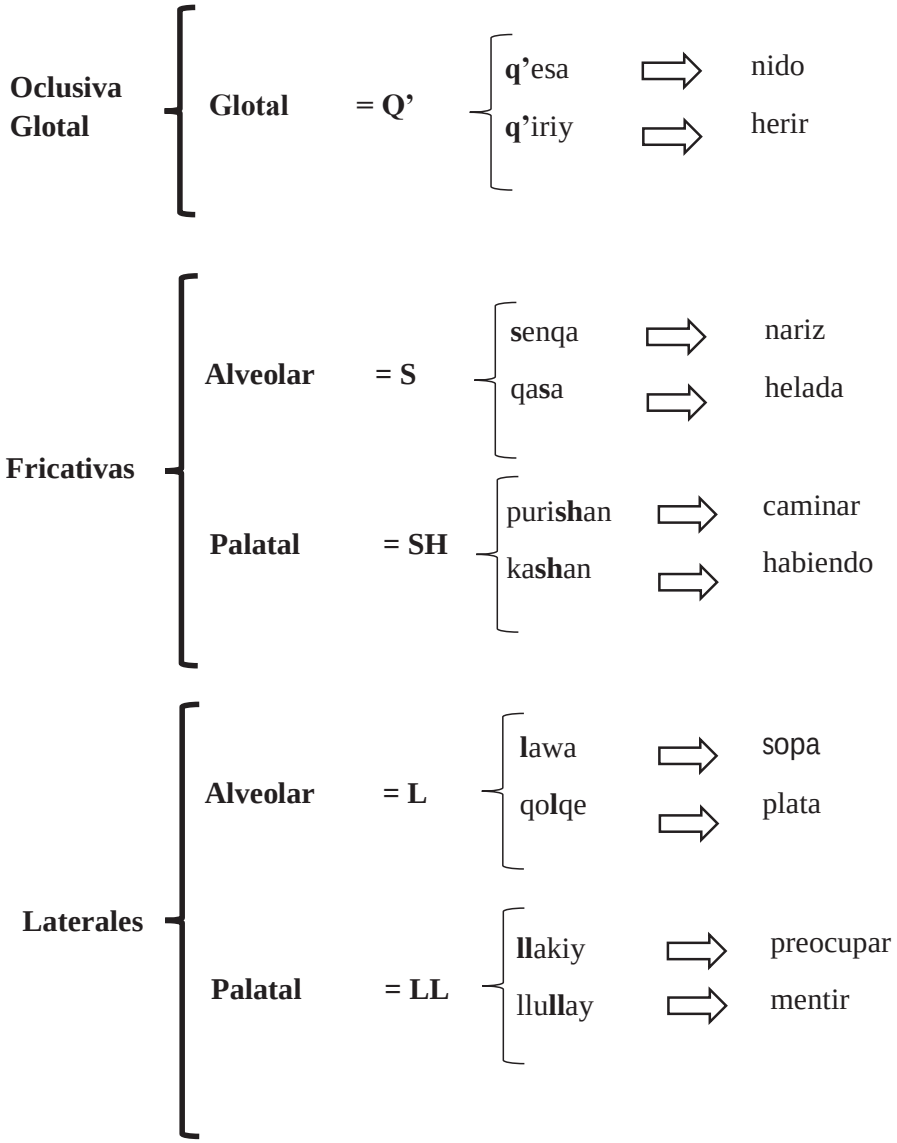
Fuente: Elaboración Propia

2.2.2. Esquema fonológico por el modo y punto de articulación




Oclusivas Aspiradas
Labial = PH
pharpa ala
raphi hoja
Alveolar = TH
thanta Objeto viejo
thoqay escupir
Palatal = CHH
chhaphchiy sacudir
chhachu objeto viejo
Velar = KH
khutu frío
khamuy morder
Post velar = QH
qhari varon
qhaway mirar


Oclusivas Explosivas
Labial = P'
hump'i sudor
hisp'a mear
Alveolar = T'
t'anta pan
t'akay separar
Palatal = CH'
ch'aki seco
ch'aran mojado
Velar = K'
k'amiy enojar
k'ucho rincón


Oclusiva Glotal
Glotal
= Q'
q'esa
nido
q'iriy
herir
Fricativas
Alveolar
= S
senqa
nariz
qasa
helada
Palatal
= SH
purishan
caminar
kashan
habiendo
Laterales
Alveolar
= L
lawa
sopa
qolqe
plata
Palatal
= LL
llakiy
preocupar
llullay
mentir

2.3. Sistema de reconocimiento automático de voz

El reconocimiento automático de voz (sigla en inglés: ASR) es el proceso que realiza una máquina para la identificación del texto correspondiente a partir del habla humano. Dicho de una manera más formal, dado una voz en forma de secuencia de características acústicas X=(x1, x2, x3,..., xT) como entrada al sistema, y la palabra más probable o secuencia de caracteres Y = (y1, y2, y3,..., yT) se trata de encontrar la salida:

$$\bar{Y} = \text{argmax } P(Y|X) \qquad (2.1)$$

Donde P(Y|X) es la distribución condicional que significa la probabilidad de que resulte la palabra Y dada la secuencia de características acústicas X, tomando así la palabra correspondiente al valor más alto obtenido en P(Y|X).

Dado que no se puede obtener de manera directa la palabra deseada mediante la ecuación (2.1) podemos descomponerlo en un producto de dos probabilidades usando la regla de Bayes, definida como:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (2.2)$$

Aplicando (2.2), en la ecuación (2.1) obtenemos:

$$\bar{Y} = \text{argmax } \frac{P(X|Y)P(Y)}{P(X)} \qquad (2.3)$$

La probabilidad mostrada en el denominador (P(X)) siempre tiene una probabilidad igual a cualquier palabra Y procesada, por tanto podemos descartar dicha probabilidad de la ecuación (2.3), por ende P(X) puede ser omitida, quedando así la ecuación final:

$$\bar{Y} = \text{argmax } P(X|Y)P(Y) \qquad (2.4)$$

La secuencia de palabra $\bar{Y}$ determina la maxima probabilidad para esa secuencia de palabra haciendo el producto de las dos probabilidades siguientes:

- P(X/Y) como el modelo acústico el cual captura la probabilidad de una secuencia de observaciones acústicas X dado una secuencia de palabra Y.
- P(Y) como el modelo de lenguaje la cual proporciona una probabilidad para la secuencia de palabra Y.

Un reconocedor estadístico de voz evalúa y combina ambos modelos a través de la generación y calificación de un gran número de secuencia de palabras alternativas (las denominadas hipótesis) durante un proceso de búsqueda compleja.

Los sistemas ASR se pueden clasificar como sigue (Saksamudre, 2015):

- **Basados en el habla**
  - **Habla continua**

    Los sistemas de reconocimiento de voz continua permiten al usuario hablar casi naturalmente, mientras la computadora determina su contenido. Básicamente, es un dictado a la computadora, en ciertos momentos las palabras son pronunciadas juntas sin pausa entre las palabras, este tipo de sistemas son difíciles de desarrollar.

- **Basado en el locutor**
  - **Independiente del locutor**

    Los sistemas independientes del locutor pueden reconocer una variedad de locutores sin un entrenamiento previo. Este tipo de sistemas son desarrollados para operar con cualquier tipo de locutores en particular. Estos son usados en sistemas de interacción por voz que pueden aceptar entradas de un número grande de usuarios diferentes. La implementación de estos sistemas es muy difícil, también es costoso y la eficiencia es más baja que un sistema dependiente del locutor.
- **Basado en el tamaño de vocabulario**
  - **Muy grande**

    Con una cantidad mayor a 10000 palabras

2.3.1. Arquitectura de un Sistema de reconocimiento de voz

Un sistema de reconocimiento automático de voz ha sido abordado desde diferentes enfoques, siendo los probabilísticos, los que han aportado los mejores resultados. (Antón y Tapias, 2015). Un sistema de estas características, tiene la finalidad de extraer la información acústica relevante contenida en la señal de voz, llevar a cabo la decodificación apoyándose en el modelo acústico, el modelo de lenguaje y el diccionario de palabras a ser reconocidas.

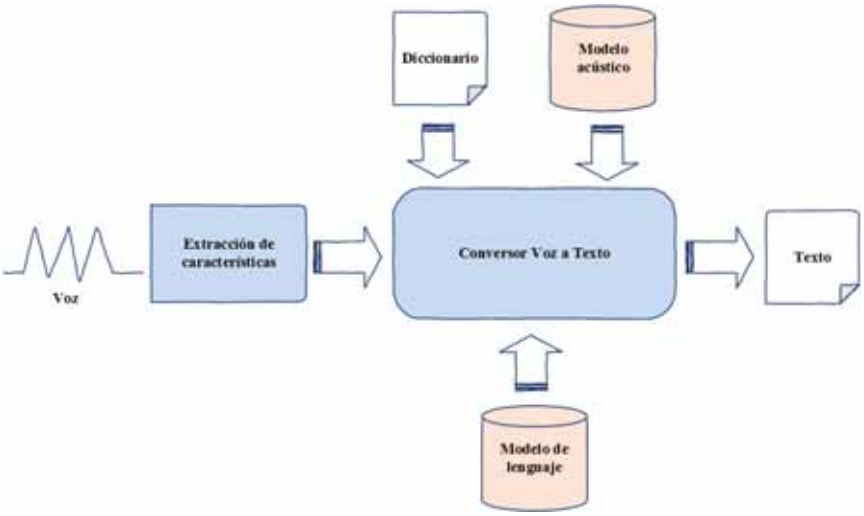


Figura 4: Arquitectura de un sistema de reconocimiento de voz.

Fuente: Elaboración propia.

Hasta el periodo 2009-2012, el estado del arte en sistemas de reconocimiento de voz usaba GMM-HMM dentro del modelo acústico (Fan, Potok, y Shroba, 2017). El HMM modela la secuencia de estados las cuales denotan fonemas (Unidad básica del sonido) y GMM asocia la característica acústica con el estado de HMM.

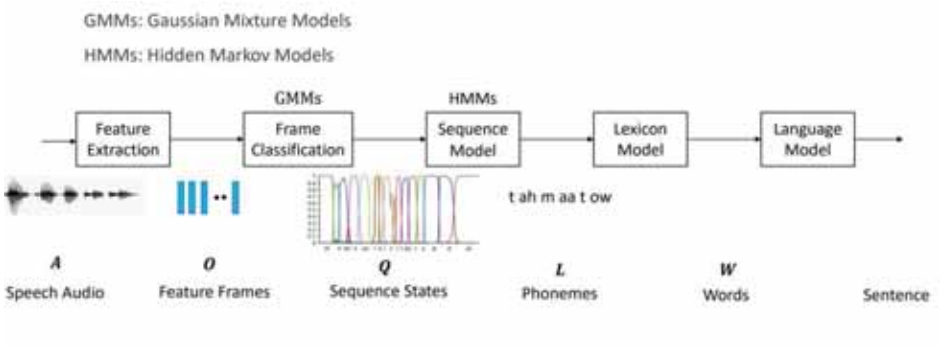


Figura 5: Modelo basado en GMM-HMM.

Fuente: (Fan, Potok, y Shroba, 2017)

Luego con modelos mucho más grandes y computadoras más potentes, el rendimiento de ASR fue drásticamente mejorado por el uso de las Redes Neuronales Profundas. Por ejemplo, el modelo DNN-HMM mejora significativamente la tasa de reconocimiento en la base de datos 'TIMIT', reduciendo la tasa de error en el reconocimiento de fonemas de aproximadamente 26% a 20.7%. (Liu, 2017). Donde 'TIMIT' es una base de datos referente para el reconocimiento de fonemas, jugando un rol similar a la base de datos 'MNIST' usado para el reconocimiento de dígitos escritos a mano.

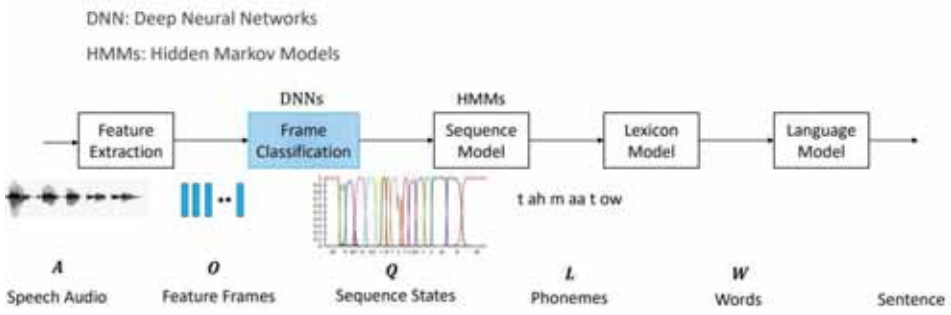


Figura 6: Modelo basado en DNN-HMM.

Fuente: (Fan., Potok, y Shroba, 2017)

2.3.2. Extracción de características

Este módulo se encarga de la extracción de un vector de coeficientes a partir de la señal de voz del locutor, este conjunto pequeño de valores característicos obtenidos a partir del algoritmo de extracción de características mantienen la información relevante y excluye aquellos valores que no sirven o no aportan información relevante (ruido de ambiente, imperfecciones en el micrófono, etc.).

El proceso inicia con la aplicación de la transformada rápida de Fourier (*Fast Fourier Transform - FFT*) a la señal de voz dentro de una ventana de análisis, la cual se desplaza un cierto intervalo de tiempo. Las energías de las frecuencias vecinas dentro de cada trama son desechadas mediante un banco de filtros en la escala Mel, esta última inspirada en el proceso auditivo humano. A la salida de los filtros se aplica un logaritmo y los coeficientes son decorrelados a partir de la

transformada discreta del coseno, dando lugar a un vector de coeficientes cepstrales de frecuencia Mel (*Mel Frecuency Ceptral Coefficiente - MFCC*). (Antón y Tapias, 2015).

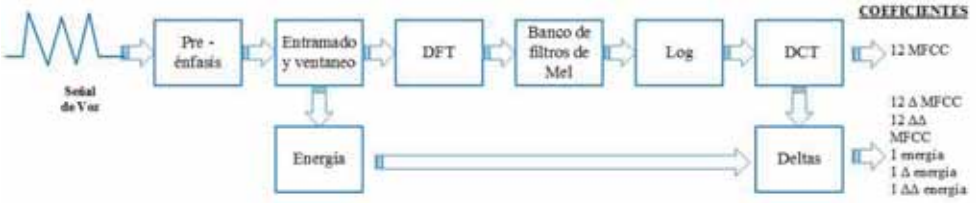


Figura 7: Proceso de extracción de características.

Fuente: Elaboración Propia

2.3.2.1. Filtro de pre-énfasis

Dado que la señal de voz se ve atenuada según se incrementa la frecuencia, se vuelve necesario aumentar la importancia de las frecuencias más elevadas. Por ello, se analiza la señal de voz mediante un filtro de pre-énfasis. Al ser una señal digital se suele aplicar un filtro FIR con función de transferencia $H(z) = 1 - a.z^{-1}$ donde a toma el valor de 0, 97.

El cero de transmisión de este filtro varía según el valor de a, resultando en un filtro plano cuando a = 0 o un cero de transmisión en la frecuencia 0 cuando a = 1. (Antón y Tapias, 2015).

2.3.2.2. Entramado y Enventanado

El siguiente paso es elegir adecuadamente el enventanado y el solapamiento. Dicho enventanado puede ser realizado con una ventana de tipo Hamming. El habla es una señal casi estacionaria, lo que implica que la forma del tracto vocal, y por lo tanto su función de transferencia, permanece casi sin cambios en intervalos de tiempo de 5 a 25 ms de duración es por ello que se elige un enventanado de 25 ms a cada 10ms.

Debido a la forma irregular de la ventana de Hamming, las muestras en los extremos sufrirán una ponderación. Para compensar este efecto lo que se hace es solapar unas ventanas con otras, de forma que se anule. Así, la ventana que se toma es de 25 ms y el desplazamiento de 10 ms, que corresponde a un 60 % de solape entre ventanas. (Antón y Tapias, 2015).

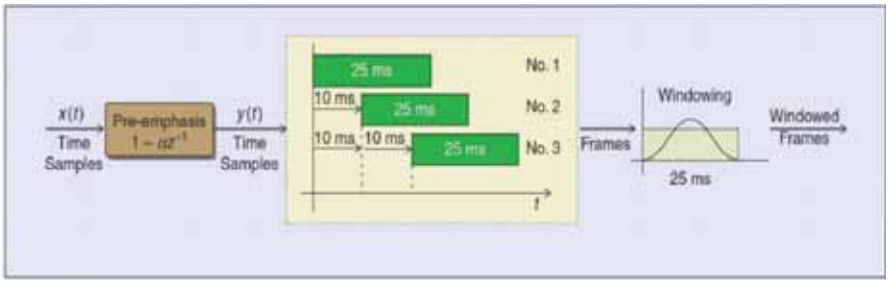


Figura 8: Proceso de entramado y enventanado.

Fuente: (Nacereddine, 2014)

2.3.2.3. Transformada discreta de Fourier

Luego del enventanado necesitamos pasar la señal de voz dada en función del tiempo y la amplitud a un espectro basada en el tiempo y la frecuencia, realizando este proceso para cada trama de 25ms a través de la transformada discreta de Fourier (*Discrete Fourier Transform - DFT*).

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}} \qquad (2.5)$$

Donde se calcula X(k) en el dominio de la frecuencia a partir de la señal en el dominio del tiempo x(n) para cada elemento del conjunto de tramas N y donde k es la frecuencia discreta.

Pero que en la práctica se hace el uso de la transformada rápida de Fourier por el menor costo computacional que ocasiona.

La salida de la transformada de Fourier se filtra con un banco de filtros en la escala de Mel, siendo este proceso de filtrado inspirado en el proceso auditivo humano, puesto que el ser humano percibe mejor un cambio en la voz a bajas frecuencias que en frecuencias altas, así el banco de filtros Mel proporciona una mayor cantidad de filtros para las frecuencias bajas y una menor cantidad para las frecuencias altas.

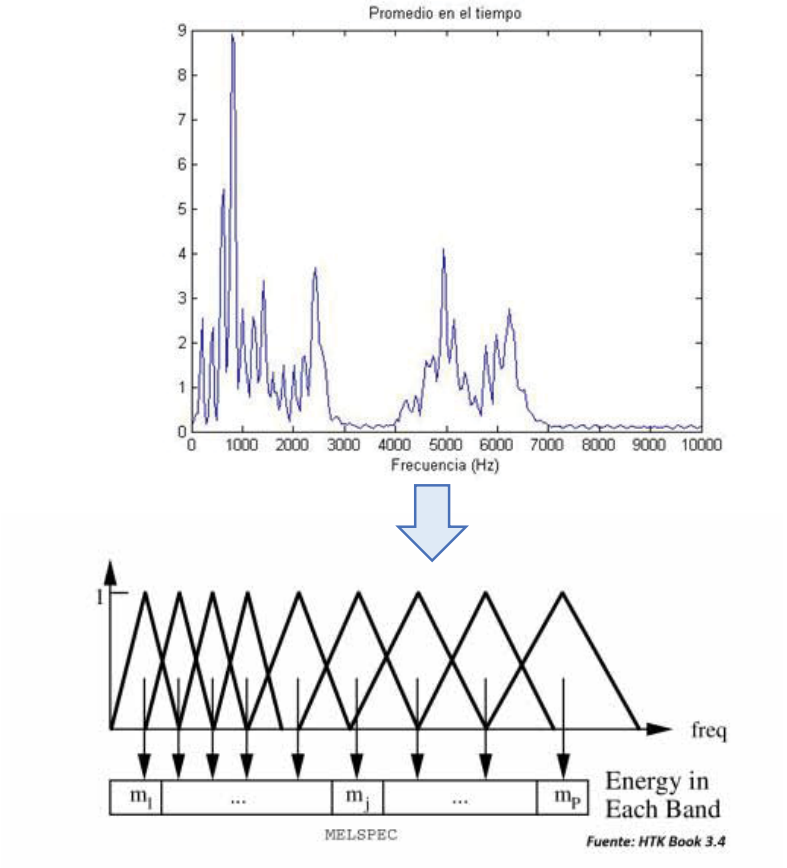


Figura 9: Banco de filtros en la escala de Mel.

Fuente: HTK Book 3.4

2.3.2.4. Transformada discreta del coseno

A la salida de los filtros se aplica un logaritmo y los coeficientes son decorrelados a partir de la transformada discreta del coseno. La transformada discreta del coseno (*Discrete Cosine Transform - DCT*) y la transformada discreta de Fourier son muy similares entre sí, pero su mayor diferencia radica en que mientras las secuencias base de la DFT son exponenciales complejas, en la DCT son funciones cosenoidales. (Antón y Tapias, 2015).

$$X(k) = \sum_{n=0}^{N-1} x(n)\cos[\frac{k\pi}{N}(\mathrm{n} + \frac{1}{2})] \qquad (2.6)$$

Dado que la mayor parte de la energía resultante de este tipo de transformada se aglutina en los coeficientes de baja frecuencia, este tipo de transformada es muy útil en aplicaciones de compresión de datos. (Antón y Tapias, 2015).

2.3.2.5. El dominio cepstral

Todo esto da lugar a un vector usualmente de 12 coeficientes cepstrales de frecuencia Mel (*Mel Frequency Cepstral Coefficiente - MFCC*). Se basan en aplicar un filtro de decorrelación homomórfica (*Cepstrum*) mediante la transformada inversa de Fourier del logaritmo del espectro de potencias, llevando los coeficientes cepstrales al dominio de la frecuencia y obteniendo así coeficientes cepstrales. Gracias a este filtrado se pueden decorrelar los espectros de los filtros en bandas adyacentes, saltando la limitación que tienen las técnicas de análisis espectral que operan en el dominio de la potencia espectral logarítmica.

Así, los coeficientes cepstrales representan la señal en el tiempo, que coincide con el espectro logarítmico de la potencia. Como el dominio de la frecuencia es un dominio homomórfico del dominio temporal, las convoluciones en el dominio temporal se convierten en sumas en el dominio de la *frecuencia*, lo cual permite separar las señales de voz de los ruidos convolucionales con los que están mezcladas.

De esta forma, las partes de excitación y envolvente espectral de la voz aparecerán en zonas distintas en el dominio de la frecuencia, permitiendo separarlo mediante ventanas, lo que se llama *liftering*.

En el dominio de la frecuencia, los análisis en el dominio espectral se llaman LPC (*Linear Predictive Coding*), MFCC (*Mel Frecuency Cepstral Coefficients*) y Cepstrum PLP entre otros. (Antón y Tapias, 2015).

Muy aparte de los 12 coeficientes resultantes de todo el proceso de extracción de características se agregan otros 12 coeficientes delta como la variación de velocidad, también se agregan 12 coeficientes delta-delta como variación de aceleración en el paso de un fonema a otro, finalmente se agregan 3 coeficientes de energía: 1 para los coeficientes simples, 1 para los coeficientes delta y la última para los coeficientes delta-delta.

Resumiendo así los coeficientes obtenidos al final de todo el proceso de extracción de características.

- 12 coeficientes cepstral
- 12 coeficientes delta cepstral
- 12 coeficientes delta - delta cepstral
- 1 coeficiente de energía
- 1 coeficiente de energía delta
- 1 coeficiente de energía delta – delta

2.3.3. El modelo acústico

El modelo acústico es la parte más importante de un sistema de reconocimiento de voz junto al modelo de lenguaje. El modelo acústico calcula la probabilidad P(X|Y) de las características acústicas X dado una palabra Y.

El modelo acústico es obtenido a partir de un entrenamiento previo que toma como entrada un corpus de voz, sus respectivas transcripciones de las voces y el diccionario de palabras con su respectiva transcripción fonética.

En la práctica el modelo acústico no calcula directamente la probabilidad P(X|Y) en su reemplazo calcula $P(X|f_1 f_2 \dots f_n)$ donde la palabra Y es reemplazada por sus respectivos fonemas Y=$f_1 f_2 \dots f_n$, donde "n" es la cantidad de fonemas por el cual está formada la palabra Y.

Los fonemas son modelados mediante tres estados del modelo oculto de Markov, inicialmente esto fue realizado mediante nono-fonemas donde cada estado de los tres estados que tiene cada fonema en el HMM alberga únicamente un fonema, pero la deficiencia de ellos en cuanto al contexto en el cual se pronuncian da inicio al surgimiento de los tri-fonemas que almacenan información del contexto de cada fonema tomando así el fonema anterior, el fonema en sí y el fonema posterior en un solo estado del HMM.

El uso del Modelo Mixto Gaussiano para calcular la probabilidad del vector observado P(O|Q) para cada estado Q de HMM, dependiendo del fonema O, pero que el uso de las Redes Neuronales Profundas incrementa la precisión a la hora de clasificar una característica acústica a un determinado fonema que está ligada con un estado de HMM.

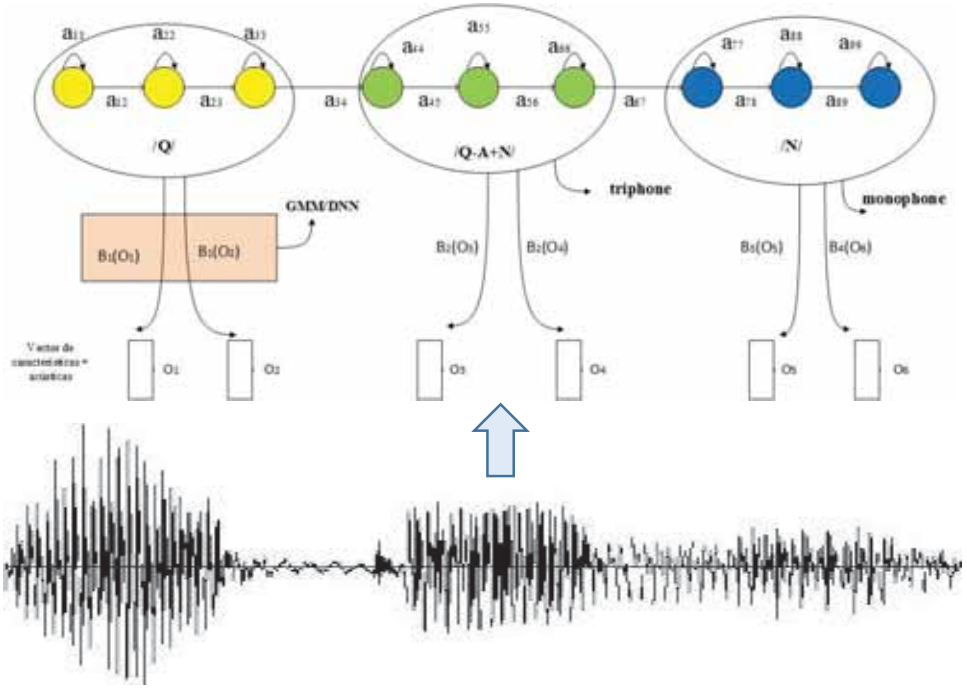


Figura 10: Arquitectura del modelo acústico.

Fuente: Elaboración Propia

2.3.3.1. Modelo oculto de Markov (HMM)

Un modelo oculto de Markov es la representación de un proceso estadístico que se compone de dos elementos: una cadena de Markov de primer orden, con un número finito de estados, y un conjunto de funciones aleatorias asociadas a cada uno de los estados. En un instante concreto de tiempo. El proceso está en un estado determinado y genera una observación mediante la función aleatoria asociada. En el instante siguiente, la cadena de Markov cambia de estado o permanece en el mismo siguiendo su matriz de probabilidades de transición entre estados, generando una nueva observación mediante la función aleatoria correspondiente. El observador externo solamente verá la salida de las funciones asociadas a cada estado, sin observar directamente la secuencia de estados de la cadena de Markov.

Tomando por mayor simplicidad la notación de modelos discretos, un HMM está caracterizado por (Anton y Tapias, 2015):

- El número de estados en el modelo, N. Se denota cada estado como S_i, un estado en el instante t como $q_t = S_i$.
- El número de símbolos observables, M. Se denota a cada símbolo observable como v_j, l observación en el instante t como O_t, y si la observación en el instante t es v_j, se tomara $O_t = v_j$.
- La matriz de probabilidades de transición se define como $A = \{a_{i,j}\}$, siendo $a_{i,j} = P[q_{t+1} = S_j | q_t = S_i]$ y cumpliéndose que 1≤ i,j ≤N.

- La distribución de probabilidad de observación en cada estado j como $B = \{b_j(k)\}$, siendo $b_j(k) = P[v_k(t)|q_t = S_t]$ y para 1 ≤ j ≤ N y 1 ≤ k ≤ M.
- La probabilidad inicial de ocupación de cada estado como $\pi = \{\pi_i\}$, donde $\pi_i = P[q_t = S_i]$ y cumpliéndose que 1 ≤ j ≤ N.

Por convenio, un modelo HMM con todos los parámetros será denotado de la siguiente forma: λ = (A, B, π).

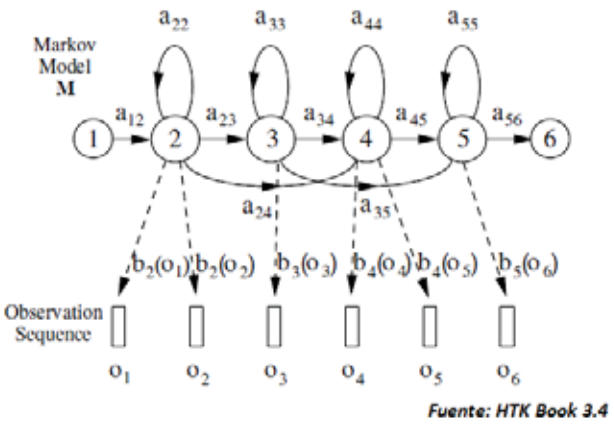


Figura 11: Modelo oculto de Markov.

Fuente: HTK Book 3.4

Cuando hablamos de los Modelos ocultos de Markov, existen 3 problemas inherentes a ello para ser considerados.

1. **El problema de evaluación**

   El problema de evaluación responde a la pregunta: ¿Cuál es la probabilidad que una secuencia de observaciones en particular es producido por un modelo particular?
   Para la evaluación se usa dos algoritmos: el algoritmo *forward* y el algoritmo *backward*.

2. **El problema de decodificación**

   El problema de decodificación responde la pregunta: dada una secuencia de observaciones y un modelo, cual es la secuencia de estados que caracteriza de mejor manera a la secuencia de observaciones.
   Para el problema de decodificación se usa el algoritmo de Viterbi.

3. **El problema de entrenamiento**

   El problema de entrenamiento responde a la pregunta: dado una estructura de modelo y un conjunto de secuencias de observaciones encontrar el modelo que mejor encaja el dato.
   Para este problema se puede usar 3 algoritmos.
   - MLE (*maximum likelihood estimation*)

- Viterbi training (no confundir con la decodificación con Viterbi)
- Baum Welch = Algoritmo *forward-backward*

Resumiendo, el algoritmo de Viterbi se usa para el problema de decodificación y el algoritmo de Baum Welch (*forward-backward*) para el entrenamiento del modelo para un conjunto de secuencia de observaciones.

2.3.4. Modelo de lenguaje

Un modelo de lenguaje complementa a la probabilidad de las características acústicas dado una palabra P(X|Y) que describe al modelo acústico, y esta representa la probabilidad de la palabra transcrita P(Y) como se puede ver en la ecuación (2.7). El modelo de lenguaje a través del modelo n-gram calcula la probabilidad de la palabra *k* de una secuencia de palabras *Y*. La probabilidad es obtenida basándose en las *n-1* palabras anteriores recientes, donde n es llamado el orden del modelo de lenguaje.

$$P(Y) = P(w_k, w_{k-1}, w_{k-2}, \dots, w_1) \approx \prod_{i=1}^{k} P(w_i | w_{i-n+1}^{i-1}) \qquad (2.7)$$

Las probabilidades para cada palabra Y_i son estimadas segun al valor de n-gram, y en la práctica, como es en el proceso de entrenamiento, este se realiza incluyendo los órdenes inferiores a n, descendiendo en una unidad, teniendo así n-1 gram, n-2 gram,..., 1-gram.

La eficiencia en la predicción del modelo de lenguaje puede ser mejorado al incrementar el orden del modelo n-gram, pero siempre y cuando no se exagere en el valor dado que incrementa mayor complejidad.

El modelo de lenguaje calcula la probabilidad de una palabra haciendo el conteo de las veces que aparece solo o posterior a otras palabras en el corpus de texto que bien podría ser el conjunto de transcripciones de texto realizados a las grabaciones de voz.

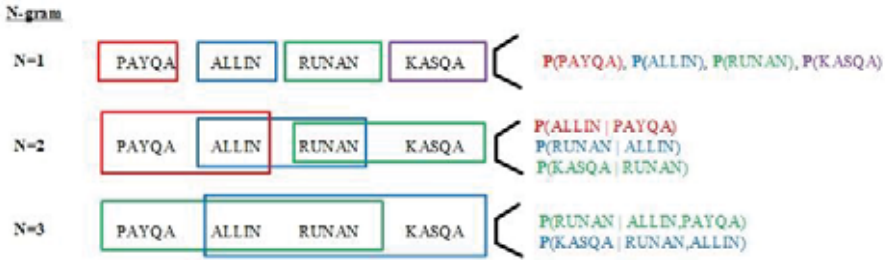


Figura 12: Ejemplo de ejecución en el modelo 3-grams.

Fuente: Elaboración propia

2.3.5. Fase de entrenamiento

Los parámetros del HMM tales como la matriz de probabilidad de transición entre estados $a_{i,j}$ así como las probabilidades de emisión de ciertas observaciones

$b_j(O_t)$son entrenados en la fase de entrenamiento del modelo mediante el algoritmo Baum Welch (forward-backward)**.** (*Calvo, 2010*).

2.3.5.1. Algoritmo de Baum Welch

Para la estimación de los parámetros del modelo $\delta = (A, B, \pi)$ que maximizan la probabilidad de observación P (O|δ), se utilizará el algoritmo de *Baum-Welch*, que consiste en un caso particular del algoritmo de *Expectation - Maximization* (EM) aplicado a los HMMs. Con esto se pretende realizar una serie de iteraciones y bajo el criterio de máxima verosimilitud (*Maximum Likelihood*), ir encontrando máximos locales de la probabilidad de observación.

En primer lugar, se comenzará por definir una función auxiliar, que depende de los parámetros anteriores del modelo δ y de la nueva estimación de ellos $\bar{\delta}$:

$$Q\left(\delta\middle|\bar{\delta}\right) = \sum_Q P(Q|\mathrm{O}, \delta) log\left[P(O, Q|\bar{\delta})\right] \qquad (2.8)$$

Según el algoritmo de EM, se garantiza que maximizando la función anterior respecto a los nuevos parámetros, se obtendrá una mayor verosimilitud en la siguiente iteración:

$$max_{\delta}\left[Q(\delta|\bar{\delta})\right] \dot{\Rightarrow} \mathrm{P(O|}\ \bar{\delta}) \geq \mathrm{P(O|}\delta) \qquad (2.9)$$

Este proceso se repetirá para ir obteniendo nuevos parámetros en cada iteración, para seguir aumentando la verosimilitud hasta el punto en el que el algoritmo converja o el incremento de verosimilitud sea mínimo.

A continuación, se hará una distinción entre los dos pasos del algoritmo:

En primer lugar, el **paso de *Expectación***, tiene como misión calcular los elementos de la ecuación 2.9, que dependen del modelo anterior, principalmente el término $P(Q|\mathrm{O}, \bar{\delta})$, es decir, las probabilidades de todas las secuencias de estados que se encuentran en el modelo anterior y las observaciones.

Los parámetros que se van a estimar son:

- La probabilidad de estar en el estado S_i en el instante t, que vienen dada por la probabilidad de ocupación del estado anteriormente definido, que se podía calcular en función de las variables *forward y backward*.
- El número esperado de transiciones desde el estado S_i, la cual puede obtenerse a partir de las variables anteriormente mencionadas, de la siguiente forma:

$$\sum_{t=1}^{T-1} \gamma_t(i) \qquad (2.10)$$

- El número esperado de transiciones desde el estado S_i al estado S_j . Para ello, inicialmente se definirá la probabilidad de

transición entre ambos estados, y posteriormente se obtendrá el número de transiciones sumando los valores obtenidos:

$$\varphi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | 0, \delta) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\delta)} \qquad (2.11)$$

$$\sum_{t=1}^{T-1} \varphi_t(i,j) \qquad (2.12)$$

En segundo lugar, se tendrá el **paso de Maximización**, consistente en maximizar la función (ecuación 2.8) una vez ya calculados los parámetros en el paso anterior, y dará lugar a unos parámetros nuevos, siendo los siguientes:

$$\bar{\pi}_i = frecuencia\ esperada\ en\ S_i en\ el\ instante (t = 1) = \gamma_1(i) \qquad (2.12)$$

$$\bar{a}_{ij} = \frac{numero\ esperado\ de\ transiciones\ desde\ S_i a\ S_j}{numero\ esperado\ de\ transiciones\ desde\ S_i} = \frac{\sum_{t=1}^{T-1} \varphi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad (2.13)$$

$$\bar{b}_j(j) = \frac{numero\ esperado\ de\ veces\ en\ el\ estado\ j\ observando\ v_k}{numero\ esperado\ de\ veces\ en\ el\ estado\ j} = \frac{\sum_{t=1, s.t. O_i = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad (2.14)$$

Los algoritmos descritos anteriormente, que solucionan los tres problemas enunciados al principio de esta subsección, serán empleados en este proyecto, tanto en el proceso de entrenamiento como en el de evaluación del sistema de reconocimiento.

2.3.6. Fase de reconocimiento

Luego de obtener el modelo acústico basado en HMM para cada transcripción fonética de las palabras contenidas en el diccionario, así como también el modelo de n-gramas en el modelo acústico a partir del conjunto de textos que bien podrían ser las transcripciones textuales del corpus de grabaciones de voz.

Entonces para una secuencia de observación $O = O_1, O_2, \dots, O_T$, y un modelo $\delta = (A, B, \pi)$ surge la necesidad de determinar la secuencia de estados correspondiente $Q = (q_1, q_2, \dots, q_r)$ como una ruta óptima para la secuencia de observación O.

Una forma de estimar la probabilidad combinada de los modelos acústico fonético y de lenguaje para una cierta secuencia de observaciones teniendo en cuenta la representación del espacio de búsqueda que hemos llevado a cabo es utilizando la aproximación de Viterbi.

2.3.6.1. Algoritmo de Viterbi

El algoritmo de Viterbi consiste en la asimilación de la probabilidad total por aquella del camino con mayor valor de esta, teniendo en cuenta que en la mayoría de los casos existe un camino dentro de los que componen con una probabilidad mucho mayor que el resto.

Además, si una vez calculada la probabilidad de Viterbi para una secuencia de observaciones recuperamos el camino (la secuencia de estados del modelo de Markov) que nos ha llevado a esta podremos saber cuál es la frase que maximiza esta estimación de la probabilidad.

La probabilidad de Viterbi que con una secuencia de longitud ***t*** de símbolos pertenecientes al alfabeto del modelo (las ***t*** primeras observaciones de la entrada) se llegue a un estado ***s.*** (*Calvo, 2010*). Puede calcularse como sigue:

$$\Pr(x_1 \dots x_t|s) = \begin{cases} p(s|s_0).q(x_1|s) & si\ t = 1 \\ max_{s' \in S} \Pr(x_1 \dots x_{t-1}|s').p(s|s').q(x_t|s) & si\ t \neq 1 \end{cases} \quad (2.15)$$

De este modo, la probabilidad total de haber emitido la secuencia acústica de longitud n caracterizada por las observaciones $x_1 \dots x_n$ dado un modelo de Markov M seria:

$$\Pr(x_1 \dots x_n|M) = max_{f \in F, s \in S} \Pr(x_1 \dots x_t|s).p(f\,|\,s) \quad (2.16)$$

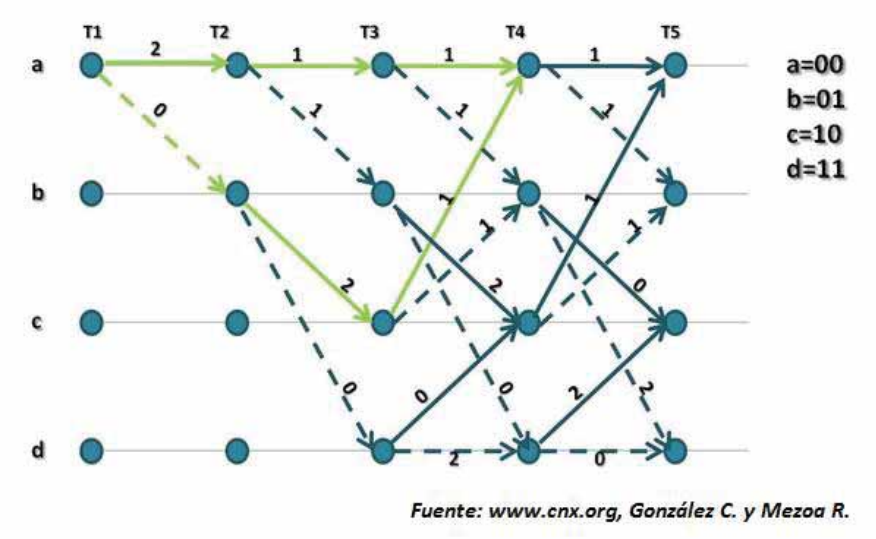


Figura 13: Ejecución del algoritmo Viterbi.

Fuente: www.cnx.org

2.3.7. Evaluación

El método más utilizado para evaluar la calidad de un sistema ASR es la tasa de error por palabra WER (Word Error Rate), una métrica común de la realización de un reconocimiento de voz.

La principal dificultad para medir el rendimiento radica en el hecho de que la secuencia de palabras reconocida puede tener una longitud diferente de la secuencia de palabras de referencia.

Este problema se resuelve alineando primero la secuencia de palabras reconocida con la secuencia de palabras de referencia usando alineación dinámica de cadenas, y cuya fórmula es la siguiente.

$$WER(\%) = \frac{S+I+D}{N} * 100 \quad (2.17)$$

Donde:

- N: Es el número de palabras en la sentencia de referencia.
- S: Es el número de reemplazos
- I: Es el número de inserciones
- D: Es el número de eliminaciones

Un ejemplo básico de alineamiento:

Sentencia de referencia:	***imatan***	***ruwashanki***	***chay mayu***	***chimpapi***	*********
Sentencia reconocida:	***maytan***	***rishanki***	***chay mayu***	***chimpa***	***chimpapi***
Evaluación:	S	S		S	I

$$WER = \frac{3+1+0}{5} * 100 = 0.8*100=80\%$$

A pesar que el conjunto de herramientas de reconocimiento de voz nos da los resultados en la medida WER, en esta tesis hacemos uso del complemento de las tasa de error por palabra, la cual es la precisión por palabra (WACC) y está representada de la siguiente forma:

$$WACC = 1-WER$$

2.4. Redes neuronales profundas

Las redes neuronales profundas son una forma alternativa de aprendizaje y de procesamiento automático, que intenta simular el funcionamiento del cerebro. Para ello utiliza una red neuronal basada en la propagación hacia adelante (*feed-forward*), la cual toma varias tramas de coeficientes como entrada y como salida genera probabilidades sobre los estados de HMM tal como se muestra en la Figura 14. Estas redes tienen un número considerable de capas ocultas y son entrenadas usando nuevos métodos que mejoran a los métodos convencionales.

La conexión entre capas adyacentes se consigue con la asignación de unos pesos iniciales de baja magnitud y aleatorios, podemos verlo con una representación matricial llamada W. Siendo esencial la aleatoriedad para evitar que todas las unidades ocultas en una capa tengan exactamente los mismos valores en cuanto a los pesos.

Los modelos más flexibles, son aquellos con un gran número de capas y de elementos en cada capa, que son capaces de modelar relaciones altamente no lineales entre entradas y salidas.

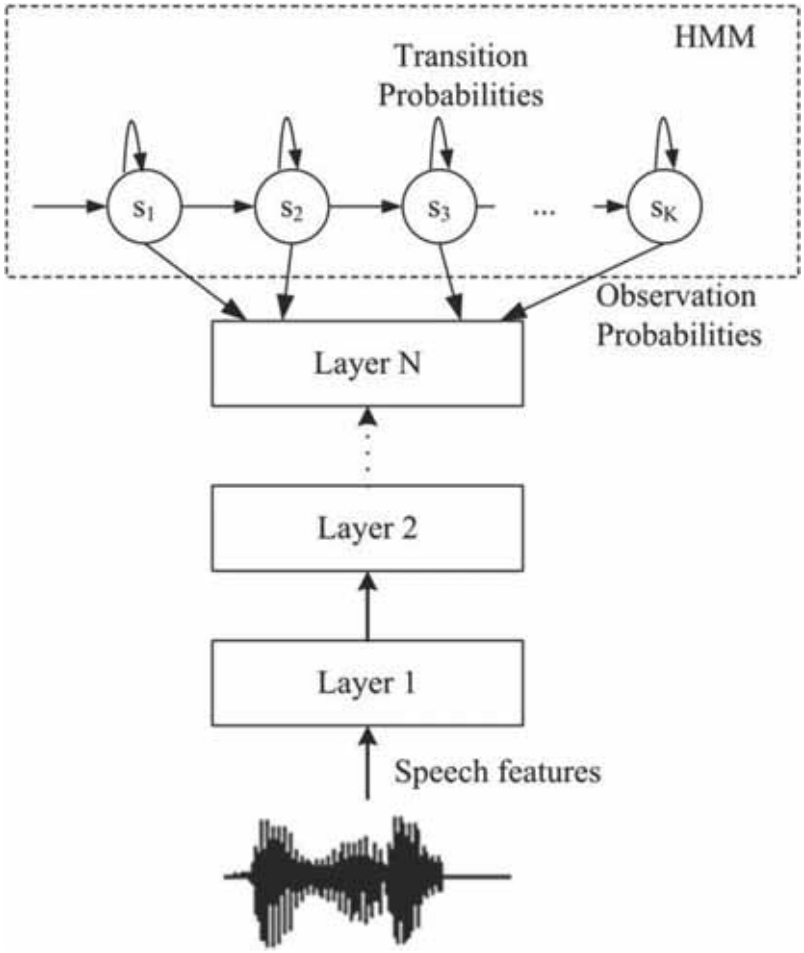


Figura 14: Arquitectura del modelo DNN-HMM.

Fuente. (Kipyatkova y Karpov ,2016)

2.4.1. Capas

Una red neuronal profunda se compone esencialmente de 3 capas.

- **Capa de entrada:** Capa encargada de tomar como entrada una cantidad de 9 frames y cada frame con 39 coeficientes MFCC (351 coeficientes).
- **Capa oculta:** Capa que se ocupa del procesamiento de los datos de entrada y enviar a la capa de salida.
- **Capa de salida:** Capa que nos da las probabilidades de ser la salida correcta de los distintos fonemas existentes en un idioma, con sus distintas variaciones fonéticas (408 fonemas).

2.4.2. Funciones de activación

La función de activación recibe como entrada la suma de todos los números que llegan por las conexiones entrantes, transforma el valor mediante una fórmula, y produce un nuevo número. Existen varias opciones. Uno de los objetivos de la función de activación es mantener los números producidos por cada neurona dentro de un intervalo.

- **Función de activación Tangencial**

Es la versión continua de la función signo y se usa en problemas de aproximación. Es importante por sus propiedades analíticas. Es continua a valores en [-1,1] e infinitamente diferenciable, Esta función está definida como:

$$\tanh(x) = \frac{senh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Otras funciones de activación muy conocidas son: la función sigmoidal y la función RELU.

2.4.3. Propagación hacia adelante (forward propagation)

En la fase de propagación hacia adelante, la red neuronal tiene como entrada un vector de características a través de las neuronas de la capa de entrada. Estas neuronas dan una respuesta igual a su entrada y dirigida a cada una de las neuronas j de la primera capa oculta. Cada una de éstas, al recibir esas entradas, calcula su suma ponderada S de acuerdo con los pesos de las interconexiones de entrada. La suma ponderada S pasa al elemento activador no lineal, que presenta como salida una función de activación f(S) de esa suma ponderada S. Esta salida constituye, a su vez, una entrada para cada neurona i de la capa de siguiente, a su vez, calcula una suma ponderada de sus entradas, la cual pasa por su activador no lineal, y así sucesivamente hasta llegar a la capa de salida. Con esto termina la fase de propagación hacia adelante.

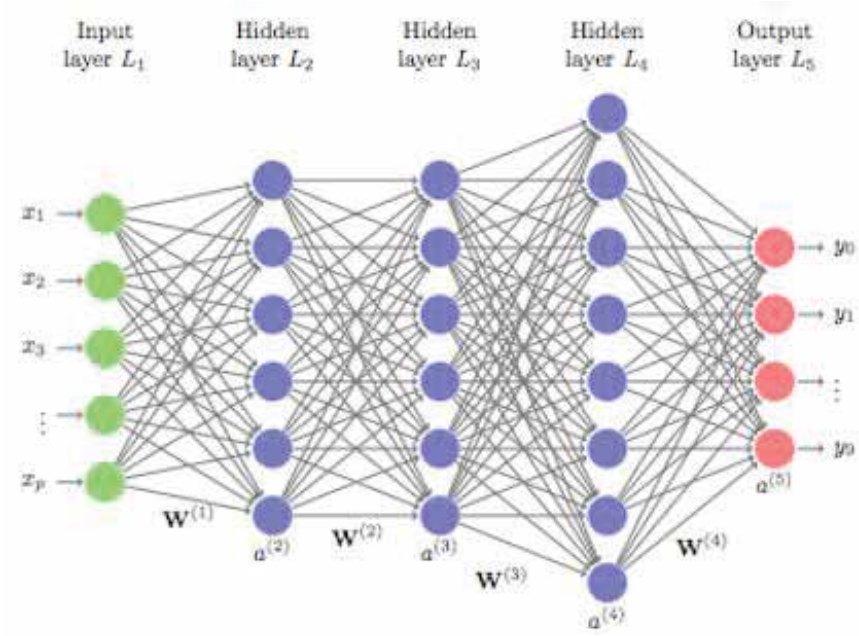


Figura 15: Arquitectura de una Red Neuronal Profunda.

Fuente: http://uc-r.github.io/feedforward_DNN

2.4.4. Propagación hacia atrás (Back propagation)

La fase de propagación hacia atrás comienza con la comparación de la salida real de la red con la salida deseada, calculándose el error δ_j para cada neurona j de salida. Este error es igual a:

$$\delta_j = (t_j - y_j) f'(S_j)$$

Donde:

t_j= la salida deseada de la neurona j.

y_j = la salida real de la neurona j.

$f'(x)$= la derivada de la función activadora sigmoidal f(x) en el punto x.

S_j = la suma ponderada de las entradas a la neurona j.

Después, se calcula el valor del error dj para cada neurona intermedia j en la forma siguiente:

Se hace la suma ponderada de los errores de salida, utilizando los pesos de interconexión w_{ij} entre esta neurona j intermedia y cada neurona k de salida, y esta suma ponderada se multiplica por la derivada f' de la función activadora sigmoidal f(S), con el fin de "escalar" el error y de forzar una corrección más grande cuando la suma ponderada queda cerca de la zona de pendiente máxima de la curva sigmoidal

$$\delta_j = \sum_k d_k w_{kj} f'(s_j)$$

A continuación se calcula el incremento Δw_{ji} a los pesos, de acuerdo con la fórmula:

$$\Delta w_{ji} = n \delta_j a_i$$

Donde:

δ_j = el valor de error de la neurona posterior.
a_i = el valor de salida real de la neurona anterior.
n = la tasa de aprendizaje.

Como parte del aprendizaje con retro alimentación el algoritmo necesita un valor de aprendizaje muy pequeño para controlar el valor de los pesos durante el cambio que surge como parte de la propagación de error.

2.5. Herramienta KALDI

Este capítulo explicará los kits de herramientas de Kaldi, el conjunto de herramientas principal que se utiliza en esta tesis, dando una breve introducción al respecto. Además, cada componente del discurso sistema de reconocimiento que es compatible con este kit de herramientas se ilustrará, como la función extracción, modelado acústico y decodificación.

2.5.1. Arquitectura de la herramienta KALDI

La herramienta Kaldi es un software de código abierto diseñado para el reconocimiento de voz. El código de este kit de herramientas está escrito en lenguaje C++ y es lanzado libremente por la Licencia de Apache V2.0.

Este kit de herramientas ASR tiene todo los algoritmos del reconocimiento de voz necesario, incluidas las plantillas listas que se pueden usar para entrenar diferentes modelos acústicos mediante el uso de corpus de voz grabados, como TIDIGITS y TIMIT. El objetivo principal de Kaldi es construir un software que sea moderno, fácil de usar, lo suficientemente flexible para extender y actualizar, y poderoso en el entrenamiento del modelo acústico. Principalmente, el kit de herramientas de Kaldi depende del autómata de estado finito (*FST-Finite State Transducer*) y archivos de script de Shell para crear un sistema de reconocimiento de voz.

La arquitectura de la herramienta Kaldi se puede describir como una secuencia de modelos que consisten en bibliotecas y scripts de entrenamiento que se utilizan para acceder a estas bibliotecas. Modelos en los niveles más bajos dependen de uno o más modelos en los niveles superiores. Como se ve en la Figura 16, Kaldi depende de dos bibliotecas externas principales: el OpenFst, que está diseñado para autómata de estados finitos (FST) y las Subrutinas de álgebra lineal básica (BLAS) y Paquete de Algebra Lineal (LAPACK) que están diseñados con fines de álgebra lineal. Ambas bibliotecas están disponibles sin costo.

Este conjunto de herramientas ASR es compatible con algunos modelos de bibliotecas que son dividido en dos grupos; cada grupo es apoyado por una sola biblioteca externa, mientras que el modelo decodificable se considera como un enlace que funciona al conectar estos dos grupos de modelos. Para alcanzar las funcionalidades de la biblioteca, Kaldi proporciona una línea de comando de C ++ herramientas que se pueden llamar utilizando el lenguaje de scripts (shell). Además, Kaldi usa un sistema de tuberías para leer y escribir estas herramientas y eso hace que sea fácil encadenar diferentes herramientas (Sarah, 2018).

Además, la herramienta Kaldi tiene un ejecutable que se utiliza para cargar la entrada, y almacenando la salida en, los archivos. Además, este conjunto de herramientas está diseñado de una manera que podría aceptar las nuevas características agregándolas como un nuevo código y línea de comando sin la necesidad de cambiar los algoritmos originales.

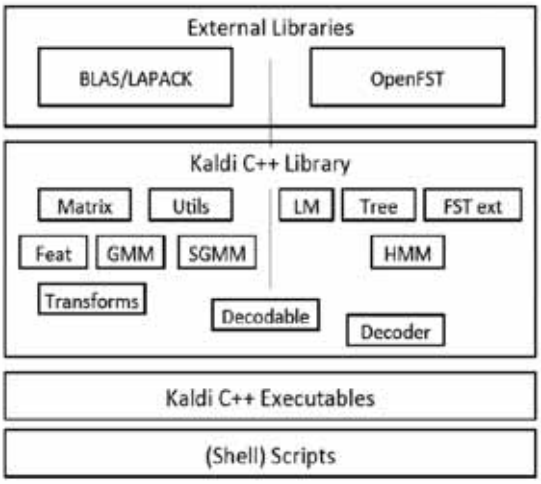


Figura 16: La arquitectura de la herramienta Kaldi.

Fuente: (Povey, 2012).

En general, el sistema ASR trata con diferentes tipos de kits de herramientas de código abierto que están diseñados para fines de reconocimiento de voz. Ejemplos de estos kits de herramientas son RWTH ASR, HTK (que es similar en concepto y objetivo a Kaldi) y Sphinx4. Sin embargo, kit de herramientas de Kaldi se puede caracterizar entre los otros kits de herramientas de ASR por las siguientes características (Povey, 2012):

1. Código de Kaldi integrado con los autómatas de estado finitos (FST) mediante el uso de la biblioteca de OpenFst como una biblioteca principal.
2. Altamente de apoyo para operaciones algebraicas lineales porque implementa la matriz biblioteca, que incluye las bibliotecas BLAS y LAPACK difamadas.
3. Diseño de código que es fácil de extender y modificar debido a los algoritmos que son proporcionado de la manera más genérica posible.
4. Licencia gratuita de la fundación software Apache.
5. Proporcionar recetas completas que se utilizan para crear sistemas ASR mediante la utilización de algunas bases de datos, como TIMIT, TIDIGITS o cualquier información grabada.
6. El código de Kaldi se prueba de forma inclusiva para que pueda proporcionar resultados precisos.

2.5.2. Componentes del sistema ASR que usa la herramienta kaldi

Es muy importante ilustrar cómo la herramienta Kaldi es compatible con cada parte de estos componentes (Povey, 2012):

- **Extracción de características:** Durante la fase de extracción de características, Kaldi apunta a extraer las características estándar de MFCC y PLP. Sin embargo, Kaldi admite otros tipos de los métodos de extracción de características, como la media cepstral y la normalización de la varianza, HLDA, STC / MLLT, VTLN y LDA.
- **Modelado acústico:** El objetivo principal de la herramienta Kaldi es soportar los tradicionales modelos mixto Gaussianos (GMM) y el sub espacio del modelo de Mixto Gaussiano (SGMM). Para GMM, Kaldi admite el diseño

de covarianza diagonal y completa utilizando un Clase DiagGmm para GMM diagonal y una clase AmDiagGmm para GMM dentro del modelo acústico. Para SGMM, Kaldi usa una clase especial llamada AmSgmm.
Además, Kaldi soporta el modelo oculto de Markov diseñando una clase especial llamado HmmTopology que permite el uso de los estados no emisores y reconectando el pdf de los estados.

- **Adaptación del altavoz:** La herramienta Kaldi admite la adaptación del modelo de espacio utilizando regresión lineal de máxima verosimilitud (MLLR) y espacio de características Regresión lineal de máxima verosimilitud (FMLLR). Además, las normalizaciones son compatibles con Kaldi mediante el empleo de una forma más genérica nombrada por la transformación exponencial.
- **Modelado del lenguaje**: La herramienta Kaldi admite cualquier tipo de modelo de lenguaje mediante el uso de herramientas específicas que están diseñadas para convertir el modelo de lenguaje a FST, como los kits de herramientas SRILM e IRSTLM.
- **Construcción de decodificación de gráficos**: Kaldi utiliza los autómatas de estados finitos ponderados (*WFST-Weighted Finite-State Transducers*) para la construcción de decodificación de gráficos.
- **Decodificación:** El kit de herramientas Kaldi admite diferentes tipos de decodificación, como SimpleDecoder, FastDecoder y BiglmDecoder con su generador de celosía versión. El decodificador simple por lo general funciona como un depurador para altamente avanzado decodificador, mientras que FastDecoder se considera como un decodificador más avanzado que puede manejar el mayor número de estados que están activos al mismo tiempo. Sin embargo, el BiglmDecoder se utiliza para manejar el enorme modelo de lenguaje que a veces excede un millón de arcos.

Además, Kaldi ofrece varias utilidades, como registro e informe de errores, análisis de línea de comandos y funciones de matemáticas y STL. (Sarah, 2018).

2.5.3. Estructuras de directorios Kaldi y bibliotecas requeridas

Como se mencionó anteriormente, la herramienta Kaldi recopila diferentes tipos de bibliotecas y algoritmos que podrían usarse para diseñar diferentes tipos de sistemas ASR. Sin embargo, hay algunos requisitos previos importantes que se deben instalar para ayudar con la instalación y ejecución este kit de herramientas:

- **ATLAS:** software de álgebra lineal sintonizado automáticamente (ATLAS)
- **Autoconf:** utilizado para las operaciones de configuración
- **git:** sistema de control de versiones que se utiliza para distribuir códigos para el usuario después de actualizado por los desarrolladores.
- **Automake:** utilizado para crear Makefiles
- **Subversion (svn):** otro sistema de control de versiones que es importante para Kaldi descarga e instalación,
- **wget**: utilizado para recuperar datos mediante protocolos HTTP, HTTPS y FTP
- **awk:** un lenguaje de programación que se utiliza para el procesamiento y la extracción de datos.

- Conjuntos de herramientas de modelo de lenguaje SRILM e IRSTLM.

Después de instalar todos estos requisitos, la herramienta Kaldi se puede descargar desde el repositorio Git Hub, e instalado siguiendo todas las instrucciones en el archivo "INSTALAR" dentro del Directorio de Kaldi. Asegúrese de que SRILM y IRSTLM estén instalados correctamente en caso de que los usuarios quieran usar el modelo de lenguaje. (Sarah, 2018).

El directorio principal de Kaldi se puede definir como una red de directorios, subdirectorios y archivos de documentación. Como se muestra en la Figura 17, el directorio de alto nivel consta de:

- **egs:** Que representa, por ejemplo, y es considerado como uno de los más importantes directorios que recopilan scripts de plantillas con diferentes versiones para algunos corpus de voz que ayudan a los usuarios a construir su sistema ASR fácilmente. Dentro de cada uno de estos directorios de plantilla hay estructuras de subdirectorios estándar que incluyen: conf (siglas de configuración), datos, exp (soportes de experimentos), utils (stands para servicios públicos), pasos y locales. Los subdirectorios de datos mantienen toda la información sobre los datos que se usan en el sistema ASR, mientras que la exp contiene resultados del proceso de entrenamiento y decodificación. Los directorios Utils y Steps contienen todo el scripts necesarios para construir los sistemas ASR. Además de estos subdirectorios, Los directorios de plantillas tienen archivos de scripts, como run.sh, cmd.sh y path.sh, que se utilizan para ejecutar el sistema de reconocimiento de voz.
- **src:** Significa fuente y contiene todos los códigos fuente del kit de herramientas de Kaldi.
- **tools:** Contiene todas las herramientas externas
- **misc:** Contiene herramientas e información adicionales, como la conversión HTK y el logotipo de Kaldi
- **windows:** Contiene todas las herramientas y archivos necesarios para ejecutar Kaldi en la ventana.
- Archivos de documentación, como INSTALAR y README.

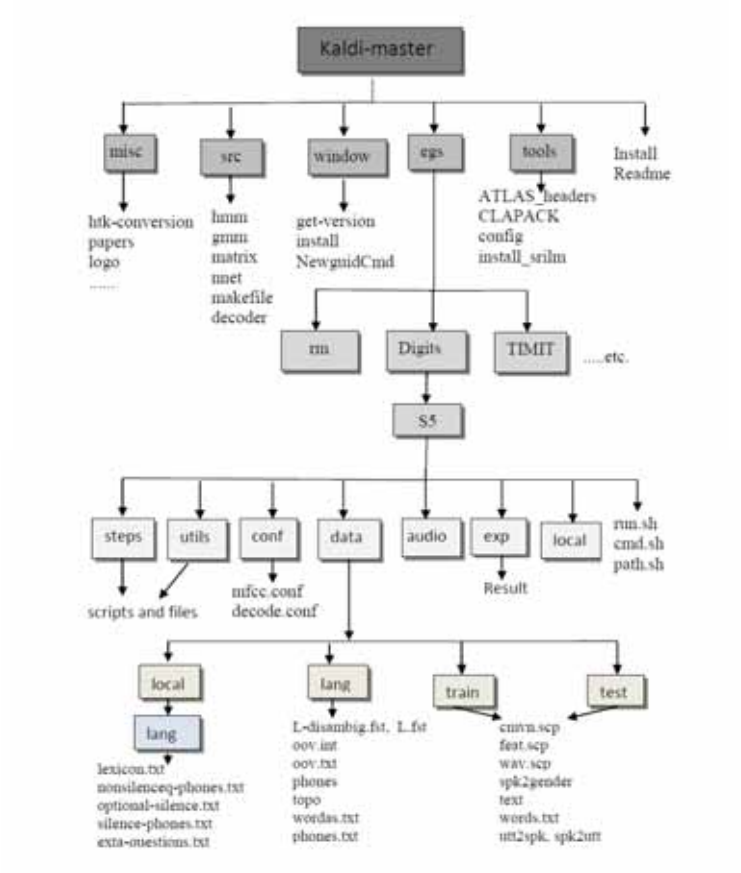


Figura 17: Visión general de la estructura de directorios de Kaldi.

Fuente: (Sarah, 2018).

2.5.4. Decodificador de Kaldi

2.5.4.1. Automata de estado finito (FST)

El autómata de estado finito (FST) etiqueta sus estados con símbolos de entrada y salida para traducir de la secuencia de entrada a la salida.

A veces, el autómata de estado finito se representa etiquetando la transición con peso, y los símbolos de entrada y salida son llamados por el autómata de estado finito ponderado (WFST). Estos pesos podrían representar una duración, probabilidad, penalización, etc.

WFST se puede definir según el concepto algebraico semiautomático que es similar a la estructura del anillo. El *semiring* es un conjunto de K equipado con dos binarios operaciones: operación acumulativa $\oplus$, y operación asociativa $\otimes$. Ambas operaciones deben tener elementos de identidad $\bar{0}$ y $\bar{1}$. La operación de multiplicación es distributiva sobre operación adicional desde ambas direcciones, y a $\otimes\bar{0} = \bar{0} \otimes$ a.

La Figura 18, muestra un diagrama simple del automata de estado finito ponderado donde cada estado se realiza por círculos y se etiqueta con diferentes números. El círculo audaz representa el estado inicial mientras que los círculos dobles representan el estado final. Además, cada arco de transición se mapea con el peso, así como con la entrada y la salida. Por ejemplo, el auto *loop* del "estado 1" ponderado por 0.5 y se etiqueta con "r" como entrada e "y" como salida.

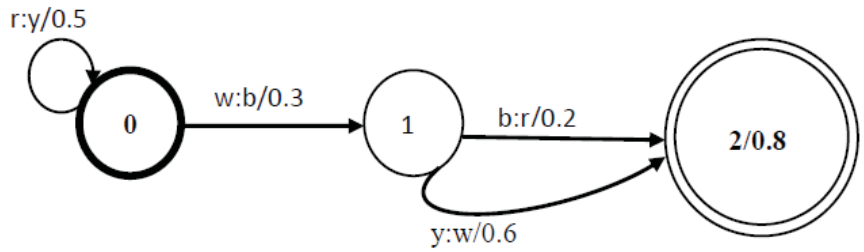


Figura 18. Máquina de estados finitos ponderado simple (WFST).

Fuente: (Sarah, 2018).

En general, el kit de herramientas de Kaldi usa el WFST en casi todo el entrenamiento y decodificación algoritmos. El gráfico de descodificación en el kit de herramientas de Kaldi se puede crear mediante la construcción de HCLG gráfico, que se puede definir en la siguiente ecuación (Sarah, 2018):

$$HCLG = H \circ C \circ L \circ G$$

La funcionalidad de cada componente en esta ecuación se puede describir como la siguiente:

G: representa la codificación del modelo de lenguaje.

L: representa el léxico, donde sus símbolos de entrada representan fonemas y su salida los símbolos representan palabras

C: representa dependencia del contexto, donde sus símbolos de entrada se refieren al contexto dependiente de los fonemas y sus símbolos de salida se refieren a los fonemas.

H: representa las definiciones de HMM. Sus símbolos de entrada representan los ID’s de transición y los símbolos de salida representan fonemas dependientes del contexto.

- el **o** representa las operaciones binarias asociativas en el FST.

Además, los desarrolladores de Kaldi trabajan en insertar el símbolo de desambiguación en minimizar y determinar la salida de HCLG. Además, trabajan para preservar la estocasticidad del HCLG y probarlo. El enfoque de HCLG se puede resumir como la siguiente ecuación:

$$HCLG = asl(min(rds(det(H' \circ min(det(C \circ min(det(L \circ G))))))))$$

Donde asl representa el *add-self-loops*, rds se refiere a los *remove-desambiguación-symbols*, y H' es H sin el auto-lazo.

2.5.4.2. SRILM

SRILM es un conjunto de herramientas para crear y aplicar modelos de lenguaje estadístico (LM), principalmente para su uso en reconocimiento de voz, etiquetado, segmentación estadística, y traducción automática. Ha estado en desarrollo en el SRI *Speech Technology and Research Laboratory* desde 1995. (Sarah, 2018).

SRILM consta de los siguientes componentes:

- Un conjunto de bibliotecas de clases C ++ que implementan modelos de lenguaje, estructuras de datos de soporte y funciones de utilidad diversas.
- Un conjunto de programas ejecutables construidos sobre estas bibliotecas para realizar tareas estándar, como entrenar LM y probarlos en datos, etiquetar o segmentar texto, etc.
- Una colección de scripts diversos que facilitan tareas menores relacionadas.

SRILM se ejecuta en plataformas UNIX y Windows.

SRILM se ha utilizado en una gran variedad de aplicaciones de modelado estadístico.

2.5.5. Programación basada en scripting (Shell)

Un Shell Script es un script para shell de comandos (terminal). Para crear un script basta con editar un fichero nuevo y poner en el nombre .sh Ejemplo: HolaMundo.sh

Una vez creado, empezamos a editarlo. Se puede utilizar un editor de textos gráfico como gedit o un editor en terminal como vim, nano o emacs.

En la primera línea del script se debe indicar que shell vas a usar (/bin/bash/, /usr/bin/perl, etc.) Aunque da igual la que uses lo importante es el contenido:

#! /bin/bash, donde #!, Se conoce con el nombre de *Sha Bang*.

Se denomina "*sha-bang*" a la secuencia #! con la que se inician los scripts. Su función es indicarle al sistema que se trata de un conjunto de comandos para que sean interpretados. En realidad, es un número mágico de dos bytes. El número mágico es un marcador especial para indicar el tipo de archivo, en este caso, indica que se trata de un script de shell ejecutable.

Para introducir comentarios se debe poner #. Por cada línea que desees poner un comentario, lo primero que debe tener es #. Es importante añadir comentarios comentando la utilidad del script o las variables que se crean. (Velasco y Martín, 2008).

CAPÍTULO III

DESARROLLO DEL SISTEMA

3.1. Construcción del Corpus de voz

Uno de los requerimientos más importantes para el desarrollo de un conversor de voz a texto es sin duda el conjunto de grabaciones de voz, dichas grabaciones se realizan a partir de frases extraídas de diferentes textos, como parte de la robustez del sistema se requiere que el corpus de voz sea rica fonéticamente y balanceado, entendiendo por fonéticamente rico al hecho de contar con todos los fonemas del idioma y balanceado por el hecho de que los fonemas aparezcan la misma cantidad de veces posible.

Teniendo una distribución de las personas de la siguiente forma.

Id-locutor	Genero	Edad	Localidad	Grupo
01	Femenino	23	Marangani	Entrenamiento
02	Masculino	28	Sicuani	Entrenamiento
03	Femenino	19	Sicuani	Entrenamiento
04	Femenino	39	Sicuani	Entrenamiento
05	Masculino	23	Sicuani	Entrenamiento
06	Masculino	26	Sicuani	Entrenamiento
07	Femenino	35	Tinta	Entrenamiento
08	Masculino	40	Tinta	Entrenamiento
09	Femenino	27	San pablo	Entrenamiento
10	Masculino	23	Marangani	Entrenamiento
11	Masculino	23	Sicuani	Entrenamiento
12	Masculino	63	Marangani	Entrenamiento
13	Femenino	30	Sicuani	Entrenamiento
14	Masculino	34	Sicuani	Entrenamiento
15	Femenino	28	Sicuani	Entrenamiento
16	Masculino	18	Sicuani	Entrenamiento
17	Masculino	27	Acomayo	Prueba
18	Masculino	36	Checca	Entrenamiento
19	Masculino	28	Yanaocca	Entrenamiento
20	Femenino	18	Sicuani	Entrenamiento
21	Masculino	40	Checca	Entrenamiento
22	Masculino	25	Sicuani	Entrenamiento
23	Masculino	30	Checca	Entrenamiento
24	Femenino	25	Sicuani	Entrenamiento
25	Masculino	23	San Pablo	Entrenamiento
26	Femenino	20	Tinta	Entrenamiento
27	Femenino	40	Sicuani	Entrenamiento
28	Femenino	27	Sicuani	Prueba
29	Masculino	24	Acomayo	Entrenamiento
30	Masculino	38	Checca	Entrenamiento
31	Femenino	24	Sicuani	Entrenamiento
32	Masculino	24	Sicuani	Entrenamiento

Tabla 4: Características de los locutores.

Fuente: Elaboración Propia.

La edad de las personas que son parte del corpus de voz fluctúa entre 19 y 40 años, esta elección es tomada debido a que es en este intervalo de edad que se obtienen audios con una buena característica acústica, esto contrasta con audios de personas que están fuera de este intervalo que se van obteniendo audios con malas características acústicas y por ende una baja precisión en el reconocimiento de voz (Pellegrini, 2012) y (Russell & D'Arcy, 2007).

3.1.1. Obtención de corpus de texto

Por lo tanto, la primera tarea a realizar fue la obtención de frases, que se realizó a partir de distintos fuentes textuales, siendo la mayor parte del aplicativo móvil "Santa Biblia Quechua Cusco" que es obtenido de la Sociedad Bíblica Peruana (2004), el cual consta de más de 20000 palabras sin nombres incluidos, producto de ello se obtuvo una cantidad de 6475 frases con un promedio de 8 palabras cada una.

Previo al proceso de grabación se tuvo que realizar los siguientes procedimientos como parte de una limpieza de palabras o símbolos que perjudican el reconocimiento de voz, como son: descartar todas las palabras en español como requisito primordial en el proceso de entrenamiento dado que no existe una transcripción fonética propia para dichos grafemas, descartar símbolos tales como acentuación, guion, punto, etc. y finalmente realizar la expansión de todas las palabras de su forma abreviada.

Después de la extracción de frases, se realizó la asignación de ellas de manera aleatoria a cada uno de las 32 personas a ser grabadas, quedando así un aproximado de 250 frases por persona donde internamente se agrupa por distintos contenidos textuales.

3.1.2. Obtención de audios de voz

Luego de la obtención del conjunto de textos en forma de frases empezamos la grabación de ellos con las 32 personas que se encuentran dentro del proceso de entrenamiento, todo ello mediante el programa de grabación y edición de audio **Audacity-win-2.2.2** disponible en forma gratuita en el entorno de Ubuntu. Como parte de las sesiones de grabación se debe de tomar en cuenta los siguientes detalles.

3.1.2.1. Propiedades de grabación

Las grabaciones de voz para cada oración son en promedio 10 segundos, con una velocidad natural al momento de pronunciar, y con una intensidad de pronunciación que no exceda la amplitud de 1 para una mejor calidad de audio y por ende un mejor rendimiento en el proceso de extracción de características y un mejor desempeño del sistema en conjunto.

- El ambiente en el cual se realiza las sesiones de grabación son ambientes de estudio, de habitación común y otros ambientes rústicos de la zona, cualquier ruido o sonido externo que no sea tan relevante no tiene ninguna repercusión en el reconocimiento de voz, gracias al proceso de extracción de características se omiten esos valores, así

como también valores no deseados como el ruido, murmuro, respiración, etc.

- Las grabaciones fueron realizadas en una laptop usando la tarjeta de sonido interna que posee.
- El micrófono que se utilizó en las grabaciones es un micrófono externo a la laptop.
- Las grabaciones fueron realizadas con un muestreo de 16kHz, codificación 16 bits, canal monomodo y guardadas como archivo PCM.
- Los archivos de voz son guardados en formato de audio .WAV, que es la representación más pura de un audio.
- Cada audio de voz grabada debe tener al inicio y al final valores de silencio en un rango de 500ms.

3.1.2.2. Entorno de micrófono

La posición del micrófono al momento de las grabaciones tiene una influencia muy significativa en la calidad del audio.

Un micrófono situado muy cerca de la boca, como un micrófono de boca, capta el habla con mucha más claridad. Sin embargo, también capta cualquier sonido o ruido que se pueda hacer con los dientes, labios o cualquier otra parte de la cavidad vocal.

Por otro lado, un micrófono ubicado muy lejano a la boca recoge la voz desde varias trayectorias reflejadas así como también ruidos externos lejanos al micrófono.

La calidad de estas dos señales es ligeramente diferente, lo que lleva a tasas de éxito de reconocimiento de voz variables.

Otro de los factores que influye en la calidad del audio es el entorno donde se desarrollan las grabaciones, puesto que existe una diferencia de grabación hecha al aire libre donde se tiene una mayor cantidad de ruido a otra grabación hecha en un ambiente cerrado.

3.1.2.3. Sesiones de grabación

Las personas graban más de 6475 frases con las siguientes indicaciones.

- Leen las frases tan clara y fluida como sea posible.
- Si ocurre algún problema de pronunciación o ruido muy resaltante, se repite con la sesión de grabación para dicha oración.
- Si la grabación de audio tiene exceso de silencio, ya sea en el inicio, final o intermedio se puede hacer un recorte de audio mediante la función de recorte del programa de Audacity.

Las 32 personas que fueron grabadas como parte del corpus de voz poseen un identificador único tal como se muestra en la Tabla 4.

La identificación de cada sentencia está representada por 4 dígitos, pero que en este caso concreto hacemos uso desde el 0000 hasta 0060.

Cada archivo de grabación .wav tendrá un identificador conteniendo el identificador del locutor, el id del libro y el id de la sentencia separados por el símbolo de guion (-).

Por ejemplo: 01-01-0000.wav

Id locutor: 01 Id libro: 01 Id sentencia: 0000

La cantidad de horas de grabación total que se tiene, inicia con el hecho de que cada oración grabada tiene un promedio de 10 segundos, multiplicada por las 240 frases por persona se obtiene 40 minutos, y finalmente multiplicada por 32 personas en el conjunto de entrenamiento se obtiene un corpus de voz mayor a 18 horas.

La riqueza del corpus de voz se puede medir por diversas características. En la siguiente tabla se pueden ver dichas características dentro del corpus de voz.

Características	Valor
Nº Archivos / Frases totales	6 475
Nº Palabras distintas	17 486
Nº de Horas de audio	18
Nº de horas sin silencios	12.5
Silencio (%)	27.7
Voz (%)	72.3
Hombres (%)	60
Mujeres (%)	40
Hablantes distintos	32
Velocidad de muestreo (Hz)	16 000
Bits por muestra	16
Canales de audio	1

Tabla 5: Características del corpus de voz para el quechua.

Fuente: Elaboración Propia

Luego de todas las sesiones de grabación y almacenamiento, se obtiene el corpus de voz final, con sus respectivas transcripciones.

Figura 19: Resultado de la construcción del corpus de voz.

Fuente: Elaboración Propia.

El corpus de voz completo está localizada en el repositorio de google drive: https://drive.google.com/drive/folders/1eI0vJsGgXyPdJ6zEf96fpBWyulsTOOB6?usp=sharing.

3.2. Preparación de la herramienta kaldi

Antes de empezar a desarrollar el conversor de voz a texto, necesitamos preparar algunas herramientas y prerrequisitos, tales como:

- Descargar la herramienta Kaldi y todas sus bibliotecas requeridas, revisar y validar los directorios de Kaldi para asegurar si todo fue instalado correctamente (como se muestra en la figura de directorios de Kaldi).
- Asegurarse de descargar la herramienta para el modelo de lenguaje (SRILM) y (IRSTLM).
- Recomendable usar sistema operativo Linux dado que ocurren menos problemas que en Windows. Este proyecto se desarrolló sobre la distribución de Linux "Ubuntu 16.04 LTS".
- Uso del intérprete de comandos basado en Unix y lenguaje de programación basado en scripts (bash).

3.3. Desarrollo del conversor de voz a texto

Después de descargar y preparar todos los requerimientos para el conversor de voz, existen 4 pasos primordiales para crear un conversor de voz a texto con la herramienta Kaldi: Preparación de archivos para el modelo acústico, para el modelo de lenguaje, entrenamiento y decodificación (Reconocimiento). También se desarrolla una interfaz gráfica para mostrar la funcionalidad del conversor de voz a texto. Cada uno de estos pasos es explicado a continuación.

3.3.1. Preparación de archivos para el modelo acústico

La preparación de archivos es un proceso que se realiza antes de la construcción de un conversor de voz a texto. Este paso logra organizar y preparar los archivos en directorios correctos y con un contenido correcto, la preparación de archivos es realizado de la siguiente forma.

- Dentro de directorio "Kaldi/egs" crear una carpeta llamado "quechua" (o cualquier nombre deseado). Este folder almacena todo lo relacionado con nuestro conversor de voz a texto para el idioma Quechua.
- Seguidamente dividimos los datos dentro de la carpeta "quechua", creamos una carpeta llamada "quechua_audio". Dentro de esta carpeta "kaldi/egs/quechua/quechua_audio", creamos dos carpetas "train" y "test", donde son almacenados los archivos ".WAV" distribuidos de tal manera que las grabaciones de 30 personas sean para la fase de entrenamiento y los 2 restantes para la fase de reconocimiento, teniendo un total de 32 personas dentro del corpus de voz.
- Regresando a una carpeta anterior "kaldi/egs/quechua" y creamos una carpeta llamado "data"; dentro de esta carpeta creamos 2 carpetas "train" y "test", estas carpetas están relacionadas a las carpetas con el mismo nombre del directorio "kaldi/egs/quechua/quechua_audio". Dentro de estas carpetas creamos los siguientes archivos necesarios (Notar que estos archivos tienen el mismo nombre, pero ellos están relacionados a diferentes datos).

wav.scp: Este archivo mapea cada identificación de frase con el directorio donde se ubica el archivo de voz correspondiente y se realiza tanto para los datos de entrenamiento, así como para los datos de prueba. Este archivo puede definirse como sigue:

<ID_frase> <ruta_completa_archivo_voz>

Una muestra del archivo wav.scp es:

```
01-01-0052  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0052.wav
01-01-0053  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0053.wav
01-01-0054  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0054.wav
01-01-0055  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0055.wav
01-01-0056  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0056.wav
01-01-0057  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0057.wav
01-01-0058  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0058.wav
01-01-0059  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/01/01-01-0059.wav
01-02-0001  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/02/01-02-0001.wav
01-02-0002  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/02/01-02-0002.wav
01-02-0003  /home/franklin/Escritorio/kaldi/egs/quechua/quechua_audio/train/01/02/01-02-0003.wav
# Y así susesivamente...
```

Archivo locutor a género (spk2gender): Este archivo define el género de los locutores que son parte de las grabaciones, estas pueden ser masculino (m) o femenino (f). En nuestro corpus de voz cada persona tiene un identificador único "ID_locutor". El formato del archivo spk2gender está definido de la siguiente forma:

<ID_locutor> <genero_locutor>

Una muestra del archivo spk2gender es la siguiente:

```
01 f
03 f
04 f
05 m
07 f
08 m
14 m
# Y así sucesivamente...
```

Teniendo en cuenta que las personas en este caso están identificadas por una serie de números de dos dígitos, pudiendo llegar a un máximo de 100 personas, se utiliza esta representación en vez de un nombre en específico por la poca cantidad de memoria que ocupa y la rapidez con la que pueden ser procesadas.

Archivo de transcripción de texto (text): Este archivo contiene las transcripciones de texto para cada archivo de audio. Este archivo está definido de la siguiente forma:

<ID_frase> <transcripción_de_texto>

Una muestra del archivo “Text” es la siguiente:

```
01-01-0001   CHAYMI SAPA P'UNCHAY INTI CHINKAYPUYTA QHAWAQPUNI KANI
01-01-0002   HINAN HUK P'UNCHAYQA IMAPUNITAQ CHAY ORQO QHEPAPIRI KAN
             NISPA RIRQANI
01-01-0005   CHAYMI LLAKISQA KARQANI CHAY TIYASQAYPI
01-01-0006   ICHAQA REQPUNI KANI CHAY ORQO PATATAQA K'ANCHASHAQ MAYU
             QHAWAQ
01-01-0008   IMATAN CHAYPI YAW P'ITU RUWASHANKI NISPA
01-01-0009   HINAN WILLARQANI IMAMANCHUS CHAY ORQO PATATA RISQAYTA
01-01-0010   ASKHA RUNAKUNAN CHAY HATUN MAYU CHIMPANPI LLAQTAKUNATA
             RIKUYTA MUNANKU
01-01-0011   ICHAQA MANAN HAYK'AQPAS MAYU PATALLAMANPAS
             CHAYARQANKUCHU
01-01-0012   PAYQA MANAN HAYK’AQPAS CHAY LLAQTATA RIKURQANCHU
# Y así sucesivamente...
```

Mapeo de frase a locutor (utt2spk): Este archivo mapea cada identificador de una frase al respectivo locutor que fue grabado. Este archivo está definido de la siguiente forma:

<ID_frase> <ID_locutor>

Una muestra del archivo utt2spk es:

```
01-01-0056 01
01-01-0057 01
01-01-0058 01
01-01-0059 01
01-01-0060 01
01-02-0001 01
01-02-0002 01
01-02-0003 01
01-02-0004 01
# Y así sucesivamente...
```

Mapeo de locutor a frase (spk2utt): Este archivo es el opuesto al archivo utt2spk, mapea cada locutor a los archivos de voz grabados por la misma. El archivo spk2utt puede ser creado a partir del archivo utt2spk haciendo el uso del algoritmo escrito en el lenguaje de programación Perl "utils/utt2spk_to_spk2utt.pl".

Archivo de segmentación (segments): Este archivo no será construido manualmente, puesto que la herramienta Kaldi realiza la segmentación internamente.

- Es muy importante que los archivos mencionados anteriormente tienen que estar ordenados y sin duplicidad. Para verificar ello la herramienta Kaldi utiliza dos *scripts*:
  - **utils/validate _data_dir.sh** (para verificar que el contenido este ordenado)
  - **utils/fix_data_dir.sh** (para solucionar problemas de orden)
- El proceso de preparación de archivos puede ser hecho manualmente o usando scripts basados en *bash*.

3.3.2. Preparación de archivos para el modelo de lenguaje

Los archivos para el modelo de lenguaje son necesarios para el desarrollo de un conversor de voz a texto. Primeramente necesitamos crear un directorio especial para contener todo lo relacionado al modelo de lenguaje, por lo tanto en el directorio "kaldi/egs/quechua/data" creamos una nueva carpeta llamada "local", dentro de esta carpeta creamos otra carpeta llamada "dict". Dentro de este directorio "kaldi/egs/quechua/data/local/dict" creamos los siguientes archivos:

lexicon.txt: Este archivo lista todas las palabras que estén en las transcripciones textuales con sus respectivas transcripciones fonéticas, donde cada palabra tiene que ser mapeada en una sola línea. También, este archivo contiene el silencio "SIL" y el ruido <UNK>. El formato del archivo es como la siguiente:

<palabra> <fonema 1> <fonema 2> ... <fonema n> , siendo n=cantidad de fonemas de la palabra.

Teniendo en cuenta que debe existir un espacio entre la palabra y sus respectivos fonemas.

Una muestra del archivo lexicon.txt es:

```
!SIL SIL
<UNK> SPN
ACHHUYRUSQA      A CHH U Y R U S Q A
ACHHUYUSQA      A CHH U Y U S Q A
AHINALLATAQ      A H I N A LH A T A Q
AHINALLATAQMI      A H I N A LH A T A Q M I
AHINAPI      A H I N A P I
AHINAPIN      A H I N A P I N
AHINAQA      A H I N A Q A
AKLLAKUN      A K LH A K U N
AKLLAKURQAN      A K LH A K U R Q A N
AKLLARIKUSPA      A K LH A R I K U S P A
AKLLARQAN      A K LH A R Q A N
AKLLASPA      A K LH A S P A
# Y así sucesivamente...
```

nosilence_phones.txt: Este archivo contiene todos los fonemas del idioma Quechua. Estos fonemas son los siguientes:

```
A
CH
CHH
CH'
E
H
I
K
KH
K'
L
LH
M
N
NH
# Y así sucesivamente...
```

silence_phones.txt: Este archivo contiene todas las formas de silencio que puedan aparecer.

```
SIL

SPN
```

optional_silence.txt: Este archivo contiene solo la representación del silencio "SIL".

extra_question.txt: Este archivo, contiene ecuaciones adicionales acerca de información contextual de los fonemas. Normalmente este archivo es creado internamente por la herramienta Kaldi.

corpus.txt: Este archivo contiene todas la frases posibles que pueden ocurrir en el conversor de voz, que en nuestro caso es el conjunto de transcripciones de texto que se tiene, y que serán ubicadas en el directorio "kaldi/egs/quechua/data/local".

Una muestra del archivo corpus.txt es:

```
CHAYMANTAQA MANAPUNIN LLOQSIMUNKICHU LLAPALLAN
MANUKUSQAYKITA KUTICHIPUNAYKIKAMA NISPA
WARMI WAWAYTA QUSAN MAQARUN
PANAYTA QUSAN MAQARUN
PAYTAQA YANANMI
QHARIN MAQAN
MANTAY TAYTAYTA MAQASQA
MANTAY TURAYTA MAQASQA
MAMAYPA QHEPA QHARIN MAMAYTA MAQAYUN
PAY HANPIQ RUNA ÑISQAMAN RIRANCHU

# Y así sucesivamente...
```

3.3.3. Preparación del directorio de lenguaje

Como parte de la preparación del modelo de lenguaje, después de crear el directorio "local/dict", necesitamos crear otro directorio requerido, llamada "lang". Este directorio puede ser generado usando el script "prepare_lang.sh" que es proporcionado por la herramienta kaldi. Este directorio tiene la siguiente forma:

```
#Código para preparar directorio de languaje
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang
```

Donde la carpeta "dict" es considerado como entrada del script "utils/prepare_lang.sh", mientras que la carpeta "data/local/lang" es temporal para las operaciones y el directorio "data/lang" es la salida del proceso de preparación de lenguaje. La frase "<UNK>" representa al ruido, pero que mapeara a palabras que no están dentro del vocabulario. La carpeta "lang" consiste de los siguientes archivos:

Phones.txt y words.txt: Estos archivos son considerados como archivos de tabla de símbolos que es usado por Kaldi para mapear un texto a un entero. Usualmente, Kaldi usa dos módulos escritos en perl para acceder a esos archivos "utils/int2sym.pl" y "utils/sym2int.pl".

```
<eps> 0
CH_B 15
CH_E 16
CH_I 17
CH_S 18
CHH_B 19
CHH_E 20
CHH_I 21
CHH_S 22

# Y así sucesivamente...
```

```
<eps> 0
!SIL 1
<UNK> 2
ACHHUYAMUQKUCHU 3
ACHHUYKUSQA 4
ACHHUYRUSQA 5
ACHHUYUSQA 6
AHINA 7
AHINA(1) 8
AHINAKAQTINQA 9
AHINAKAQTINQA(1) 10

# Y así sucesivamente...
```

L.fst: Este archivo es el lexicón en forma de una máquina de estado finito traduce el símbolo fonético a símbolos de palabras. Como el siguiente FST para números.

L_disambig.fst: Este archivo contiene las máquinas de estado finito con símbolos de desambiguación y con propio bucle "#0".

Archivo Topo: Este archivo representa la topología del Modelo Oculto de Markov que es usado para el sistema de reconocimiento de voz. El archivo Topo es creado a partir del módulo en perl "gen_topo.pl" que es proporcionado por la herramienta Kaldi. El archivo topo tiene una apariencia como la siguiente:

```
<Topology>
<TopologyEntry>
<ForPhones>
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112
113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
132 133 134
</ForPhones>
<State> 0 <PdfClass> 0 <Transition> 0 0.75 <Transition> 1 0.25 </State>
<State> 1 <PdfClass> 1 <Transition> 1 0.75 <Transition> 2 0.25 </State>
<State> 2 <PdfClass> 2 <Transition> 2 0.75 <Transition> 3 0.25 </State>
<State> 3 </State>
</TopologyEntry>
<TopologyEntry>
<ForPhones>
1 2 3 4 5 6 7 8 9 10
</ForPhones>
<State> 0 <PdfClass> 0 <Transition> 0 0.25 <Transition> 1 0.25 <Transition> 2
0.25 <Transition> 3 0.25 </State>
<State> 1 <PdfClass> 1 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3
0.25 <Transition> 4 0.25 </State>
<State> 2 <PdfClass> 2 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3
0.25 <Transition> 4 0.25 </State>
<State> 3 <PdfClass> 3 <Transition> 1 0.25 <Transition> 2 0.25 <Transition> 3
0.25 <Transition> 4 0.25 </State>
<State> 4 <PdfClass> 4 <Transition> 4 0.75 <Transition> 5 0.25 </State>
<State> 5 </State>
</TopologyEntry>
</Topology>
```

De la información anterior podemos notar que cada fonema tiene 3 estados emisores con el estándar de 3 estados (topología de izquierda-a-derecha), también se tiene un estado final que no emite y que sirve para conectarse con otros estados. Por ejemplo el valor (0.75) en el estado cero representa el valor de la probabilidad de transición del estado cero hacia sí mismo, y el valor (0.25) representa el valor de la probabilidad de transición del estado cero hacia el estado uno. Existe un caso especial para el fonema silencio que tiene una topología diferente, la cual consiste en un estado inicial emitente, un estado final emitente y unos tres estados emitentes en el medio.

oov.txt: Este archivo tiene solo una línea de contenido con el valor de "<UNK>" que representa a cualquier ruido que ocurra en la grabación. Durante el proceso de entrenamiento todas las palabras fuera del vocabulario son mapeados a "<UNK>", sin embargo en el archivo "lexicon.txt" la frase "<UNK>" es mapeado al fonema "SPN", que representa a un fonema inservible.

oov.int: Este archivo es el formato en enteros del archivo "oov.txt".

G.fst: Este archivo representa la máquina de estado finito del modo de lenguaje. El archivo G.fst puede ser creado por diferentes métodos, pero en este caso depende de 2 pasos. Primero, se crea el modelo de lenguaje n-gram en formato ARPA usando la herramienta de modelado de lenguaje SRILM. Esto puede ser hecho usando un archivo ejecutable proporcionado por la herramienta SRILM llamado "ngram-count". Durante este proceso se crean dos archivos: "lm.arpa" que contiene el modelo de lenguaje n-gram en formato ARPA y "vocab-full.txt". Estos dos últimos archivos están dentro del directorio "data/local/tmp". Como segundo paso es construir el "G.fst" convirtiendo este modelo de lenguaje ARPA a un formato FST usando un archivo ejecutable proporcionado por la herramienta kaldi llamado "arpa2fst".

```
echo
echo "===== MAKING G.fst ====="
echo
lang=data/lang
arpa2fst --disambig-symbol=#0
```

Phones: Adicionalmente a todos los archivos anteriores, el directorio de lenguaje tiene una carpeta llamada "phones", que contiene diferentes archivos acerca del conjunto de fonemas. La mayoría de estos archivos contiene la misma información pero con diferentes versiones, tal como, ".txt",".int" y ".cls". Otros ejemplos son "disambig.txt", "disambig.int" y "disambig.cls".

3.3.4. Preparación de archivos de configuración

Los archivos de configuración relacionados al sistema de reconocimiento de voz son muy útiles y buenos hábitos para mantener algunos parámetros invariables.

Para los archivos de configuración se necesita crear la carpeta llamada “conf” dentro del directorio "kaldi/egs/quechua". Dentro de esa carpeta se crean estos dos archivos.

mfcc.conf: Este archivo de texto contiene información relevante para el proceso de extracción de características, tales como la frecuencia de muestreo que es usado en los archivos de audio.

```
--use-energy=false
--sample-frequency=20000
```

decode.config: Este archivo configurable para la operación de reconocimiento.

```
first_beam=10.0
beam=13.0
lattice_beam=6.0
```

Otros archivos que tienen mucha importancia a la hora de configurar son los archivos cmd.sh y path.sh.

cmd.sh: Este archivo define el modo de ejecución tanto en el entrenamiento como en la decodificación del sistema, que puede ser una ejecución simple o en paralelo.

```
#export train_cmd="queue.pl --mem 2G"
#export decode_cmd="queue.pl --mem 4G"
#export mkgraph_cmd="queue.pl --mem 8G"

export train_cmd="run.pl"
export decode_cmd="run.pl"
```

path.sh: Este archivo define las rutas completas de las distintas librerías y directorios principales para ser agregadas al variable de entorno.

```
export KALDI_ROOT=`pwd`/../..
[ -f $KALDI_ROOT/tools/env.sh ] && . $KALDI_ROOT/tools/env.sh
export PATH=$PWD/utils/:$KALDI_ROOT/tools/openfst/bin:$PWD:$PATH
[ ! -f $KALDI_ROOT/tools/config/common_path.sh ] && echo >&2 "The standard
file $KALDI_ROOT/tools/config/common_path.sh is not present -> Exit!" && exit
1
. $KALDI_ROOT/tools/config/common_path.sh
export LC_ALL=C
```

Muestra del archivo env.sh

```
export LIBLBFGS=/home/franklin/Escritorio/kaldi/tools/liblbfgs-1.10
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:-
}:${LIBLBFGS}/lib/.libs
export SRILM=/home/franklin/Escritorio/kaldi/tools/srilm
export PATH=${PATH}:${SRILM}/bin:${SRILM}/bin/i686-m64
```

3.3.5. Extracción de características

Después de preparar todos los archivos necesarios para el sistema de reconocimiento de voz, inicia el proceso de extracción de características que en esta tesis está basada en la extracción de los coeficientes ceptrales de la frecuencia de Mel (MFCCs) como características acústicas de las señales de voz, la media ceptral y Estadísticas de normalización de varianzas (CMVN) también son aplicadas como parte de la extracción de características. Las características MFCC pueden ser extraídos usando el script "make_mfcc.sh", en tanto las estadísticas de CMVN pueden ser calculados usando el script "compute_cmvn_stats.sh". Ambos archivos de scripts de la herramienta kaldi están localizadas en el directorio "kaldi/egs/quechua/steps" y son llamados desde el script principal run.sh.

```
# Making feats.scp files
nj=8
mfccdir=mfcc
# utils/validate_data_dir.sh data/train
# script for checking if prepared data is all right
utils/fix_data_dir.sh data/train
utils/fix_data_dir.sh data/test
# tool for data sorting if something goes wrong above
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/train exp/make_mfcc/train $mfccdir
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/test exp/make_mfcc/test $mfccdir
# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir
```

3.3.6. Entrenamiento del modelo de lenguaje

Luego de construir el corpus de texto con una buena cantidad de frases se procede a realizar el proceso de entrenamiento basado en el modelo n-grams con n=3, siendo el modelo 3-grams un estándar dentro del procesamiento de lenguaje natural (NLP).

El proceso de entrenamiento requiere como parámetro de entrada el archivo "corpus.txt" y el tamaño del n-gram (3-grams). El proceso de entrenamiento se realiza mediante el siguiente script.

```
local=data/local
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt -wbdiscount -
text $local/corpus.txt -lm $local/tmp/lm.arpa
```

El resultado obtenido del proceso de entrenamiento del modelo de lenguaje es el archivo "vocab-full.txt" y "lm.arpa", donde el primero almacena una lista de todas las palabras diferentes encontradas en el corpus de texto (similar al lexicón.txt) y el segundo es el resultado probabilístico de las palabras empezando de 1-gram hasta n-grams, siendo n la cantidad de grams elegidas de acuerdo a la cantidad de datos almacenados en nuestro corpus de texto.

El archivo "lm.arpa" contiene el siguiente resultado como parte del entrenamiento del modelo de lenguaje.

```
ngram 1=17068
ngram 2=43496
ngram 3=3433

\1-grams:
-1.095025       </s>
-99     <s>     -0.4164996
-4.583505       ACHHUYAMUQKUCHU       -0.3008826
-4.583505       ACHHUYKUSQA     -0.2986951
-4.583505       ACHHUYRUSQA     -0.3002985
-4.583505       ACHHUYUSQA      -0.3008826
-3.770591       AHINA           -0.6017482
-4.407413       AHINAKAQTINQA   -0.474906
-4.282475       AHINALLATAQ     -0.3977643
-4.185565       AHINALLATAQMI   -0.3622854
-4.282475       AHINAMAN  -0.361545
-4.583505       AHINAMANMI      -0.301013
…
\2-grams:
-2.307496       HATUN ALLIN
-2.307496       HATUN ALQOKUNA
-2.006466       HATUN ATIYNINKUNATAWAN      0
-2.307496       HATUN ATIYNINMANTA
-2.307496       HATUN ATIYNINTAN
-2.006466       HATUN ATIYNIYOQ
-2.006466       HATUN ATIYNIYOQMI
-1.830375       HATUN ATIYWAN   -0.30103
-2.307496       HATUN CHALLWA
-2.307496       HATUN CHALLWAQ
-2.307496       HATUN HALLP'APIPAS
-2.307496       HATUN HATUN
-2.307496       HATUN K'ANCHAQKUNATA
…
\3-grams:
-0.4771213      P'UNCHAY CHAY WASIPI
-0.30103        PAYMI CHAY LLAQTAMAN
-0.1760913      PAYPAS CHAY SUYUPI
-0.1760913      PAYPIWAN CHAY LLAQTAMAN
-0.8239087      PAYQA CHAY LLAQTAMAN
-0.1760913      QESPICHIWANANCHISPAQ CHAY WIÑAY
-0.1760913      QHARIQA CHAY WAÑUQQA
-0.39794        QHAWARISPATAQ CHAY RUNATA
-0.1760913      QOYKULLARQANTAQ CHAY T'ANTAKUNATAQA
-0.09691001     RAYMIPAQ CHAY LLAQTAMAN
-0.39794        TUKUY CHAY HAMUQKUNAMANTA
-0.1760913      UMANCHARQANKUCHU CHAY NISQAN
-0.1760913      WIKCH'UYKUNQACHU CHAY YARWASHAQMAN
-0.1760913      YUYANKICHISCHU CHAY RUNAKUNAQA
-0.1760913      TIEMPO CHAYAMUNANKAMA NISPA
…
```

3.3.7. Entrenamiento y alineamiento del modelo acústico

Luego de construir el directorio "/kaldi/egs/quechua", preparar los archivos necesarios para el modelo acústico y el modelo de lenguaje, así como la extracción de características para la señales de voz. El siguiente paso y quizá el más importante es el proceso de entrenamiento del modelo acústico junto al alineamiento del audio a su transcripción textual. El siguiente proceso lo realizamos aplicando diferentes métodos de entrenamiento para obtener resultados más razonables y modelos más robustos. Cada uno de los procesos de entrenamiento es seguido por un proceso de alineamiento, dado que los métodos más avanzados de entrenamiento como el modelo DNN-HMM dependen de los valores de alineamiento de un proceso de entrenamiento previo.

El proceso de entrenamiento se realiza mediante el uso de scripts proporcionados por la herramienta Kaldi, que requieren algunos argumentos necesarios para el inicio del proceso de entrenamiento. Estos argumentos son las siguientes:

- El directorio de entrenamiento "data/train".
- El directorio de los datos del lenguaje "data/lang", la cual contiene todo los archivos relacionados al modelo de lenguaje.
- El directorio destino "exp/modelo-actual", la cual contiene los resultados del modelo actual.
- El directo de alineamiento "exp/modelo-alineado", la cual en un inicio requiere como argumento el modelo actual entrenado y contiene los archivos relacionados al alineamiento de los audios al modelo entrenado.

Los métodos de entrenamiento aplicados en esta tesis son los siguientes:

3.3.7.1. Entrenamiento monophone (un fonema)

Este es el primer método de entrenamiento que es usado para entrenar el modelo acústico. Este modelo no utiliza ninguna información contextual de fonemas anteriores y posteriores. Este modelo es entrenado de forma plana usando MFCC, delta, y las características de aceleración (delta-delta) que luego pueden ser utilizados en el modelo *triphone*. Entrenar el modelo *monophone* puede ser hecho usando el script en "train_mono.sh", mostrado como lo siguiente.

```
#steps/train_mono.sh [options] <training-data-dir> <lang-dir> <exp-dir>"
echo
echo "===== MONO TRAINING ====="
echo
steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang
exp/mono || exit 1
```

Como se puede ver el script "steps/train_mono.sh" necesita como entrada el conjunto de datos dentro de "data/train" y los datos dentro de la carpeta "data/lang". Como salida de este proceso se obtiene resultados dentro del directorio "exp/mono".

Luego del entrenamiento se realiza el proceso de alineamiento mediante el script "align_si.sh", mostrado con el siguiente.

```
echo
echo "===== MONO ALIGNMENT ====="
echo
steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono
exp/mono_ali || exit 1
```

Del script de alineamiento previo podemos ver que, hace uso de del modelo entrenado y el resultado del alineamiento es almacenado en el directorio "exp/mono_ali".

3.3.7.2. Entrenamiento triphone (tres fonemas)

El modelo *triphone* representa un fonema contextual donde en cada fonema está representada por el fonema anterior a ella, el fonema real, y el fonema posterior a ella. Entrenar el modelo *triphone* puede ser hecho usando el script "steps/train_deltas.sh". La entrada para este script son el directorio "data/train", "data/lang" y el alineamiento previo "exp/mono_ali". Muy a parte de estos requisitos el entrenamiento *triphone* requiere también de algunos argumentos tales como el número de estados en el árbol de decisión del Modelo Oculto de Markov y el número de Gaussianos. Y tiene el siguiente formato.

```
echo
echo "=========TRI1 (first triphone pass) TRAINING ======="
echo
steps/train_deltas.sh --cmd "$train_cmd" $hojas $gaussianas data/train
data/lang
exp/mono_ali exp/tri1 || exit 1
```

Como se puede ver el script "steps/train_deltas.sh" necesita como entrada el conjunto de datos dentro de "data/train" y los datos dentro de la carpeta "data/lang" y alineamiento previo "exp/mono_ali". Como salida de este proceso se obtiene resultados dentro del directorio "exp/tri1". Donde el valor de $hojas representa el número de estados HMM y el valor de $gaussianas representa el número total de Gaussianos.

EL proceso de alineamiento para el modelo *triphone* es como sigue:

```
echo "==============align for the tri1==============="
echo
steps/align_si.sh --nj $nj --cmd "$train_cmd" \
--use-graphs true data/train data/lang exp/tri1 exp/tri1_ali
```

Del script de alineamiento previo podemos ver que, hace uso de del modelo entrenado y el resultado del alineamiento *triphone* es almacenado en el directorio "exp/tr1_ali".

3.3.7.3. Entrenamiento Red Neuronal Profunda (DNN)

El modelo principal del conversor de voz a texto está basado en una Red Neuronal Profunda (DNN) junto al modelo oculto de Markov (HMM), para ello es necesario el alineamiento del modelo *triphone*, dado que la red neuronal profunda realiza el trabajo de clasificación de fonemas a partir de características acústicas. El entrenamiento del modelo DNN-HMM se realiza mediante el script "steps/nnet2/train_simple.sh" que proporciona la herramienta Kaldi. El modelo DNN-HMM requiere de los datos de entrenamiento y alineamiento anterior localizados en las carpetas "data/train", "data/lang" y "exp/tri1_ali", y como salida los archivos se almacenan en el directorio "exp/nnet2/nnet2_simple".

```
steps/nnet2/train_simple.sh \
      --stage -10 \
      --num-threads "$num_threads" \
      --feat-type raw \
      --splice-width 4 \
      --lda_dim 65 \
      --num-hidden-layers 3 \
      --hidden-layer-dim 512 \
      --add-layers-period 3 \
      --num-epochs 10 \
      --iters-per-epoch 1 \
      --initial-learning-rate 0.01 \
      --final-learning-rate 0.001 \
      --minibatch-size "$minibatch_size" \
      data/train \
      data/lang \
      exp/tri1_ali \
      $experiment_dir \
      || exit 1;
```

Donde además para una Red Neuronal Profunda es necesario algunos argumentos tales como: número de capas ocultas, numero de neuronas en cada capa ocultas, numero de épocas, la cantidad de iteraciones por época, tasa de aprendizaje inicial, la tasa de aprendizaje final y el tamaño de *minibatch*.

3.4. Proceso de decodificación

3.4.1. Decodificación monophone

Luego de entrenar el modelo *monophone* usando el script "train_mono.sh", se realiza la decodificación de los archivos de voz, todo este proceso es necesario para ver la precisión del sistema de reconocimiento de voz. El proceso de decodificación requiere el grafo decodificado el cual puede ser creado usando el script "mkgraph". El proceso de decodificación se realiza mediante el script “steps/decode.sh” proporcionada por la herramienta Kaldi, que como valores de entrada necesita el grafo representativo del modelo de lenguaje que es obtenida mediante el script “utils/mkgraph.sh” y los datos del directorio “data/test”. Los valores obtenidos del proceso de decodificación se almacenan en el directorio “exp/mono/decode”.

```
echo
echo "===== MONO DECODING ====="
echo

utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd "$decode_cmd"
exp/mono/graph data/test exp/mono/decode
```

3.4.2. Decodificación triphone

Luego de entrenar el modelo *triphone* usando el script "train_deltas.sh", se realiza la decodificación de los archivos de voz, todo este proceso es necesario para ver la precisión del sistema de reconocimiento de voz mediante el modelo GMM-HMM. El proceso de decodificación requiere el grafo decodificado el cual puede ser creado usando el script "mkgraph". El proceso de decodificación se realiza mediante el script “steps/decode.sh” proporcionada por la herramienta Kaldi, que como valores de entrada necesita el grafo representativo del modelo de lenguaje que es obtenida mediante el script “utils/mkgraph.sh” y los datos del directorio “data/test”. Los valores obtenidos del proceso de decodificación se almacenan en el directorio “exp/tri1/decode”.

```
echo
echo "===== TRI1 (first triphone pass) DECODING ====="
echo

utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd "$decode_cmd"
exp/tri1/graph data/test exp/tri1/decode
```

3.4.3. Decodificación DNN

Luego de entrenar el modelo DNN usando el script "/steps/nnet2/train_simple.sh", se realiza la decodificación de los archivos de voz, todo este proceso es necesario para ver la precisión del sistema de reconocimiento de voz mediante el modelo DNN-HMM. El proceso de decodificación requiere el grafo decodificado el cual puede ser creado usando el script "mkgraph". El proceso de decodificación se realiza mediante el script "steps/nnet2/decode_simple.sh" proporcionada por la herramienta Kaldi, que como valores de entrada necesita el grafo representativo del modelo de lenguaje que es obtenida mediante el script "utils/mkgraph.sh" y los datos del directorio "data/test". Los valores obtenidos del proceso de decodificando se almacenan en el directorio "exp/nnet2/nnet2_simple/decode".

```
steps/nnet2/decode_simple.sh \
      --num-threads "$num_threads" \
      --beam 8 \
      --max-active 500 \
      --lattice-beam 3 \
      exp/mono/graph \
      data/test \
      $experiment_dir/final.mdl \
      $unknown_phone \
      $silence_phone \
      $experiment_dir/decode \
      || exit 1;
```

3.5. Desarrollo de interfaz del conversor de voz a texto

Luego de conocer la gran mejora que puede aportar el modelo DNN-HMM logrando una precisión por palabra de 59.20% en comparación a la tasa de error de palabra obtenido por el modelo GMM-HMM de 56.59%, dentro de un sistema de reconocimiento de voz (conversor de voz a texto). Se inicia con el desarrollo de una interfaz amigable para mostrar el funcionamiento de nuestro modelo DNN-HMM junto al modelo de lenguaje.

El diseño del interfaz está basada en la librería Tkinter, la cual es una librería estándar para el desarrollo de interfaces (GUI) en el lenguaje de programación Python. La interfaz contiene un botón interactivo dentro de la ventana principal, la cual está representada por el ícono de un micrófono y es utilizado para el proceso de grabación de voz, mediante la librería PortAudio que captura el audio en formato digital. Además se encarga de almacenar el archivo de audio en la dirección "quechua/quechua_audio/test/21/01/" como el archivo "21-01-000.wav".

Como producto de la decodificación se genera el archivo "/quechua/exp/nnet2/nnet2_simple/decode/log/decode.1.log" que contiene aparte de datos relevantes de la decodificación, la oración reconocida para el archivo de audio dado, como un paso final se recoge la información del archivo "decode.1.log" y se obtiene el texto del archivo de voz "21-01-000" y se muestra en la interfaz del conversor de voz a texto mediante una componente "Label".

La interfaz del conversor de voz a texto tiene la siguiente apariencia.

Figura 20: Interfaz de usuario para el conversor de voz a texto.

Fuente: Elaboración Propia

El código utilizado para los dos procesos primordiales a la hora del reconocimiento de voz se muestra a continuación:

Para la interacción entre los datos de Kaldi y la interfaz del conversor de voz, implementamos la clase "Audio_grabado.py" en el lenguaje de programación Python 2.7 y dentro de la misma creamos dos módulos: uno para el grabado de audio y el otro para la obtención de texto.

CAPITULO IV

ANALISIS E INTERPRETACION DE RESULTADOS

Con el objetivo de lograr un conversor de voz a texto con una precisión cercano a 60 %, se realizó distintos experimentos, teniendo que variar el tamaño del corpus de voz. Así también optimizar el conversor de voz a texto haciendo la búsqueda de los parámetros óptimos de la Red Neuronal profunda dentro del modelo acústico.

Los procesos de entrenamiento y reconocimiento son usados para las distintas combinaciones de valores en los parámetros, teniendo en cuenta el tamaño del corpus de voz, haciendo una distribución del corpus de voz de 90% para la fase de entrenamiento y 10% para la fase de reconocimiento, siendo esta distribución la más utiliza dentro de los sistemas ASR.

- **Elección de número de archivos de entrenamiento**

Para empezar, se hace un análisis de la cantidad de datos que es necesario para alcanzar una precisión cercana al 60%, es así que se procedió a realizar un conjunto de experimentos, con el objetivo de encontrar la cantidad de archivos de audio que sean capaces de alcanzar dicha precisión.

Para ello, se realiza un incremento en la cantidad de horas de audio en cada experimento. Los experimentos fueron realizados con el modelo *monophone* para las siguientes cantidades: 4.53, 9.85, 12.24, 14.67 y 18.35 horas de audio.

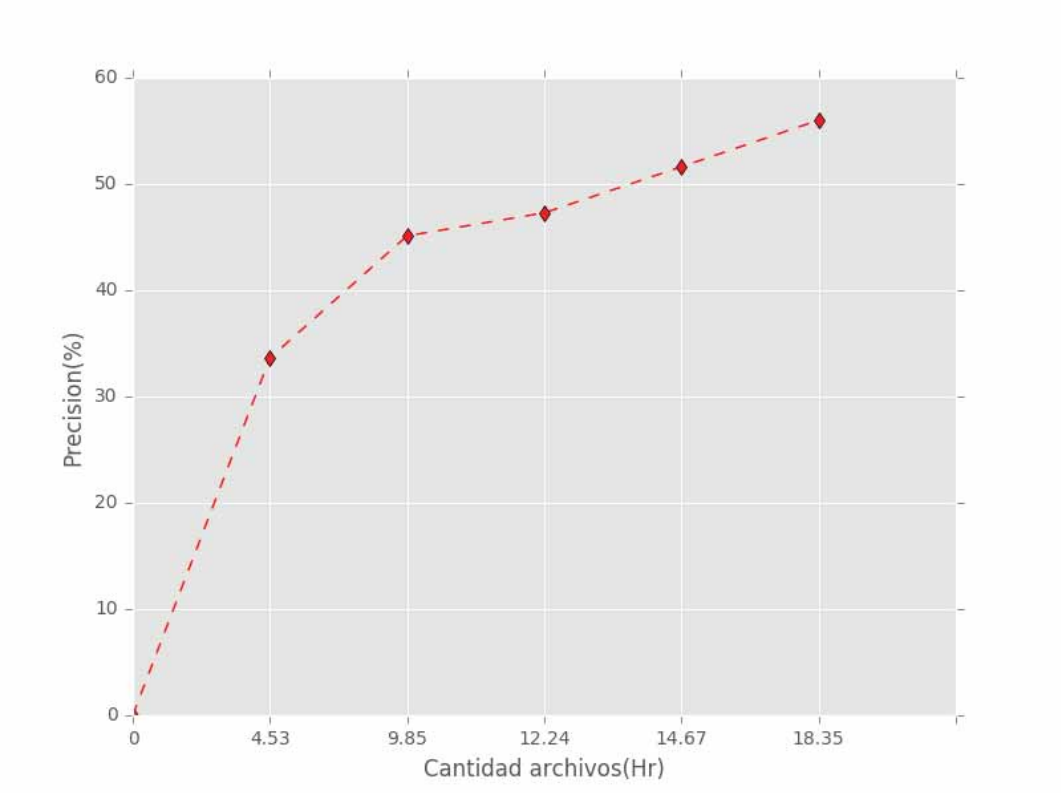


Figura 21: Tasa de precisión para distintos tamaños de corpus de voz.

Fuente: Elaboración propia.

Con relación a la precisión en el reconocimiento de voz, se aprecia que a medida que incrementamos el tamaño del corpus de voz, la precisión de reconocimiento también se incrementa de forma logarítmica, esto nos sugiere a la vez que hay posibilidad de incrementar el corpus de voz para lograr mejores resultados, desafortunadamente no se dispone de más archivos para el entrenamiento, debido a las limitaciones de recursos textuales en el idioma Quechua.

Llegando así a obtener una precisión de 56.00% con un corpus de voz de 18.35 horas, con un vocabulario de 17486, y con un corpus de texto de 6475 frases.

- **Elección de número óptimo de hojas y gaussianas para el modelo *triphone***

Un punto de partida para el modelo DNN-HMM, es el modelo GMM-HMM que contiene una representación contextual en los fonemas (*triphone)*, que es esencial para el entrenamiento supervisado de las DNN.

Este experimento se basó en entrenar los HMM, incrementando el número de hojas del árbol de decisión y el número total de gaussianas en cada experimento, con la intención de obtener mejores resultados que el modelo base *monophone*.

En la Tabla 6, se muestran los resultados de reconocimiento para los distintos experimentos en el modelo *triphone*.

# Hojas **#Gaussianas**	138 12420	138 13110	456 41046	456 43320	512 46080	512 48640
WACC (%)	55.49	55.17	55.92	**56.59**	56.20	55.95

Tabla 6: Resultados de precisión de los distintos modelos triphone con delta.

Fuente: Elaboración propia.

Teniendo en cuenta que la representación fonética contextual en los estados del modelo oculto de Markov modela de mejor manera el habla continua, podemos ver que una representación fonética *triphone* obtiene una mayor precisión en comparación al modelo base *monophone*, logrando así una precisión de 56.59%.

- **Elección de número óptimo de capas y nodos internos para el modelo DNN**

Los parámetros más importantes y variantes dentro del entrenamiento de una red neuronal profunda son la cantidad de capas internas, cantidad de nodos internos y la tasa de aprendizaje, siendo necesario la búsqueda de valores óptimos para dichos parámetros en cada experimento. Por otra parte, los parámetros que no tienen una variación significativa tales como el tamaño de *batch* y el número de épocas quedan con valores por defecto.

Este experimento, primeramente determina la tasa de aprendizaje y está basada en el valor que utiliza el propio conjunto de herramientas "Kaldi" de acuerdo al tamaño del corpus con la que se trabaja, teniendo así una tasa de aprendizaje de 0.001 para sistemas ASR con un gran corpus (100-1000 horas) y una tasa de aprendizaje de 0.002 para sistemas ASR con un corpus

regularmente pequeño (10-100 horas). Situándonos en este último con una tasa de aprendizaje igual a 0.002.

Como segunda paso y el más laborioso fue el entrenamiento de los distintos modelos DNN, con las distintas configuraciones en la cantidad de capas y nodos internos que tenga la mayor precisión a la hora de clasificar un fonema.

Para cada modelo generado, se realiza un proceso de reconocimiento para un conjunto de archivos de audio (342), que cuenta con 2813 palabras para determinar la precisión por palabra, tal como se muestra en la Tabla 7.

# Capas ocultas	# Nodos internos	WACC (%)
2	128	56.46
	256	57.06
	512	57.84
	1024	55.96
3	128	57.15
	256	58.01
	512	**59.20**
	1024	57.02
4	128	56.16
	256	57.23
	512	55.64
	1024	54.87

Tabla 7: Resultados de precisión de las distintas configuraciones en DNN.

Fuente: Elaboración propia.

Para una mejor interpretación de los resultados, se presenta el grafico de barras para las distintas configuraciones de la Red Neuronal Profunda.

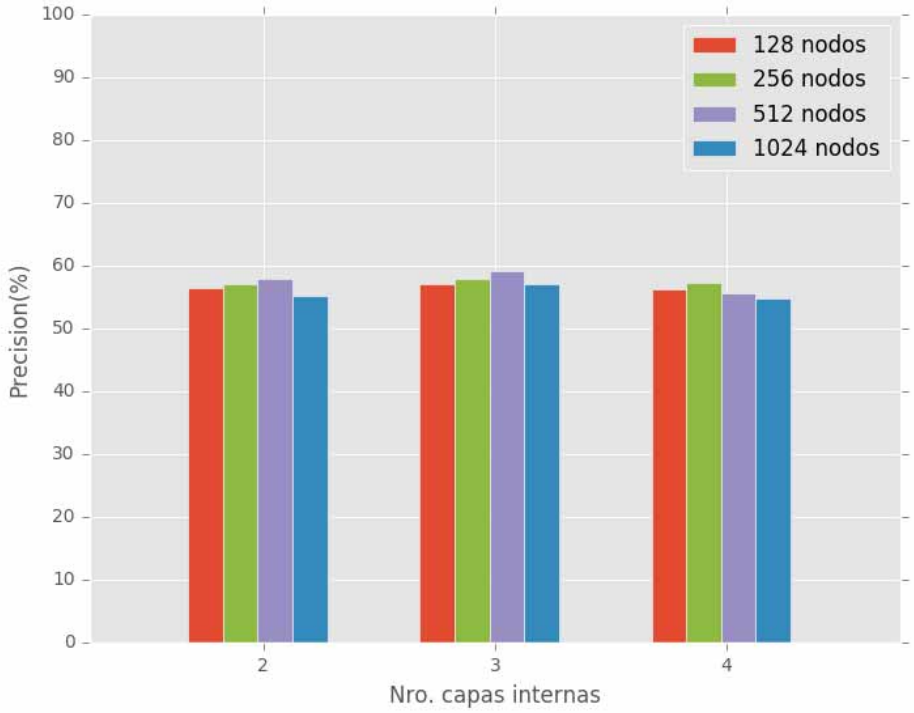


Figura 22: Tasa de precisión de las distintas configuraciones DNN.

Fuente: Elaboración propia.

En el grafico anterior, se puede ver que la mejor precisión se obtiene con la configuración de número de capas igual a 3 y el número de nodos igual a 512, obteniendo una precisión de 59.20%. Analizando el resto de las arquitecturas podemos ver que con una configuración poco compleja de 2 capas y 128 nodos internos, se obtiene una precisión de 56.46%, una precisión inferior al conseguido con 3 capas y 512 nodos, dado que este tipo de configuración es capaz de representar sistemas de reconocimiento de voz con un corpus de voz de unas cuantas horas (3, 4, 5 horas), por otro lado una configuración muy compleja de 4 capas y 1024, se logra una precisión de 54.87%. De este último podemos observar que a mayor complejidad en las configuraciones DNN no siempre se obtiene mejores resultados, dado que las configuraciones más profundas están pensadas para representar sistemas de reconocimiento de voz con un corpus de voz de cientos o miles de horas.

Luego de todos los experimentos realizados se tiene un resumen de la mejor precisión en los distintos modelos*: Monophone, triphone y DNN*.

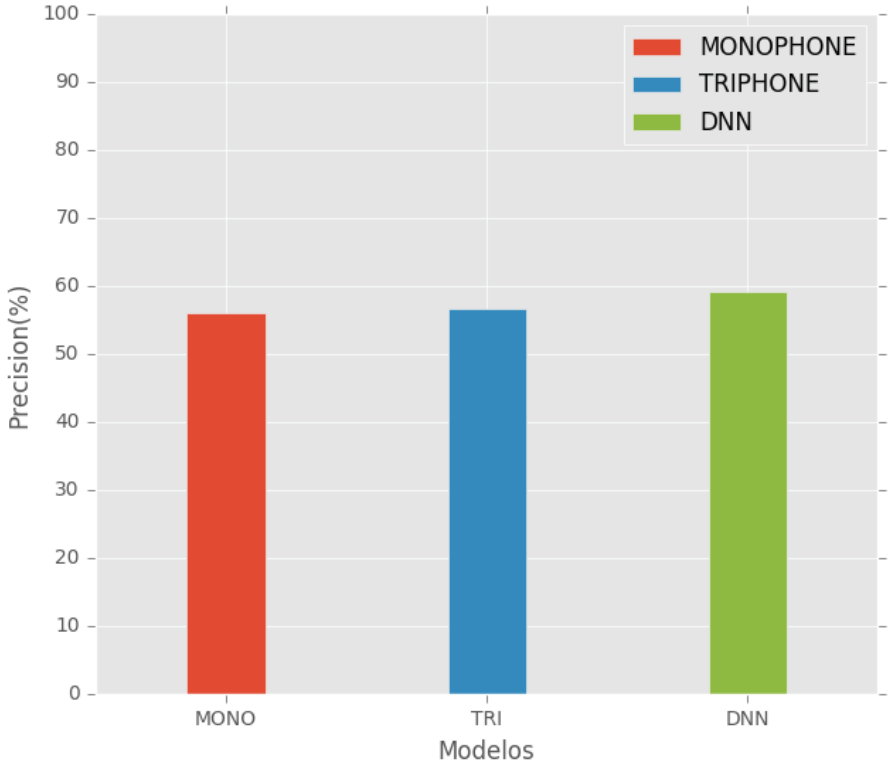


Figura 23: Resumen distintos modelos acústicos.

Fuente: Elaboración propia.

De la Figura 23, podemos observar que el modelo DNN mejora sustancialmente a los modelos anteriores (*monophone*, *triphone*), llegando a obtener una precisión de 59.20%, obteniendo así una mejora del 2.61% en comparación al modelo *triphone* que logra una precisión de 56.59% y una mejora de 3.2% en comparación al modelo *monophone* que logra una precisión de 56.00%.

CONCLUSIONES

1. El corpus de voz es el recurso más importante dentro de un conversor de voz a texto para el habla continua con un gran vocabulario e independiente del locutor, por lo tanto, para su construcción, se debe tener en cuenta algunas características para ser considerado un buen corpus de voz: la cantidad de personas distintas, el balanceo de género y la cantidad de audios. Es así que se obtuvo un corpus de voz con una cantidad de 32 personas distribuidas en 60% hombres - 40% mujeres y la cantidad de audios con un total de 6475 frases equivalente a más de 18 horas de grabación.
2. La construcción de los distintos recursos de voz necesarios para el proceso de entrenamiento de los modelos acústico y de lenguaje, tales como: diccionario fonético, fonemas, representaciones de silencio, grandes colecciones de texto, género de los locutores, rutas para los audios, mapeo locutor-audio y el mapeo audio-locutor, se realizó satisfactoriamente para el entrenamiento de los modelos acústico y de lenguaje.
3. Se realizó el proceso de entrenamiento para las distintas configuraciones de la Red Neuronal Profunda junto al Modelo oculto de Markov. Llegando a obtener la configuración más óptima, cuando la tasa de aprendizaje es igual a 0.002, la cantidad de capas internas es igual a 3 y el número de nodos internos es igual a 512, dejando con valores por defecto de la propia herramienta Kaldi los parámetros tales como: número de épocas igual a 15, y el número de *minibatch* igual a 128.
4. Luego del entrenamiento del modelo acústico y del modelo de lenguaje haciendo uso de los distintos recursos de voz. Se desarrolla satisfactoriamente el conversor de voz a texto basado en el modelo acústico DNN-HMM y el modelo de lenguaje 3-grams, con una precisión del 59.20%, logrando así el primer sistema de este tipo con una precisión bastante aceptable para el idioma Quechua de la región Cusco.

TRABAJOS A FUTURO

- Incrementar el corpus de voz a un aproximado de 100 horas para una mayor precisión en el reconocimiento de voz basado en DNN-HMM.
- Entrenar Redes Neuronales Recurrentes (LSTM, GRU) dentro del modelo acústico y el modelo de lenguaje.
- Realizar el entrenamiento en procesamiento paralelo, basado en máquinas con Unidades de Procesamiento Grafico (GPUs).
- Implementar aplicaciones finales tales como software de dictado, traductores Quechua-Español y software para el aprendizaje de idioma basado en el conversor de voz a texto para el idioma quechua desarrollado.
- Realizar pruebas de precisión con un mayor porcentaje de datos de prueba dentro del corpus de voz.

BIBLIOGRAFÍA

Academia Mayor de la lengua Quechua (2006). Diccionario Quechua-Español-Quechua, Cusco, Perú, Multiservicios e Imprenta Edmundo Pantigozo EIRL.

Antón, J. (2015). Desarrollo de un sistema de reconocimiento de habla natural independiente del locutor (Tesis de pregrado). Universidad Autónoma de Madrid, España.

Branislav P., Stevan O., Edwin P., Niksa J., & Vlado D. (2015) Deep Neural Network Based Continuous Speech Recognition for Serbian Using the Kaldi Toolkit.https://www.researchgate.net/10.1007/978-3-319-23132-7_23.

Cahuana, R. (2007). Manual de Gramatica Quechua Cusco-Collao. Sicuani, Perú.

Calvo, M. (2010). Desarrollo de un sistema de reconocimiento del habla (Tesis de pregrado). Universidad Politécnica de Valencia, Madrid, España.

Camacho, O. (2016). AudioVisual Systems on Telecomunications Engineering (Tesis de grado).Universidad Politécnica de Cataluña, España.

Castañeda, P. (2012).El Lenguaje Verbal del Niño. Recuperado de http://sisbib.unmsm.edu.pe/bibvirtual/libros/linguistica/leng_ni%C3%B1o/sist_aud_influ_habla.htm. Accedido el 07/07/2019.

Castro, G. & Pinares, A. (2015). Sistema de reconocimiento automático de voz (conversor voz a texto) para el Quechua hablado en el Cusco con CMU SPHINX4 (Tesis de Grado), Universidad Nacional de San Antonio Abad del Cusco, Perú.

Chan, Willian (Diciembre, 2016). End-to-End Speech Recognition Models (Tesina), Carnegie Mellon Unversity, USA.

Dan Lim (Octubre, 2017). Convolutional Attention-based Seq2Seq Neural Network for End-to-End ASR (Tesis de Maestria), Department of Computer Science and Engineering, Korea University Graduate School, Korea.

Dimitri Palaz (Julio, 2016). Towards End-to-End Speech Recognition (Tesis doctoral), Facultad de ciencias tecnológicas e Ingeniería, Suiza.

Fan, Y., Potok, M. & Shroba, C. (Abril, 2017). Deep Learning for Audio. Recuperado de http://slazebni.cs.illinois.edu/spring17/lec26_audio.pdf. Accedido el 07/072019.

Fan,Y., Potok,M., y Shroba C.(2017). Deep Learning for Audio. Recuperado de http://slazebni.cs.illinois.edu/spring17/lec26_audio.pdf. Accedido el 07/072019.

Fontelles, M. J., Simões, M. G., Farias, S. H., & Fontelles, R. G. S. (2009). Metodologia da pesquisa científica: diretrizes para a elaboração de um protocolo de pesquisa. Revista Paraense de Medicina, 23(3), 1-8.

Gaida, C., Lange, P., Petrick, R., et al (2014), Comparing Open-source Speech Recognition Toolkits, DHBW, Stuttgart, Germany.

Habeeb, S. (2018).Digital Automatic Speech Recognition using Kaldi (tesis de pregado). Florida Institute of Technology, Melbourne, Florida.

Hammami N. (2014).Contribution to the Automatic Speech Recognition of Arabic Language And its Apllications (tesis de Doctor).Universite Badji Mokhtar-Annaba, Arabie Saoudite.

Hancco, N. (2005).Diccionario "SIMI PIRWA". Bartolomé de las C., Cusco, Perú.

Hernandez, C., Meza, I. & Herrera, J. (2017) Automatic speech recognizers for Mexican Spanish and its open resources. Laboratorio de Tecnologías del lenguaje (LTL), Universidad Nacional Autónoma de México (UNAM), México, 15(2017) 259-270.

Kipyatkova, I. & Karpov, A. (2016). DNN-Based Acustic Modeling for Russian Speech Recognition Using Kaldi. https://www.researchgate.net/10.1007/978-3-319-43958-7_29.

Liu, X. (2017).Deep Convolutional and LSTM Neural Networks for Acoustic Modelling in Automatic Speech Recognition. Pearson Education Inc. USA.

Meyer, J. (2016). How to train a Deep Neural Net Acoustic Model with Kaldi.USA: Josh Meyer's Website. Recuperado de http://jrmeyer.github.io/asr/2016/12/15/DNN-AM-Kaldi.html. Accedido el 07/072019.

Mohamed, M., Odile, M., Dominique, F., Denis, J., David, L., & Kamel, S. (2017). An enhanced automatic speech recognition system for Arabic. Loria, Campus Scientifique, BP 239, 54506 Vandoeuvre Les-Nancy, France.

Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.

Murillo, W. (2008). La investigación científica. Recuperado el 11 de enero de 2018 de https://www.monografias.com/trabajos15/invest-cientifica/invest-cientifica.shtml. Accedido el 07/072019.

Nacereddine, H. (2014). Contribution to the Automatic Speech Recognition of Arabic Language and its Applications (tesis doctoral). Badji Mokhtar-Annaba University, Algeria.

Povey, D., Ghoshal, A., Boulianne, G., et al (2012).The Kaldi Speech Recognition Toolkit. Microsoft Research, USA.

Pellegrini, T., Trancoso, I., Hämäläinen, A., Calado, A., Dias, M. S., & Braga, D. (2012). Impact of age in ASR for the elderly: preliminary experiments in European Portuguese. In Advances in Speech and Language Technologies for Iberian Languages (pp. 139-147). Springer, Berlin, Heidelberg.

Ravanelli, M. (2017). Deep Learning for Distant Speech Recognition (Tesis Doctoral), Department of Information Engineering and Computer Science, University Degli Studi di Trento, Italia.

Rosillo, V. (2016). Automatic Speech recognition with Kaldi toolkit (Tesis de grado). Universidad Politécnica de Cataluña, España.

Russell, M., & D'Arcy, S. (2007). Challenges for computer recognition of children's speech. In Workshop on Speech and Language Technology in Education.

Saksamudre, S., Shishrimal, P. & Deshmukh, R. (2015). A review on different approaches for Speech recognition System. International Journal of Computer Applications, 115(22), 975-8887.

Sociedad Biblica Peruana (2004). Santa Biblia Quechua Cusco-Diospa simin qelqa, Lima, Perú.

Torres, B. & Ferran, G. (Barcelona 2008). Sistema Fonatorio. Recuperado de https://vozplena.wordpress.com/2015/07/28/sistema-fonatorio-proceso-fisiologico-para-la-produccion-de-la-voz-parte-2/. Accedido el 07/072019.

Trost, M. (2015). Convolutional Neural Networks CNNs. Recuperado de http://recognize-speech.com/acoustic-model/knn/comparing-different-architectures/convolutional-neural-networks-cnns. Accedido el 07/072019.

Tsiartas, A. (2006). Automatic speech recognition using soft feature decoding (Tesis de pregrado). Technical University of Create, Chania 731 00, Grecia.

Velasco, S. & Martín, A. (2008).Creación de Scripts en Linux. Recuperado de https://webs.ucm.es/info/aulasun/archivos/SCRIPTS.pdf. Accedido el 07/072019.

Zielinski, W. (2016). Creating a simple ASR System in Kaldi toolkit from scratch using small digits corpora. Recuperado de http://www.dsp.agh.edu.pl/_media/pl:dydaktyka:kaldi_for_dummies.pdf. Accedido el 07/072019.

ANEXOS

A. Instalación de la herramienta de reconocimiento de voz Kaldi en Linux

El proceso detallado de instalación de la herramienta kaldi junto a todas sus bibliotecas necesarias para el proceso de entrenamiento y reconocimiento de voz, en el sistema operativo LINUX 16.04 LTS. Detalles que fueron extraídos de la fuente oficial de Kaldi en el repositorio GitHub.

Especificaciones Hadware y Software

La maquinaria utilizada fue una computadora portátil TOSHIBA, con procesador de 2.40GHz, 8 GB de memoria RAM y disco duro de 600 GB.

El sistema operativo es UBUNTU 16.04 LTS de 64-bits, sistema operativo recomendado por desarrolladores de la herramienta Kaldi por mostrar menor cantidad de errores en el proceso de instalación y uso de la herramienta.

Proceso de instalación

Como paso previo a la instalación, realizamos el proceso de clonación del repositorio oficial de kaldi en GitHub hacia nuestra computadora portátil, mediante el siguiente comando.

```
josh@yoga:~/Desktop$ git clone https://github.com/kaldi-asr/kaldi.git
Cloning into 'kaldi'...
remote: Counting objects: 63320, done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 63320 (delta 5), reused 0 (delta 0), pack-reused 63298
Receiving objects: 100% (63320/63320), 74.94 MiB | 8.26 MiB/s, done.
Resolving deltas: 100% (49427/49427), done.
Checking connectivity... done.
```

Echando un vistazo dentro del directorio kaldi, vemos lo siguiente:

```
josh@yoga:~/Desktop$ cd kaldi/
josh@yoga:~/Desktop/kaldi$ la
COPYING  .git          .gitignore  misc      src    .travis.yml
egs      .gitattributes  INSTALL     README.md  tools  windows
```

Para el proceso de instalación se recomienda hacer uso del archivo de instalación adherente en la herramienta kaldi llamada “INSTALL”, donde nos dice que el proceso se realiza en dos pasos, que son las siguientes:

- Primeramente, ir al directorio “kaldi/tools” y seguir las instrucciones del archivo “INSTALL”.

To check the prerequisites for Kaldi, first run

  extras/check_dependencies.sh

and see if there are any system-level installations you need to do. Check the output carefully. There are some things that will make your life a lot easier if you fix them at this stage. If your system default C++ compiler is not supported, you can do the check with another compiler by setting the CXX environment variable, e.g.

  CXX=g++-4.8 extras/check_dependencies.sh

Then run
  make

which by default will install ATLAS headers, OpenFst, SCTK and sph2pipe. OpenFst requires a relatively recent C++ compiler with C++11 support, e.g. g++ >= 4.7, Apple clang >= 5.0 or LLVM clang >= 3.3. If your system default compiler does not have adequate support for C++11, you can specify a C++11 compliant compiler as a command argument, e.g.

  make CXX=g++-4.8

If you have multiple CPUs and want to speed things up, you can do a parallel build by supplying the "-j" option to make, e.g. to use 4 CPUs

  make -j 4

In extras/, there are also various scripts to install extra bits and pieces that are used by individual example scripts. If an example script needs you to run one of those scripts, it will tell you what to do.

- Finalmente, ir al directorio “kaldi/src” y seguir las instrucciones del archivo “INSTALL”

These instructions are valid for UNIX-like systems (these steps have been run on various Linux distributions; Darwin; Cygwin). For native Windows compilation, see ../windows/INSTALL.
You must first have completed the installation steps in ../tools/INSTALL (compiling OpenFst; getting ATLAS and CLAPACK headers).

The installation instructions are

  ./configure --shared
  make depend -j 8
  make -j 8

Note that we added the "-j 8" to run in parallel because "make" takes a long time. 8 jobs might be too many for a laptop or small desktop machine with not many cores.

B. Código bash completo para el proceso de entrenamiento y prueba del sistema de reconocimiento de voz.

El código fuente completo del conversor de voz a texto para el idioma quechua está localizada en el repositorio de *github*: https://github.com/franklin0714/quechuaASR.

```
#!/bin/bash
. ./cmd.sh || exit 1
. ./path.sh || exit 1
nj=4
# number of parallel jobs - 1 is perfect for such a small data set

lm_order=4
# language model order (n-gram quantity) - 1 is enough for digits
grammar

# Safety mechanism (possible running this script with modified
arguments)
. utils/parse_options.sh || exit 1
[[ $# -ge 1 ]] && { echo "Wrong arguments!"; exit 1; }

# Removing previously created data (from last run.sh execution)
 sudo rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp
data/train/feats.scp data/train/split1 data/test/spk2utt
data/test/cmvn.scp data/test/feats.scp data/test/split1
data/local/lang data/lang data/local/tmp
data/local/dict/lexiconp.txt

echo
echo "===== PREPARING ACOUSTIC DATA ====="
echo
#Needs to be prepared by hand (or using self written scripts):
#
#spk2gender     [<speaker-id> <gender>]
#wav.scp   [<uterranceID> <full_path_to_audio_file>]
#text      [<uterranceID> <text_transcription>]
#utt2spk   [<uterranceID> <speakerID>]
#corpus.txt     [<text_transcription>]

# Making spk2utt files
utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt
echo
echo "===== FEATURES EXTRACTION ====="
echo
# Making feats.scp files
mfccdir=mfcc
# utils/validate_data_dir.sh data/train
# script for checking if prepared data is all right
utils/fix_data_dir.sh data/train
utils/fix_data_dir.sh data/test
# tool for data sorting if something goes wrong above
```

```
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/train
exp/make_mfcc/train $mfccdir
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/test
exp/make_mfcc/test $mfccdir
# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir


echo
echo "===== PREPARING LANGUAGE DATA ====="
echo

# Needs to be prepared by hand (or using self written scripts):
#
# lexicon.txt                   [<word> <phone 1> <phone 2> ...]
# nonsilence_phones.txt              [<phone>]
# silence_phones.txt            [<phone>]
# optional_silence.txt               [<phone>]

# Preparing language data
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang
data/lang


echo
echo "===== LANGUAGE MODEL CREATION ====="
echo "===== MAKING lm.arpa ====="
echo

loc=`which ngram-count`;
if [ -z $loc ]; then
     if uname -a | grep 64 >/dev/null; then
          echo "entro al primero++++++++++++++++++++++++++++++++"
          sdir=$KALDI_ROOT/tools/srilm/bin/i686-m64
     else
          sdir=$KALDI_ROOT/tools/srilm/bin/i686
     fi
     if [ -f $sdir/ngram-count ]; then
          echo "entro al segundo++++++++++++++++++++++++"
          echo "Using SRILM language modelling tool from $sdir"
          export PATH=$PATH:$sdir
     else
          echo "SRILM toolkit is probably not installed.
Instructions: tools/install_srilm.sh"
          exit 1
     fi
fi
local=data/local
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt
-wbdiscount -text $local/corpus.txt -lm $local/tmp/lm.arpa
```

```
echo
echo "===== MAKING G.fst ====="
echo

lang=data/lang
cat $local/tmp/lm.arpa | arpa2fst - | fstprint |
utils/eps2disambig.pl | utils/s2eps.pl | fstcompile --
isymbols=$lang/words.txt --osymbols=$lang/words.txt --
keep_isymbols=false --keep_osymbols=false | fstrmepsilon |
fstarcsort --sort_type=ilabel > $lang/G.fst

echo
echo "===== MONO TRAINING ====="
echo

steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang
exp/mono  || exit 1

echo
echo "===== MONO DECODING ====="
echo

utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd
"$decode_cmd" exp/mono/graph data/test exp/mono/decode

echo
echo "===== MONO ALIGNMENT ====="
echo

steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang
exp/mono exp/mono_ali || exit 1

echo
echo "===== TRI1 (first triphone pass) TRAINING ====="
echo

steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train
data/lang exp/mono_ali exp/tri1 || exit 1

echo
echo "===== TRI1 (first triphone pass) DECODING ====="
echo

utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd
"$decode_cmd" exp/tri1/graph data/test exp/tri1/decode
```

C. Código fuente de la interfaz grafica

```
import pyaudio
import wave
import numpy as np
import scipy.io.wavfile
import os

class Audio_grabado:
	ruta_principal="/home/franklin/Escritorio/kaldi/egs/quechua"
	def __init__(self,*args):
		self.FORMAT = pyaudio.paInt16
		self.CHANNELS = 1
		self.RATE = 16000
		self.CHUNK = 1024
		self.RECORD_SECONDS = 6
		self.WAVE_OUTPUT_FILENAME =
ruta_principal+"/quechua_audio/test/21/01/21-01-0000.wav"
		self.ARCHIVO =
ruta_principal+"/experiment/nnet2/nnet2_simple/decode/log/decode.1.log"

	def grabar(self):
		audio = pyaudio.PyAudio()
		# start Recording
		stream = audio.open(format=self.FORMAT, channels=self.CHANNELS,
				rate=self.RATE, input=True,
				frames_per_buffer=self.CHUNK)
		print "recording..."
		frames = []

		for i in range(0, int(self.RATE / self.CHUNK * self.RECORD_SECONDS)):
		   data = stream.read(self.CHUNK)
		   frames.append(data)
		print "finished recording"

		# stop Recording
		stream.stop_stream()
		stream.close()
		audio.terminate()


		#scipy.io.wavfile.write(self.WAVE_OUTPUT_FILENAME,self.RATE,frames)
		waveFile = wave.open(self.WAVE_OUTPUT_FILENAME, 'wb')
		waveFile.setnchannels(self.CHANNELS)
		waveFile.setsampwidth(audio.get_sample_size(self.FORMAT))
		waveFile.setframerate(self.RATE)
		waveFile.writeframes(b''.join(frames))
		waveFile.close()
```

```
#importando libreria para desarrollo de interfaz
from Tkinter import *
from PIL import Image, ImageTk
from Audio_grabado import *

#funciones
class interfaz_real:
def __init__(self,*args):
self.texto="aver"

def Enviar():
audio1=Audio_grabado()
grabado=audio1.grabar()

def Obtener():
audio2=Audio_grabado()
texto=audio2.obtener_texto()
Texto=Label(text=texto,font=("verdana",10)).place(x=250,y=413)

# creando una ventana nueva
ventana=Frame(height=400,width=600)
ventana.pack(padx=20,pady=20)
Texto=Label(text="Conversor de voz a texto en Quechua Cusco-
Collao",font=("verdana",15)).place(x=5,y=5)
load = Image.open("audio_img.jpg")
render = ImageTk.PhotoImage(load)
Imagen_label=Label(image=render).place(x=0,y=50)
# creando botones para interaccion con la clase Audio_grabado
boton_grabar=Button(ventana,command=Enviar,text="Empezar
Grabacion",font=("verdana",13),background="blue").place(x=0,y=360)
boton_reconocer=Button(ventana,command=Obtener,text="Obtener
Texto",font=("verdana",13),background="red").place(x=200,y=360)
Texto=Label(text="Texto reconocida:",font=("verdana",15)).place(x=20,y=410)
ventana.mainloop()#fin de ventana
```

D. Muestra parcial de textos reconocidos por el conversor de voz texto

TEXTO REAL DE LA GRABACIÓN	TEXTO RECONOCIDO
HINAQTINMI PAYQA AYLLUNTA LLIW PAYWAN KAQKUNATAPAS NIRQAN HAP'ISQAYKICHISTA WIKCH'UYCHIS HINASPA CH'UYANCHAKUYCHIS HUK P'ACHAWAN	HINAQTINMI PAYQA AYLLUNTA LLIW PAYWAN KAQKUNATAPAS NIRQAN HAP'ISQAYKI CHIQA WIKCH'UYCHISCHU HINASPA CH'UYANCHAKUYNIN HUK P'ACHAWAN
WICHALLASUNÑA CHAYPIN ALTARTA PERQASAQ PAYQA LLAKIKUSQAY P'UNCHAYMI RIMAPAYAWARQAN HINALLATAQ MAYPIÑA KAQTIYPAS PURIYSIWARQAN	WICHAY AS UÑA CHAYPIN ALTARTA KASAQ PAYQA LLAKIKUSPA AY P'UNCHAYMI RIMAPAYWAYCHU HINALLATAQ MAYPIÑA KAQ PIPAS PURISHARQAN
AHINAPIN PAYMAN QORQANKU HAP'ISQANKU LLAPAN CHAYKUNATA PAKARQAN SACH'A CHAKIPI	KAQ AHINAPIN PAYMAN QORQANKU AYQESQANKU LLAPAN CHAYKUNATA WAQARQAN SACH'A CHAKIPI
LLOQSIQTINKUTAQ PAYQA MANCHACHIRQAN MUYURIQNINKUPI KAQ LLAQTAKUNATA CHAYMI CHURINKUNATAQA MANA QATIYKACHARQANKUCHU	LLOQSIQTINTAQ PAYQA MANCHACHIRQAN MUYURIQNIN KUTI KAQ LLAQTAKUNATA CHAYMI CHURINKUNA TAQA MANA QATIKACHASQANKU
PAYMI PURIYSIMUQNIN LLAPA RUNAKUNAPIWAN CHAYARQAN LLAQTAMAN	PAYMI PURIY SEÑOR HUKNIN LLAPAN RUNAKUNAPIWAN CHAYARQAN LLAQTAMAN
HINASPANMI ALTARTA CHAYPI PERQARQAN WAWQENMANTA AYQEKUSHAQTIN	HINASPAN NIRQAN KARPA CHAYPIPAS KARQAN WAYQEN UMANTA KASHAQTIN
UYWAQEN WAÑUKAPURQAN P'AMPASQATAQ KARQAN SACH'A CHAKIPI CHAYMI CHAY SACH'AQA WAQAY SUTICHASQA KARQAN	HUCHU WAKIN WAÑUKAPURQAN P'AMPASQA KAQ KARQAN SACH'A CHAKIPI CHAYMI CHAY SACH'AQA WAQAY SUTICHASQA KARQAN
HUKTAWANMI RIKHURIRQAN ARAMMANTA KUTIMPUQTIN HINASPAN PAYTA SAMINCHARQAN	HUKTAWANMI RIKHURIRQAN HARANMANTA KUTIMPUQTINTAQ HINASPAN PAYTA SAMINCHARQAN
NIRQANTAQ SUTIYKIQA MANAN SUTIYKIÑACHU KANQA ASWANPAS ISRAELMI SUTIYKIQA KAPUNQA NISPA AHINATAN ISRAELTA SUTICHAPURQAN	NIRQANTAQ SUTIYKI QAN MANAN SUTIYKI ÑA CHUÑU KANQA ASWANPAS ISRAEL NI SUTIYKITA KAPUNQA NISPA AHINATAN ISRAELPA SUTICHAKUSPA
HINASPAN NIRQAN ÑOQAN KANI TUKUY ATIYNIYOQ ASKHA LLAQTAKUNAN QANMANTA PAQARINQA HINALLATAQ REYKUNAPAS QANMANTA PAQARINQA	HINASPAN NIRQAN ÑOQAN KANI TUKUY ATIYNIYOQ ASKHA LLAQTAKUNAN QANMANTAN PAQARINQA HINALLATAQ REYKUNAPAS QANMANTAN PAQARINQA
QOSQAY HALLP'ATAN QANMAN QOSQAYKI HINALLATAQ MIRAYNIYKIMANPAS QOSAQ CHAY HALLP'ATA NISPA	QOSQAY HALLP'ATAN QANMAN QOSQAYKI HINALLATAQ MIRAYNIYKIMANPAS QOSAQ CHAY HALLP'ATA NISPA
HINASPAN PAYWAN RIMASQAN CHEQASMANTA RIPURQAN	HINASPAN PAYWAN RIMASQAN CHEQASMAN TARPURQAN

Tabla 8: Resultado textual conversor de voz a texto.

Fuente: Elaboración propia