UNIVERSIDAD POLITÉCNICA DE VALÉNCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Transcripción asistida de audio.

Proyecto Final de Carrera - Ingeniería Informática

Juan Daniel Valor Miró

Supervisado por:
Dr. Jorge Civera Saiz
Dr. Alfons Juan Ciscar

10 de septiembre de 2012

Gracias a los que me han ayudado
en el transcurso de este proyecto.

Índice general

Capítulo 1

Introducción

1.1. Motivación del Proyecto

En la Sociedad de la Información en la que estamos inmersos, es mucha la información que podemos encontrar en muchos medios de comunicación digitales o analógicos. Tradicionalmente, esta información se ha transmitido en un medio escrito, llegando así al alcance de la mayoría de la población existente; sin embargo actualmente esta tendencia está cambiando, tomando especial relevancia otro tipo de medios de transmisión como el audiovisual.

De esta forma, este medio nacido en el siglo XIX, ha ido tomando relevancia en muchos ámbitos de la vida diaria como la educación [Per02], el arte [Mag05], el turismo [Mar10], el cine [Med02], etc. Sin embargo, y a pesar de las muchas ventajas que este medio aporta a la transmisión de información, existen claras desventajas de este medio respecto a los medios clásicos como el escrito.

En primer lugar, se trata de un medio que requiere de tecnología avanzada para poder ser utilizado (algún dispositivo con capacidad de reproducción de ficheros multimedia). Además, es muy dificil encontrar en este medio algún segmento o frase con las técnicas clásicas de computación; de forma que si buscamos una parte en concreto de, por ejemplo, una conferencia, probablemente deberemos escucharla completamente para localizar dicho fragmento. Por otro lado, este medio se vuelve muy ineficiente para personas con algún tipo de discapacidad acustica [dA11], ya que gran cantidad de la información se transmite mediante el sonido. Destacar por último, que la traducción de un medio audiovisual es mucho más compleja que la de un medio escrito, límitando así su expansión geográfica.

Así pues, una solución ampliamente utilizada es la de disponer de las transcripciones escritas del contenido sonoro del medio audiovisual, incrustadas en el propio video en forma de subtitulos. Así pues, de esta forma se soluciona con relativa facilidad todos los problemas mencionados previamente, al pasar a trabajar con un medio puramente escrito.

Sin embargo, el proceso de transcripción de un medio audiovisual suele ser muy costoso en tiempo y recursos [RG05], ya que se trata de un proceso manual. Así pues,

para realizar la transcripción de un medio audiovisual ámplio se requiere de mucho personal, tiempo, y recursos que no siempre se pueden obtener.

Así pues, de la mano del *Reconocimiento de Formas* [DH73] se plantea la solución a este problema mediante técnicas de *Reconocimiento Automático del Habla Continua*. De esta forma a partir de un medio audiovisual, de forma automática y en base a unos modelos previamente generados, se puede extraer la transcripción con una tasa de error relativamente baja. Además, para mejorar más aún esta tasa de error, pueden utilizarse técnicas interactivas de transcripción asistida de audio, que pregunte al transcriptor únicamente aquellos fragmentos de audio en los que crea que la transcripción automática ha podido fallar.

1.2. Reconocimiento de Formas

El Reconocimiento de Formas es una rama de la Informática cuyo objetivo es la percepción de objetos por un sistema, entendiendo por percepción de objetos la capacidad de un sistema de darle un significado semántico a un conjunto de datos mediante su etiquetado o clasificación [Seg92]. Un sistema de Reconocimiento de Formas básico consta de los siguiente modulos:

1. Preproceso: Adquiere la señal desde un sensor externo o de alguna base de datos de información externa al sistema, la filtra eliminando el ruido y la prepara para ser procesada por el computador.

2. Extracción de Características: Obtiene una serie de medidas, de la señal procesada por el modulo previo, que son relevantes para llevar a cabo el proceso de etiquetado o clasificación en la etapa posterior.

3. Clasificación: A partir de un modelo debidamente entrenado y de las características extraidas en el modulo anterior, utiliza técnicas para asociar al objeto su significado semántico.

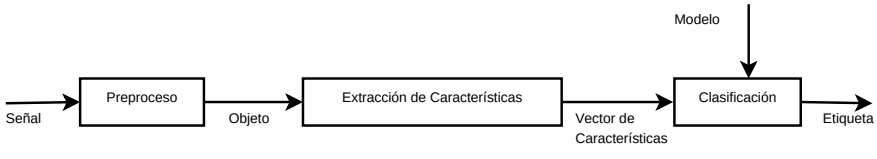

Figura 1.1: Sistema Básico de Reconocimiento de Formas

Como se puede observar en la figura 1.1, para llevar a cabo el proceso de clasificación nos hace falta un modelo, que se obtiene mediante el uso de técnicas estadísticas de aprendizaje sobre un conjunto de datos de ejemplos ya clasificados o etiquetados previamente de forma manual y correcta.

Para hacernos una idea un poco más aproximada de como funciona de forma básica un sistema de Reconocimiento de Formas desde el punto de vista estadístico;

supongamos que E es un espacio de representación de las muestras a etiquetar, y $\Omega = \{\omega_1, \omega_2, \omega_3, ..., \omega_c\}$ un conjunto de C etiquetas que pueden ser asignadas a cada una de las muestras $x \in E$ del total de muestras E.

Asi pues, si conocemos la proporción de muestras de la clase ω_i respecto al total, o lo que es lo mismo la probabilidad a priori $P(\omega_i)$; y la probabilidad de que se de un x concreto sabiendo de antemano ω_i, o lo que es lo mismo la probabilidad condicional $P(x|\omega_i)$; mediante la Regla de Bayes podemos estimar la probabilidad de que dicha x pertenezca a ω_i, o lo que es lo mismo la probabilidad a posteriori $P(\omega_i|x)$ [DH73].

$$P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{\sum_{j=1}^{C} P(x|\omega_j)P(\omega_j)} \tag{1.1}$$

Sin embargo, en la mayoría de aplicaciones de Reconocimiento de Formas, es muy dificil estimar adecuadamente $P(\omega_i)$ y $P(x|\omega_i)$, por lo que la mayoría de veces se opta por utilizar técnicas de aprendizaje que los estiman a partir de un conjunto de muestras de entrenamiento ya clasificadas.

1.3. Reconocimiento del Habla Continua

La rama que nos interesa del Reconocimiento de Formas es la que estudia el Reconocimiento del Habla Continua, y que es considerada como una rama cuya tarea es muy compleja [Ser05] debido a todos los requerimientos implicitos de la misma.

Los inicios de esta rama está en los años 50 en los laboratorios Bell, donde consiguieron realizar un reconocimiento de dígitos aislados monolocutor. Al mismo tiempo, en la *University England College* crearon el primer reconocedor fonético, y en el *MIT Lincoln Lab* un reconocedor de vocales independiente del hablante. En los años 60, se consiguió la Alineación Dinámica en el Tiempo en Vintsyuk (la Unión Sovietica), y un primer intento de reconocimiento del habla continua en el *Carnegie Mellon University*.

En los años 70, el mundo de la estadística llegó al Reconocimiento del Habla Continua, permitiendo por primera vez un sistema de reconocimiento de palabras aisladas que funcionara muy bien. Además IBM desarrolló varios proyectos de reconocimiento de grandes vocabularios, y la *Carnegie Mellon University* desarrolló el primer sistema de reconocimiento del habla con éxito.

A partir de los años 80, se produce una explosión de los métodos estadísticos, cuyo éxito se debió a los Modelos Ocultos de Markov. Además el area de las Redes Neuronales se empezó a incluir en diversas partes del reconocimiento de voz, mejorando su índice de éxito en el reconocimiento. Además nació el sistema de reconocimiento del habla SPHINX [HCS] considerado como uno de los mejores reconocedores del habla que existen hoy en día.

Finalmente, desde los años 90 hasta la actualidad se han conseguido diversas aplicaciones como sistemas de dictado, integración en aplicaciones telefónicas, integración en el Sistema Operativo, aparición del estándar VoiceXML, etc [RG06]. Además la aparición del HTK Hidden Markov Model Toolkit [YEG+06] en el año 1989 de la mano de *Cambridge University Engineering Department* fue un gran avance para el Reconocimiento del Habla Continua.

Un esquema básico de Reconocimiento del Habla Continua es el que podemos observar en la figura 1.2, que consta de los tres procesos básicos del Reconocimiento de Formas, concretados en el caso del habla continua.

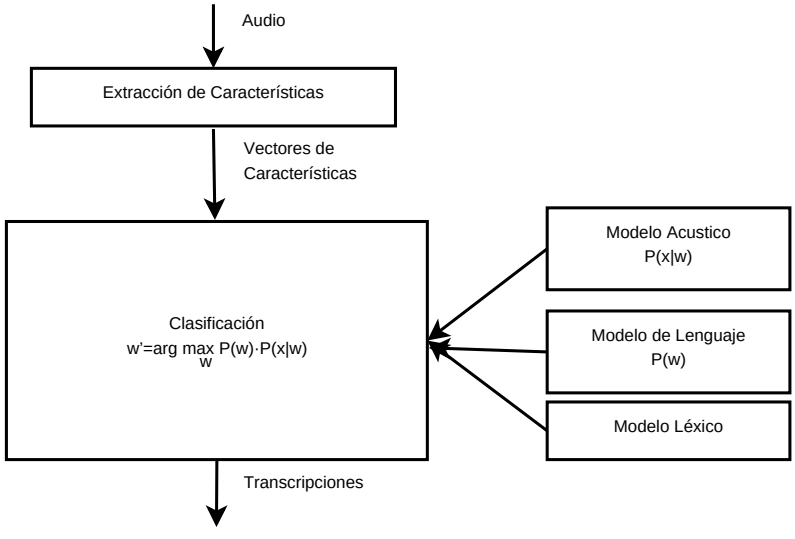

Figura 1.2: Sistema Básico de Reconocimiento del Habla Continua

Desde el punto de vista estadístico podemos definir este problema de una forma un poco más formal basandonos en la Regla de Bayes [DH73]. Supongamos que dado un segmento de audio x de un conjunto X, deseamos encontrar la frase w que lo transcribe extraida de un vocabulario de destino W; o lo que es lo mismo la transcripción más probable para w' según la siguiente expresión:

$$\hat{w} = \underset{w}{\operatorname{argmax}} P(w|x) \tag{1.2}$$

En otras palabras, buscamos la transcripción w de todas las posibles transcripciones de W que maximice el valor de $P(w|x)$, que es la transcripción w' más probable de entre todas las posibles. Sin embargo, $P(w|x)$ no puede ser estimada directamente, sino que debe ser estimada mediante $P(w)$ y $P(x|w)$ según la siguiente expresión extraida mediante la Regla de Bayes:

$$\hat{w} = \underset{w}{\operatorname{argmax}} P(w|x) = \underset{w}{\operatorname{argmax}} \frac{P(w)P(x|w)}{P(x)} = \underset{w}{\operatorname{argmax}} P(w)P(x|w) \tag{1.3}$$

Estas dos probabilidades vienen estimadas por modelos previamente entrenados con un conjunto de segmentos de audio con sus respectivas transcripciones. Concretamente se utilizan dos modelos: el de lenguage (con un modelo léxico como parte importante del mismo) y el acústico. Así pues, es fundamental conocer la forma correcta de estimar estos dos modelos mediante un conjunto de muestras de entrenamiento.

1.4. Extracción de Características

La extracción de características de los segmentos de audio (ver la figura 1.3) se realiza utilizando los coeficientes cepstrales en las frecuencias de Mel, también conocidos como MFCC, que se basan en la representación del habla según la percepción auditiva humana [RJ]. Los MFCC modela la respuesta auditiva humana en el computador, de forma que con esto logramos un procesado de datos más eficiente. La forma de calcular estos coeficientes es la siguiente:

1. Se hace la transformada de Fourier del segmento de audio.
2. Mapear el espectro de audio a la escala especificada por Mel (más ajustada al oido humano) utilizando una función de ventana triangular.
3. Calcular el logaritmo de la energía de cada frecuencia Mel.
4. Tomar la transformada de coseno discreta de la lista de los logaritmos calculados en el paso previo como si fuera una señal de audio.
5. Las amplitudes del espectro resultante conforman los MFCC.

El problema fundamental de los MFCC es que no son muy robustos ante la presencia de ruido aditivo, y aunque existen algunas soluciones que suavizan el problema (como aumentar las amplitudes de los logaritmos en el cálculo en un factor de 2 o 3 [TW05]), no se acaba de solucionar dicho problema.

Sin embargo, se trata de uno de los sistemas de extracción de características más avanzado de entre todos los que existen hoy en día; ya que permite adaptar el audio genérico a la percepción humana, y después extraer valores númericos discretos con los que el ordenador es capaz de trabajar muchísimo mejor que con una señal continua.

1.5. Entrenamiento de los Modelos

Los modelos que se utilizan en la fase de clasificación del Reconocimiento del Habla continua (como podemos ver en la figura 1.2), son fundamentalmente el *Modelo de Lenguaje* [CG98] y el *Modelo Acustico* [Jel98]. Estos modelos se estiman a partir de un conjunto de entrenamiento, que se trata en este caso de un grupo de audios transcritos de forma manual. Estos audios, con sus respectivas transcripciones, siguen el proceso básico que podemos observar en la figura 1.3, que ilustra de forma sencilla el proceso que se lleva a cabo para generar los Modelos.

Como podemos observar en la figura 1.3, lo primero que se realiza para el caso del audio es la extracción de características (MFCC), que son exactamente los mismos procesos que se siguen en el sistema de Reconocimiento del Habla Continua planteado en la figura 1.2.

Por otro lado, el preproceso que se le aplica a las transcripciones, determina los resultados que se van a obtener posteriormente. Por ejemplo, un preproceso que elimine los acentos, los signos de puntuación y coloque en mayusculas todas las letras; generará un modelo de lenguaje que cuando se utilice para reconocer un segmento

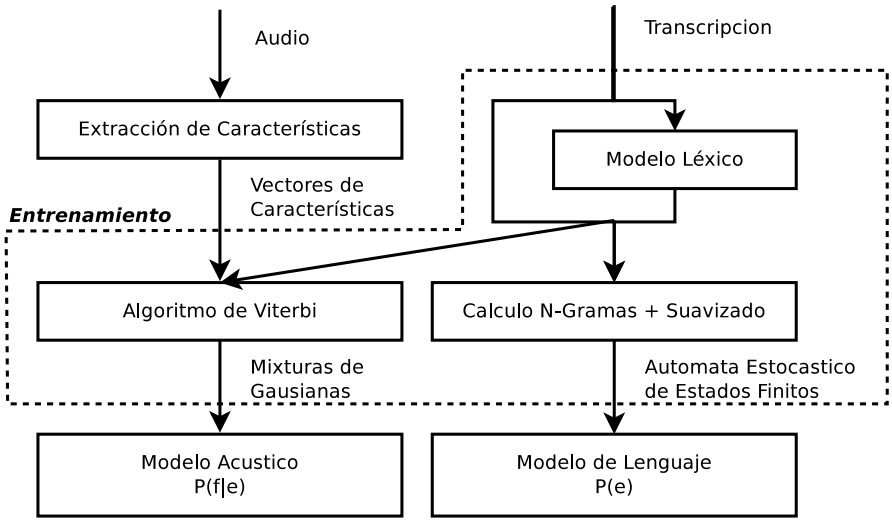

Figura 1.3: Proceso de Generación de los Modelos

de audio, genere como resultado un texto en mayusculas, sin acentos, y sin signos de puntuación.

El Modelo de Lenguaje es en realidad lo que se conoce de forma técnica como un Modelo de Lenguaje Basado en N-Gramas (ver sección 1.5.3) al que se le aplican algunas técnicas de suavizado para mejorarlo, mientras que el Modelo Acustico está basado a su vez en lo que se conoce como Modelos de Markov de Capa Oculta (ver sección 1.5.1), que se entrenan mediante el conocido Algoritmo de Viterbi [Vit67].

Una vez tenemos estos modelos debidamente entrenados, se puede aplicar con éxito la fase de reconocimiento del habla continua mostrada en la figura 1.2.

1.5.1. Modelos de Markov de Capa Oculta

Para calcular un Modelo Acustico se utilizan los llamados Modelos de Markov de Capa Oculta o también llamados HMMs desde el punto de vista continuo. Formalmente un HMM Continuo M es una máquina de estados finitos [YEG+06] definida por la séxtupla (Q, I, F, X, a, b) donde:

1. Q es un conjunto finito de estados que incluye a I y a F.
2. I es el estado inicial del HMMs que está incluido en Q.
3. F es el estado final del HMMs que está incluido en Q.
4. X es un espacio real multidimensional de vectores x.
5. a es una función de distribución de probabilidad (rango 0-1) de transición entre dos estados q_i, q_j.

6. b es una función de densidad de probabilidad de emitir un vector $x \in X$ en un estado $q_i \in Q$.

Para modelar las probabilidades de emisión de los vectores $x \in X$ de cada uno de los estados del HMM Continuo se utiliza una mixtura de densidades de Gausianas. Destacar que para entrenar a y b se utiliza el algoritmo de Viterbi (existen otros algoritmos como Backward o Forward [Vit67], pero no se utilizan en el presente proyecto), que responde a la siguiente expresión:

$$vit_j(t) = \begin{cases} a_{0j} b_j(x_1) & \text{si } t = 1 \\ \left[\max_{i \in [1,N-1]} vit_i(t-1) a_{ij}\right] b_j(x_t) & \text{si } 1 < t \leq T \end{cases} \tag{1.4}$$

Hay que tener en cuenta que T es el número de vectores de la secuencia, $t \in T$ el número del vector $x \in X$ actual, y N el número de estados del modelo excluyendo el inicial (o lo que es lo mismo $N = |Q| - 1$). Además destacar que el orden de complejidad temporal es $O\{|Q^2| \times T\}$ pero esta se reduce a $O\{|Q| \times T\}$ si la topología del modelo M es de izquierda-derecha.

En resumen, un Modelo de Markov de Capa Oculta desde el punto de vista continuo es una máquina de estados finitos cuyas transiciones son distribuciones de probabilidad gausianas, y los estados emiten vectores de reales según una densidad de probabilidad asociada. Estas dos probabilidades se estiman mediante el algoritmo de Viterbi adaptado al punto de vista continuo, utilizando los vectores de características resultantes de la fase de extracción de características (ver la figura 1.3).

1.5.2. Modelo Léxico

El modelo léxico es la base del modelo de lenguaje (ver la sección 1.5.3), y determina las palabras que finalmente pueden llegar a ser reconocidas correctamente en el proceso de Reconocimiento del Habla Continua. El Modelo Léxico contiene las siguientes definiciones:

1. Una lista de todos los fonemas que aparecerán más adelante. Cada uno de los fonemas representa un sonido fonético diferente (así pues la r debil se representa como r y la r fuerte como @, por ejemplo).

2. Una lista de todas las palabras que han aparecido en las transcripciones de entrenamiento con sus correspondientes (una o más, si hubiera) transcripciones fonéticas de las mismas.

Este modelo léxico hace de enlace entre el modelo acústico y el de lenguaje al especificar los sonidos (del modelo acústico) que equivalen a cada palabra (del modelo de lenguaje).

1.5.3. Modelos de Lenguaje de N-Gramas

Los modelos de lenguaje basados en N-Gramas intentan especificar que construcciones sintácticas de palabras son gramaticalmente correctas, o lo que es lo mismo,

contienen la información para indicar si una posible frase w es gramaticalmente posible (y en el caso de que lo sea con que probabilidad se puede dar respecto a otras construcciones gramaticalmente posibles) [CG98].

Para ello, los modelos basados en N-Gramas dividen una frase en grupos de N palabras (o gramas); por lo que son posibles modelos como los Bigramas (2-Gramas), que almacenan las posibles combinaciones de dos palabras que hay en una frase. Al igual que estos podemos encontrarnos Trigramas (3-Gramas) para combinaciones de tres palabras, Cuatrigramas (4-Gramas) para combinaciones de cuatro, etc.

Así pues, desde el punto de vista estadístico, un modelo de N-Gramas estima la probabilidad de que se dé una frase w formada por distintas palabras $w_1, w_2, ..., w_I$:

$$p(w) = \prod_{i=1}^{I} p(w_i|w_1, ..., w_{i-1}) \tag{1.5}$$

Sin embargo, calcular la probabilidad de una frase como el producto de las probabilidades de cada palabra dada su história completa (todas las palabras prévias) es muy costoso, por lo que tradicionalmente se opta por un sistema más reducido que acota la historia a N-1 palabras. Así pues, para un modelo de trigramas la expresión anterior se reduce a la siguiente:

$$p(w) = \prod_{i=1}^{I} p(w_i|w_{i-2}, w_{i-1}) \tag{1.6}$$

Destacar que los modelos de N-Gramas se entrenan por máxima verosimilitud a partir de un corpus de entrenamiento [Jel98]. Así pues, para estimar a partir de este corpus la probabilidad de una palabra w_i dado w_{i-2} y w_{i-1}, se analiza la frecuencia de aparición de w_i después de la secuencia w_{i-2}, w_{i-1} y se normaliza para todos los casos que compartan dicha secuencia.

Sin embargo, los modelos de N-Gramas poseen una desventaja muy grande, y es que pueden asignar una probabilidad de cero a las secuencias de N-Gramas que no tengan en su corpus de entrenamiento, imposibilitando reconocer posibles nuevas frases. Para solucionar este problema se emplean técnicas de suavizado [KN95].

Un ejemplo de estas técnicas es la que se conoce como técnica de suavizado mediante interpolación lineal, que para el caso concreto de los trigramas se basa en la siguiente ecuación:

$$p(w_i|w_{i-2}, w_{i-1}) = \lambda_1 \frac{N(w_{i-2}, w_{i-1}, w_i)}{N(w_{i-2}, w_{i-1})} + \lambda_2 \frac{N(w_{i-1}, w_i)}{N(w_{i-1})} + \lambda_3 \frac{N(w_i)}{N} + \lambda_4 \tag{1.7}$$

Esto es, en primer lugar calculamos la frecuencia de w_i dado w_{i-2} y w_{i-1} normalizada para todas las secuencias de w_{i-2} y w_{i-1} (trigrama). Calculamos también la frecuencia de w_i dado w_{i-1} normalizado para todas las secuencias que comparten w_{i-1} (bigrama), y las veces que aparece w_i en el total de palabras N (unigrama). A estas frecuencias se multiplican por unos coeficientes λ cuya suma es 1, y que priorizará los trigramas, luegos los bigramas, luego el unigrama, y finalmente una pequeño margen para las palabras nuevas [CG98].

Así pues, gracias a esta técnica de suavizado, si en una frase aparece una palabra completamente nueva, a los trigramas que involucren dicha palabra se les asignara la probabilidad λ_4 permitiendo (si el resto de la frase es conocida), que se reconozca adecuadamente.

1.6. Evaluación de los Resultados

Para llevar a cabo la evaluación de las prestaciones de nuestros modelos, se debe utilizar una métrica adecuada a la hora de medir el éxito de la transcripción realizada con unos modelos determinados. Después, básandonos en dicha métrica ya se pueden realizar mejoras en el sistema para obtener cada vez una transcripción más fiable de los videos a analizar.

Así pues, es muy importante fijar una buena métrica, ya que esta va a ser la base de evaluación de nuestro modelos, y de todas las mejoras que les apliquemos. Destacar que es muy importante también realizar esta evaluación en un conjunto de videos de los que poseamos la transcripción, pero que no se hayan utilizado previamente para entrenar los modelos, ya que de ser así se obtendría una medición muy optimista y distorsionada del sistema.

En este caso nos hemos decantado por el *Word Error Rate* más conocido como WER, que se trata de una metrica muy utilizada en el contexto del Reconocimiento de Formas, y que define tres operaciones básicas de edición que se aplican sobre la frase reconocida (respecto a la transcrita de forma manual):

1. Sustitución: La palabra de la transcripción es distita a la reconocida.
2. Inserción: Se ha introducido una palabra que no existe en la transcripción.
3. Borrado: En la transcripción hay una palabra que no ha sido reconocida.

Destacar que por supuesto, el WER se aplica sobre el alineamiento óptimo que lo mínimece según la distáncia de *Levenshtein* [SK83]. Así pues, la expresión final que define el WER es el número de operaciones de sustitución n_s, inserción n_i y borrado n_b partido el número total de palabras que hay (borrados n_b, sustituciones n_s y aciertos n_a):

$$WER = \frac{n_s + n_i + n_b}{n_b + n_s + n_a} \tag{1.8}$$

Es importante saber que el WER no es un porcentaje, ya que si el número de inserciones es muy alto puede llegar a dar un valor superior a 100. Sin embargo se trata de una métrica muy utilizada en el Reconocimiento de Formas, que nos da una aproximación muy buena de los resultados de nuestro sistema de Reconocimiento del Habla Continua.

Capítulo 2

Descripción del Corpus

2.1. Plataforma poliMedia

El corpus que se ha utilizado en el presente proyecto se ha extraido de los videos con licencia libre del repositorio oficial de la *Universidad Politécnica de Valencia* conocido como poliMedia.

Según su descripción propia, poliMedia es un sistema diseñado en la UPV para la creación de contenidos multimedia como apoyo a la docencia presencial, que abarca desde la preparación del material docente hasta la distribución a través de distintos medios (TV, Internet, CD, etc.) a los destinatarios de los mismos (que generalmente son los alumnos de la UPV) [dV12b].

Las características básicas que definen a poliMedia (según su web oficial [dV12b]) son las que se enumeran a continuación:

1. poliMedia es un sistema de producción de materiales educativos de calidad.
2. Es un recurso integrado con todas las herramientas de PoliformaT.
3. Es muy adecuado como apoyo y complemento a la enseñanza presencial.
4. El autor es el propietario intelectual de la obra.
5. Sistema completamente innovador y único, disponible sólo en la UPV.
6. Disponibilidad de instrumentos, materiales y técnicos para el profesorado.
7. Lleva asociado un plan de incentivos económicos.
8. Es fácil: no requiere conocimientos audiovisuales o técnicos.

En resumen, polimedia es una plataforma donde los profesores pueden publicar videos docentes según un modelo estándar y utilizando los medios de la UPV para ello. Además se trata de una plataforma que a día 15 de Febrero de 2012 posee 6830 grabaciones que suman más de 1350 horas de video de más de 1033 profesores diferentes, lo cual le da muchisima relevancia a nivel global.

2.2. El Formato de poliMedia

Los videos docentes que existen en la plataforma poliMedia, siguen un formato estandar en todos los videos, característico de la plataforma. Se trata de una vista conjunta de profesor y pantalla (diapositivas, programas, etc) en un mismo plano de cámara que no se mueve en todo el video.

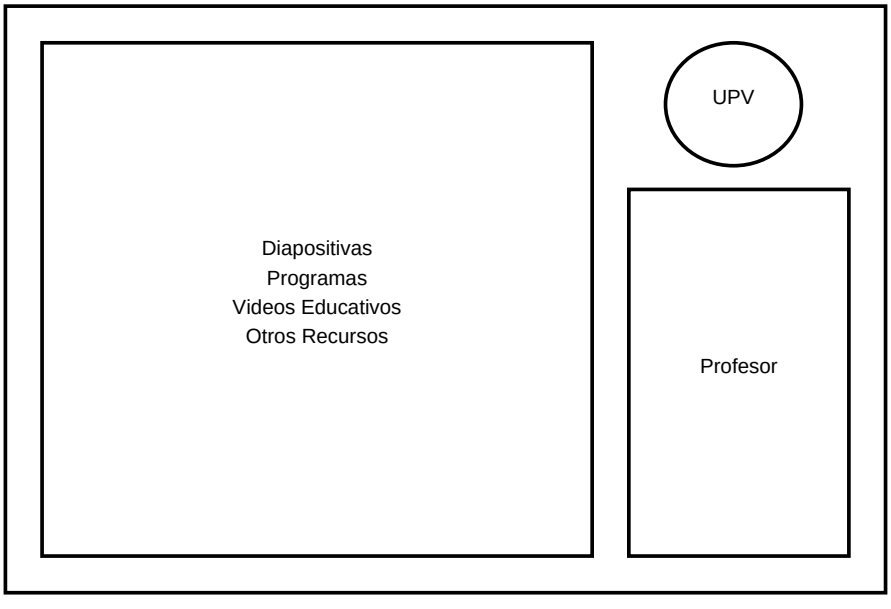

Figura 2.1: Formato de los Videos de poliMedia

El proceso es muy simple: se graba en un estudio al profesor sobre un fondo blanco (audio y video), y simultaneamente a la pantalla del ordenador (que previamente habrá cargado algún tipo de recurso multimedia). Despúes se extrae la grabación del profesor y se mezcla con la de la pantalla según el esquema de la figura 2.1 y se le añade el logo de la UPV, añadiendo el finalmente el audio del profesor al video generado [dV12b].

Este video ya editado finalmente se distribuye a través del portal oficial de poliMedia en formato MOV o MP4 (dependiendo del navegador utilizado para acceder a la plataforma). Sin embargo, para poder trabajar con estos videos, en el presente proyecto se ha optado por eliminar la parte gráfica del video extrayendo del mismo el audio en formato WAV, que se utilizará como entrada para nuestro sistema junto a la transcripción manual del mismo.

Destacar finalmente que la calidad del audio de los videos del polimedia es muy alta, y no posee practicamente ningún tipo de ruido de fondo (ya que ha sido grabado en un estudio cerrado). Sin embargo, los locutores de los videos (más de 1.000 profesores) son de diferentes sexo y edad, lo que hace muy variables los tonos de voz y los ritmos de habla asociados. Además las áreas temáticas de los videos pueden llegar a

ser muy diferentes (con más de 650 areas temáticas bien diferenciadas), y además se tiende a emplear términos muy técnicos que es improbable que se repitan más de tres o cuatro veces en todo el polimedia.

Esto hace del poliMedia un repositorio de audio bueno, pero muy variable en cuanto a acustica y semántica, lo cual representa un grave inconveniente cuando se trabaja con sistemas de Reconcimiento del Habla Continua que se basan en modelos de aprendizaje.

2.3. El Corpus poliMedia

El conjunto utilizado se corresponde con una pequeña parte del total de grabaciones del polimedia, cuyos autores han dejado la licencia completamente libre. En total, hemos extraido 1350 horas de video aproximadamente en 732 grabaciones de diferentes temas y profesores [dV12b].

Tabla 2.1: Comparativa del poliMedia Completo con Nuestro Corpus

Conjunto	Horas	Videos
poliMedia Completo	1350	6830
Nuestro Conjunto	119	732
Porcentaje Total	8,85 %	10,72 %

Dicho corpus ha sido transcrito manualmente al completo por 10 personas, y el autor de este proyecto final de carrera con un total de 832 minutos de video. Para ello se ha utilizado la herramienta de software libre conocida como Transcriber [BGWL00].

2.3.1. Proceso de Transcripción

Para transcribir cada uno de los videos se ha seguido un proceso estándar por todos los miembros del grupo, para que los resultados sean los mismos en todas las transcripciones y de esta manera conseguir un corpus unificado y completo.

En primer lugar, y partiendo del video, se ha utilizado *ffmpeg* para extraer su contenido en formato WAV (una pista de audio única). Una vez extraido el audio, pasamos a iniciar su transcripción con el programa Transcriber, que nos permite realizar dos tareas fundamentales a la hora de llevar a cabo una transcripción:

1. Segmentar: Hay que colocar marcas de tiempo para separar las frases con un significado sintáctico y semántico propio, para poder aplicar el sistema descrito en la sección 1.5.

2. Transcribir: Cada segmento debe poseer en caracteres UTF-8 el contenido de las palabras que se pronuncian en él, siguiendo las reglas que se describen en esta sección.

Además, hay que destacar que se han optado por una serie de convenciones a la hora de realizar dichas transcripciones, con el objetivo de representar adecuadamente las disfluencias orales en el habla. Estas convenciones son las siguientes:

1. Se identificará el profesor que habla en cada video, asi como su sexo.
2. Se segmentará el audio en frases, transcribiendo cada una de ellas.
3. Cuando se produzca una disfluencia, se anotara utilizando la siguiente notación estándar global: /sonido pronunciado/palabra correcta/
4. Cuando se produzca un silencio largo se creará un segmento cuya transcripción será únicamente [sonido de fondo].
5. Cuando se produzca un silencio corto se añadirá la notación /SF//.

Es importante destacar la laboriosa tarea que supone la transcripción un video de forma manual y lo importante que ha sido para la puesta en marcha de este proyecto final de carrera. Por destacar algunos datos, comentar que el RTF (ratio de tiempo de transcripción de audio respecto al tiempo del propio audio) es de una media de 11. Esto implica que por cada minuto de audio se emplean 11 para transcribirlo, lo cual implica que para transcribir todo el corpus utilizado en este proyecto se han empleado aproximadamente 1314 horas.

Después se debe verificar los errores de ortografía cometidos al transcribir rápidamente un video, por lo que utilizaremos la herramienta *aspell* para verificar y corregir dichos errores. Además utilizaremos un pequeño script para comprobar si las expresiones especiales que hemos colocado entre barras están correctas o existe algúna de más o de menos, ya que esto provocaría a la hora de parsear la transcripción numerosos errores.

Tabla 2.2: Ejemplo de una Transcripción de Varios Segmentos

Seg.	Tiempo	Transcripción
1	3,1s	El bucle for /ich/each/ es un bucle que itera una lista.
2	1,9s	Por ejemplo, la variable entera /jota/J/.
3	4,0s	Se trata /e//, de una sentencia realmente útil.
4	5,4s	No olvideis /que que/que/ no existe en /ce/C/.

Como podemos ver en la Tabla 2.2, la transcripción de un fragmento de audio no es trivial. En el segmento 1 podemos observar como se pronuncia una palabra en inglés, y como esta mediante la sintáxis de las barras debe ser especificada primero foneticamente y luego bien escrita. En el segmento 2 pasa algo similar, pero en este caso se trata de la pronunciación de una letra del alfabeto. En el segmento 3 se produce una duda del locutor, que pronuncia un sonido *eee* cuando en realidad debería haber continuado la frase. En el segmento 4, el locutor repite la pronunciación del *que* siendo únicamente uno de los dos válido, y de nuevo ocurre la situación del segmento número 2.

Capítulo 3

Software de Reconocimiento

3.1. Software de Preproceso

En primer lugar, destacar que para el preproceso del corpus original (videos en formato estádar MP4), se han realizado los siguientes filtrados, para extraer una señal definitiva de audio en formato WAV:

1. Extracción del canal de audio del video original utilizando *ffmpeg*.

2. Recodificación del audio en formato WAV a un canal y 16000Hz con SOX.

Así pues, el software implicado es el conocido manejador de ficheros multimedia *ffmpeg* y el recodificador de diversos formatos SOX, muy conocidos en el area de la multimedia y el procesado de ficheros.

Por otro lado, para el preproceso de las transcripciones se ha utilizado un script manual en Python 3, creado por el autor de este proyecto final de carrera, para adaptar las transcripciones a las convenciones tomadas en el capítulo anterior.

3.2. BLISS Lexicon

BLISS es una librería escrita en Python que nos permite generar un modelo léxico, que es el que contendrá todos los fonemas posibles, y la transcripción fonetica de todas las palabras que aparecen en el corpus [Uni12a]. En otras palabras, es el nexo de unión entre el Modelo de Lenguaje y el Modelo Acustico.

Para generar este modelo, lo primero que hacemos es parsear todas las transcripciones y extraer todos los textos en un fichero (cada segmento transcrito en una linea), eliminando acentos, signos de puntuación y demás carácteres no pronunciables. Además convertimos la cadena resultante a mayúsculas, ya en cuando se habla no se diferencian las mayúsculas de las minúsculas.

El siguiente paso es obtener una lista de todas las palabras que aparecen en el corpus de entrenamiento a partir de los ficheros generados anteriormente. A esta lista

de palabras se le añade a la derecha su transcripción fonética, como se puede observar en la tabla 3.1.

Tabla 3.1: Ejemplo de varias Transcripciones Fonéticas

Palabra	Transcripción Fonética
ABACO	a b a k o
ACCEDIENDO	a k z e d i e n d o
NOTACION	n o t a z i o n
PREVIAMENTE	p r e b i a m e n t e
PEDIAN	p e d i a n
RECUERDO	@ e k u e r d o
REFLEJAR	@ e f l e x a r
TOTAL	t o t a l

Por último, utilizamos sobre el fichero resultante con la lista de todas las palabras y su transcripción fonética, la herramienta Bliss Lexicon [Uni12a] que genera un modelo léxico acorde al formato que necesita el RWTH ASR System (ver sección 3.4). Este modelo léxico posee las siguientes características:

1. Posee una lista completa de todos los fonemas utilizados.
2. Genera cuatro lemas especiales para el silencio, inicio de frase, final de frase y palabra desconocida (fuera del vocabulario del léxico).
3. Formatea la lista de lemmas (palabra con su transcripción fonética).
4. Es un fichero XML bien formado, estándar y legible.

Generar correctamente el modelo léxico es el primer paso para poder entrenar los modelos que se necesitan para nuestro sistema.

3.3. SRILM NGram

Para generar el Modelo de Lenguaje, hemos utilizado la herramienta SRILM [Sto02] que es compatible con el RWTH ASR System (ver sección 3.4). Esta herramienta se ha ido desarrollando desde 1995 por el *SRI Speech Technology and Research Laboratory* y cuenta ahora con una gran variedad de algoritmos y optimizaciones.

El Modelo de Lenguaje se genera utilizando esta herramienta sobre el fichero de frases parseado creado previamente (en la sección 3.2), y genera un fichero en texto plano con todos los n-gramas y su probabilidad de aparición. Sin embargo existen muchas variantes que se listan a continuación:

1. Permite fijar cualquier orden de N-Gramas al modelo de lenguaje (unigramas, bigramas, trigramas, cuatrigramas, etc).

2. Permite utilizar multiples técnicas de descuento entre las que destaca el algoritmo Kneser-Ney [KN95] que mejora los resultados.

3. Permite aplicar técnicas de suavizado al modelo de lenguaje según lo explicado en la sección 1.5.3 de este documento.

Como hemos comentado, el modelo de lenguaje generado es un fichero en texto plano con diversas secciones [Sto02]. La primera sección indica para cada orden de n-gramas que cantidad de ellos se ha generado (si se aplica suavizado habrán datos desde el n-grama escogido hasta el unigrama). Después aparece la lista de los n-gramas divididas en secciones para cada uno de los n-gramas (una sección con unigramas, otra con bigramas, otra trigramas, etc). A cada uno de estos n-gramas le acompaña la probabilidad en formato logarítmico en base 10, para aumentar la precisión ya que suelen ser probabilidades muy pequeñas (al haber muchisimas palabras en el modelo de lenguaje).

3.4. RWTH ASR System

El RWTH ASR System es un toolkit de Reconocimiento del Habla Continua creado en el 2001 y mantenido por la *RWTH AACHEN University*. Es un toolkit fundamentalmente centrado en la generación de Modelos Acusticos [Uni12c], sin embargo puede utilizar el Modelo de Lenguaje de SRILM para realizar el proceso de entrenamiento y reconocimiento del habla completos.

Este toolkit posee una gran cantidad de características que lo hacen uno de los toolkits más avanzados para el Reconocimiento del Habla Continua [LGH+07] [RGH+09]. Entre estas características destaca la posibilidad de realizar con este toolkit todas las fases del Reconocimiento del Habla Continua (excepto la generación del Modelo de Lenguaje) que se ilustran en las figuras 1.2 y 1.3.

En esta sección vamos a introducir el funcionamiento básico de este toolkit, y a explicar el proceso que se sigue para realizar las diferentes fases del Reconocimiento del Habla con él.

3.4.1. Extraccion de Características

El proceso de extracción de características requiere inicialmente de segmentos de audio en formato WAV; por lo que debemos preprocesar los videos para extraerles el audio y segmentarlos según las marcas de tiempo que se hayan colocado manualmente en la transcripción del video. Para esto hemos utilizado las herramientas *ffmpeg* y SoX, según se ha comentado en la sección 3.1.

A continuación se deben crear unos ficheros recording y corpus que simplemente listan todos los segmentos que se deben utilizar para la extracción de características. En este caso hemos creado un par de estos ficheros por cada video del corpus, listando todos los segmentos del video. El formato de estos ficheros es muy simple y puede consultarse en la documentación oficial del toolkit [Uni12b].

A partir de este punto, como se puede observar en la figura 3.1, pasamos a la fase de extracción de características propiamente dicha. Para esta fase ya utilizamos el

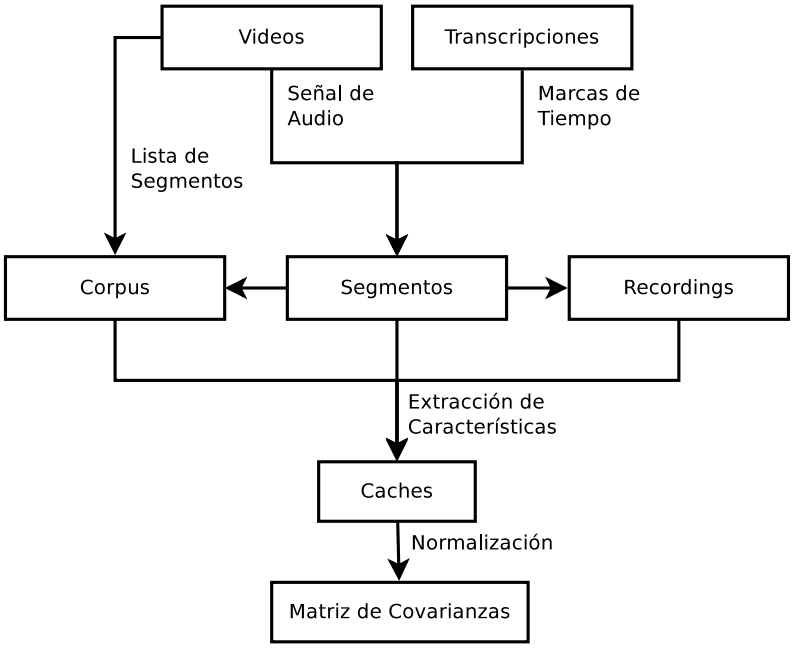

Figura 3.1: Proceso de Extracción de Características con RASR

toolkit RASR, definiendo un fichero de configuración indicandole todos los parámetros que necesita para esta fase. Aproximadamente con un RTF de 0,06 todo el corpus presentado en la sección 2.3 ha sido procesado con éxito por el RASR obteniendo un fichero cache por cada video que contiene su MFCC (ver sección 1.4).

A continuación se van a normalizar las cachés creando una matriz de covarianzas mediante dos ficheros de configuración más, que el RASR aplicará en dos pasos consecutivos. Esto nos permite normalizar todas las características entre si para que no se distorsione la fase de aprendizaje. Este proceso es muy rápido y apenas tarda con el corpus del presente proyecto, obteniendo un RTF de procesamiento que se aproximó al 0,01.

3.4.2. Entrenamiento con Monofonemas

Para iniciar la fase de entrenamiento lo primero que se debe de hacer es definir unas particiones en el corpus de datos. De esta forma tradicionalmente se divide en un conjunto de entrenamiento, uno de validación y otro de testing, y se aplica esta fase únicamente sobre el conjunto de entrenamiento creando un fichero Bundle que lista todas las cachés creadas en el paso previo.

A continuación se debe definir un fichero de configuración que utilizará el RASR para realizar el entrenamiento [Uni12b]. En este fichero no se indica únicamente las rutas, y algunos detalles sin importancia, sino que se deben especificar dos parámetros

muy relevantes:

1. Estados: Se deben de especificar el número de estados del Modelo de Markov y sus repeticiones para cada uno de los fonemas definidos en el Modelo Léxico. Así pues, un valor de 5 provocará que para cada fonema de una palabra hayan cinco estados para representarla.

2. Mixturas: Se deben definir el número de mixturas de gausianas por estado. Generalmente este valor es un número múltiplo de 2 como 128 o 512. Además se define el número de iteraciones que se utilizan para cálcular estas mixturas (un valor adecuado oscila entre 1 y 4).

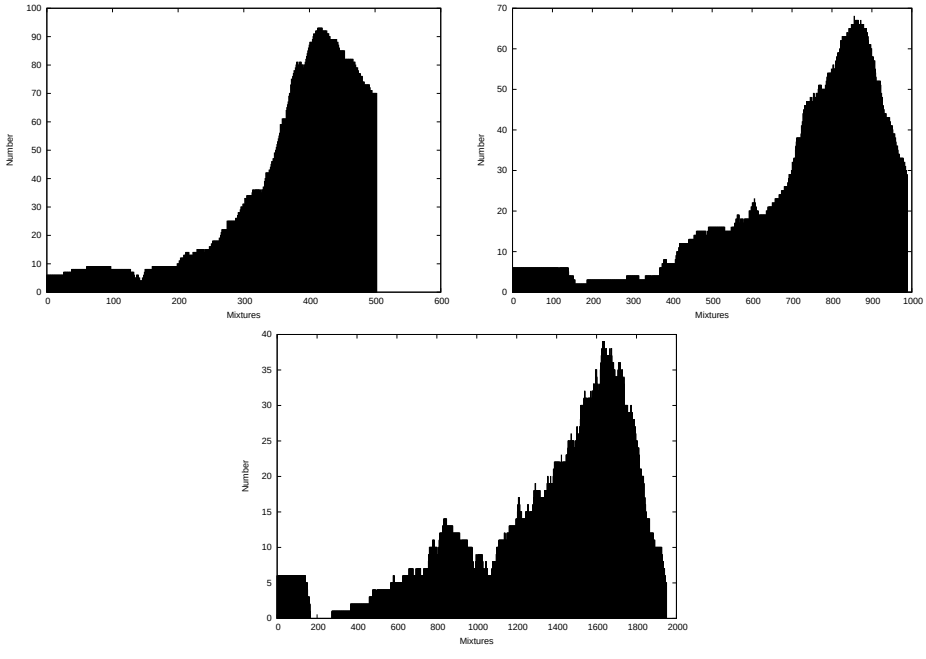

Figura 3.2: Cantidad de Estados que tienen un número de Mixturas.

Destacar que el número de mixturas es el máximo que un estado puede tener, pero pueden haber estados con muchas menos mixturas. Esto se puede observar claramente en la figura 3.2 que representa en cada columna un número de mixturas y su valor es el número de estados que poseen esa cantidad de mixturas, y que ha sido extraida de un corpus de entrenamiento que se define con el corpus presentado en la sección 2.3 casi completo.

Así pues, en la figura 3.2 comprobamos que para un límite máximo de 512 mixturas hay muchos estados en la zona de 450-512 mixturas; para 1024 ya vemos como el pico está sobre las 850 mixturas, pero aún hay muchas con el límite máximo de mixturas definido. Sin embargo, para 2048 ya vemos como prácticamente no hay estados con

ese número de mixturas (que marca el límite), lo cual lo convierte en un valor idoneo, ya que de continuar aumentando el límite muy probablemente sufriremos problemas de sobreentrenamiento (aunque esta técnica del RASR lo evita en gran medida).

Respecto al número de estados, comentar que debe estimarse sobre el conjunto de validación probando diferentes configuraciones y analizando el WER final y sus variaciones.

3.4.3. Entrenamiento con Trifonemas

Una vez el modelo acústico con monofonemas se ha entrenado adecuadamente, y partiendo de este, podemos generar un modelo acústico con trifonemas. Un trifonema es una secuencia de tres fonemas consecutivos diferentes; por lo que ahora en vez de tener el fonema *k* tenemos por ejemplo las combinaciones *aka*, *ake*, *aku*, *aki*, *ako*, *oka*, *eke*, *kei*, *kai*, *koa*, etc.

Lo que se hace en primer lugar es lo que se denomina un *cart*, que es un árbol de decisión fonético. En el se agrupan mediante una serie de preguntas los fonemas similares en clusters fonéticos, como por ejemplo trifonemas con un silencio al final (que representa los trifonemas fin de palabra) o los que contengan una vocal en medio. Destacar que en esta fase existe otro parámetro que se debe optimizar:

1. Número de Hojas Máximo: Este valor define el número de hojas máximo que puede puede tener el cart. Un valor muy pequeño quita mucha variabilidad al árbol de forma que los trifonemas no se entrenan todos lo que podrían; sin embargo un valor muy alto provoca demasiada variabilidad y sobreentrenaría a los trifonemas.

Después de generar este árbol y basandonos en él, ajustando otro fichero de configuración se pueden entrenar los trifonemas con el RASR utilizando la información del modelo acústico con monofonemas. En este caso es necesario también definir los mismos dos parámetro anteriores (número de mixturas y número de estados por trifonema). Destacar que en este caso, el número de mixturas necesarias suele ser menor, ya que los trifonemas son mucho más especificos que los monofonemas.

3.4.4. Reconocimiento del Habla

Para reconocer se utiliza el modelo acústico entrenado en la fase 3.4.3, el modelo de lenguaje y el modelo léxico. Además se necesita de nuevo de un conjunto de videos segmentados (con o sin transcripción) sobre el cual aplicar el reconocimiento.

Tipicamente para la fase de ajuste de parámetros se utiliza un conjunto de validación independiente del de entrenamiento, y para el reconocimiento final se utiliza un conjunto de testing. La idea consiste en utilizar el conjunto de validación para estimar dos parámetros que ponderan el uso del modelo acústico respecto al de lenguaje para conseguir la mejor estimación final. Dichos parámetros son:

1. Grammar Skip Factor (GSF): Este factor pondera el uso del modelo acústico respecto al modelo de lenguaje, de forma que valores bajos dan prioridad al modelo acústico, y valores altos la otorgan al modelo de lenguaje.

2. Word Insertion Penality (WIP): Fija una penalización por palabra introducida, ya que el modelo de lenguaje tiende a dar mayor probabilidad a las palabras cortas respecto a las palabras largas.

Una vez se han estimado correctamente estos dos factores podemos con esos valores obtenidos sobre el conjunto de validación, reconocer sobre el conjunto de testing obteniendo así un valor de WER final que medirá la eficacia de nuestro sistema.

Sea cual sea el objetivo de reconocer (modelos de monofonemas o trifonemas), el proceso es similar para ambos. Se emplea un fichero de configuración (indicando entre otras cosas el valor del GSF y del WIP), que el RASR utilizará para realizar el reconocimiento sobre el conjunto de test que se le especifique [Uni12b].

Si dicho conjunto tiene asociadas unas transcripciones, además de reconocerlas podrá entre otras cosas generar los indices de error que luego se utilizaran para calcular el índice de error del sistema (WER). Destacar que el resultado de dicho reconocimiento se genera en un fichero de log que está en formato XML, donde se pueden observar multiples informaciones.

1. Toda la configuración utilizada en el reconocimiento al completo (las que asigna el por defecto, o las compuestas resueltas incluidas).

2. Computos globales como número total de mixturas, rango de las mismas, número total de n-gramas, número de fonemas, número máximo de estados intermedios que ha generado el algoritmo, etc.

3. Para cada segmento reconocido, su ruta, el locutor, las trancripciones foneticas estimadas, y una comparativa con el texto original.

4. Finalmente una serie de estadísticas y puntuaciones de confianza, y de palabras reconocidas, añadidas, editadas, borradas, etc.

Como podemos comprobar se trata de un toolkit realmente completo que nos permite no solo realizar el reconocimiento del habla continua sino muchas otras cosas relacionadas.

3.5. Analizador WER++

Una vez el proceso de reconocimiento ha finalizado, podemos utilizar la herramienta WER++ creada por Nicolás Serrano Martínez-Santos en el 2011 para evaluar de una forma mucho más cómoda los resultados obtenidos en el reconocimiento.

Así pues, y aunque lo fundamental de esta herramienta es que a partir de un log de reconocimiento del RASR nos extrae el WER final (junto con el total de borrados, inserciones y renombrados) según lo visto en la sección 1.6, esta herramienta tiene muchas más utilidades.

Entre estas utilidades alternativas nos permite ver por ejemplo, una comparativa mediante colores del texto original con el de referencia, para ver mejor cuales han sido los fallos más comunes. Además te permite ver los N fallos que más puntos suman al WER para poder atacar la parte importante de los errores de reconocimiento.

En otras palabras, es una herramienta muy completa para poder explorar a fondo todos los resultados del reconocimiento del habla mediante el toolkit RASR.

3.6. Programación de un Front-End

Debido a la complejidad de todas las herramientas de software implicadas en el Reconocimiento del Habla Continua, se ha optado en primer lugar por realizar un front-end unificado para facilitar el uso de todas las herramientas.

Así pues, utilizando el lenguaje Python hemos creado un front-end, que utilizando un único fichero de configuración y las herramientas mencionadas en las secciones 3.2, 3.3, 3.4 y 3.5 es capaz de realizar el proceso completo de entrenamiento o reconocimiento.

El fichero de configuración es un simple fichero de texto en el cual se especifican las siguientes propiedades que se utilizarán durante la ejecución del presente front-end:

- Entrenamiento: Numero de estados, repeticiones de los estados, mixturas, refinamientos de las mixturas, y n-gramas.
- Ajuste: GSF máximo, mínimo y el incremento a probar; WIP máximo, mínimo y el incremento a probar, WIP del silencio máximo, mínimo y el incremento a probar.
- Reconocimiento: GSF, WIP y WIP del silencio finales para el reconocimiento.
- Miscelaneo: Número de hilos paralelos de proceso, y filtrado o no de carácteres acentuados del modelo de lenguaje.
- Cluster: Datos de login y directorio de trabajo del cluster, si se desea utilizar uno para llevar a cabo el ajuste de parámetos.
- Particiones: Nombre del subdirectorio de videos que define la partición de entrenamiento, ajuste y testing final.

Una vez especificados dichos parámetros se requiere definir un directorio de trabajo en el cual inicialmente debe haber un directorio videos, y dentro de este un directorio por cada particion de datos que se quiera definir, con los videos en su interior. Una vez el programa se ejecute se generarán todos los resultados y los ficheros intermedios de forma organizada en su interior.

Los modos de ejecución del front-end que se ha creado, permiten realizar de forma atómica cada una de las fases de forma independiente y sencilla, y son los que se enumeran en la siguiente lista:

- Extract: Realiza la extracción de características de todos los videos del directorio de videos (de todas las particiones de datos), y las guarda de forma ordenada generando los directorios correspondientes.
- Prepare: Une las características de los videos según las particiones definidas y calcula las matrices de varianzas y covarianzas preparando el entrenamiento.

- LexLm: Genera el modelo léxico y el modelo de lenguaje de la partición de entrenamiento utilizando la configuración especificada.
- Google: Unifica el modelo léxico y el de lenguaje actuales con los de Google N-Grams utilizado en la mejora presentada en la sección 5.3.
- 1Training: Realiza el proceso completo de entrenamiento con monofonemas, utilizando la configuración especificada en el fichero de configuración.
- 3Training: Realiza el proceso completo de entrenamiento con trifonemas, basandose en el entrenamiento inicial de monofonemas generado en el paso previo, y utilizando la configuración especificada.
- 1Adjust: Inicia el proceso de ajuste automatico del GSF y WIP sobre la partición de validación utilizando el último modelo generado en el entrenamiento con monofonemas.
- 3Adjust: Inicia el proceso de ajuste automatico del GSF y WIP sobre la partición de validación utilizando el último modelo generado en el entrenamiento con trifones.
- 1Testing: Inicia el proceso de testing sobre dicho conjunto utilizando el modelo de monofonemos y el GSF y el WIP especificado en el fichero de configuración, que deberán ser los optimos obtenidos en la fase 1Adjust.
- 3Testing: Inicia el proceso de testing sobre dicho conjunto utilizando el modelo de trifonemas y el GSF y el WIP especificado en el fichero de configuración, que deberán ser los optimos obtenidos en la fase 3Adjust.

Así pues, el proceso de Reconocimiento del Habla se puede simplificar mucho, y de esta forma acelerar todo el proceso de experimentación que se va a llevar a cabo como parte fundamental de este proyecto.

Capítulo 4

Experimentos Preliminares

4.1. Particion Utilizada

El objetivo de esta experimentación preliminar es realizar un ajuste amplio y preciso de los parámetros básicos necesarios en un sistema de reconocimiento del habla continua. Para ello, debido al gran coste computacional de las tareas de entrenamiento y validación, se ha optado por utilizar un conjunto de datos mucho más limitado con el fin de estimar de forma mucho más rápida los intervalos óptimos para cada uno de los parámetros del ajuste.

Hemos utilizado un subconjunto del corpus original, con 47 horas de entrenamiento, 2 horas y media de validación, y 5 horas de test final. Destacar que el conjunto de validación y el de test respecto al de entrenamiento, poseen independencia del locutor, pero no del área temática de los mismos.

Tabla 4.1: Comparativa del Corpus Empleado y el Total de Videos

Conjunto	Horas	Videos
Polimedia Completo	1350	6830
Nuestro Conjunto	120	732
Corpus Entrenamiento	47	300

4.2. Proceso de Optimización

Para llevar a cabo la optimización del WER (como medida de los resultados de nuestro sistema) hemos llevado a cabo un proceso de ajuste de los parámetros mostrados en la lista siguiente, siguiendo el esquema que se muestra en la figura 4.1.

1. Grammar Scale Factor: Rango de exploración de valores entre 3 y 20, a intervalos de la unidad.

2. Word Insertion Penality: Rango de exploración de valores entre 3 y 20, a intervalos de la unidad.

3. Número de Estados: Pruebas de 2 a 6 estados por fonema con 1 o 2 repeticiones por estado.

4. Número de Mixturas: Lista de valores discretos de 2^7, 2^8, 2^9, 2^{10} y 2^{11} mixturas de gausianas.

5. Monofonemas o Trifonemas: Todos los parámetros anteriores se han optimizado para un modelo de monofonemas, y para otro de trifonemas.

6. Número de Hojas en Trifonemas: Para el modelo de trifonemas se han probado con un máximo de 400, 600 u 800 hojas.

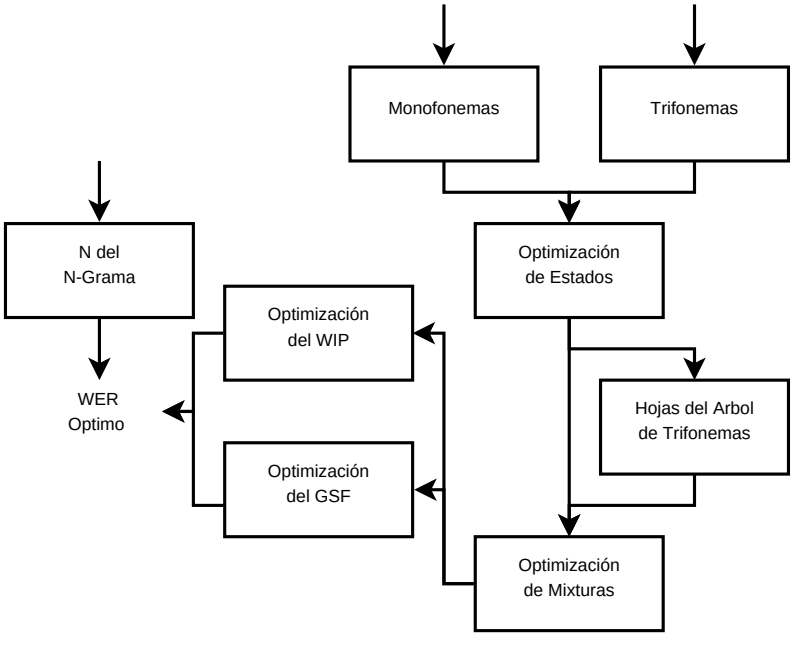

Figura 4.1: Priorización de las Optimizaciones Realizadas

Así pues, como podemos ver en la figura 4.1, para optimizar un entrenamiento con monofonemas (por ejemplo) debemos probar distintos valores de posibles estados, y para cada uno de esos valores distintos valores de mixturas, y para cada una de esas mixturas distintos valores de GSF y WIP. Para el caso de trifonemas se añade de forma intermedia el número de Hojas del Arbol de trifonemas.

Destacar que el tiempo de entrenamiento para monofonemas de cada una de las pruebas es muy elevado, siendo de aproximadamente de 0,26 de RTF para 2^7 mixturas, 0,39 de RTF para 2^9 mixturas, y hasta 1 de RTF para el caso de 2^{10} mixturas de

gausianas. En otras palabras, el tiempo de entrenamiento de los modelos depende de forma directa del número de mixturas por estado que se empleen en su entrenamiento, y tiene una evolución exponencial de tiempo respecto al número de mixturas utilizado.

En el caso de trifonemas, el tiempo de entrenamiento es aproximadamente un poco más del doble que en el entrenamiento de monofonemas, ya que deben entrenarse primero estos y a partir de ellos los modelos de trifonemas. Así pues, el tiempo de entrenamiento vuelve a depender del número de mixturas, que para el caso más común de 2^9 mixturas alcanza un RTF de 1,07.

4.3. Resultados Obtenidos

En primer lugar, se ha realizado el ajuste del número de estados por fonema utilizando un modelo de monofonemas y fijando las mixturas a un número arbitrario de 2^7 mixturas por gausianas. Los resultados de esta exploración pueden verse en la figura 4.2, en la que podemos ver como el WER varía de 66.80 de máximo a 48.69 en el valor óptimo de cinco estados por fonema.

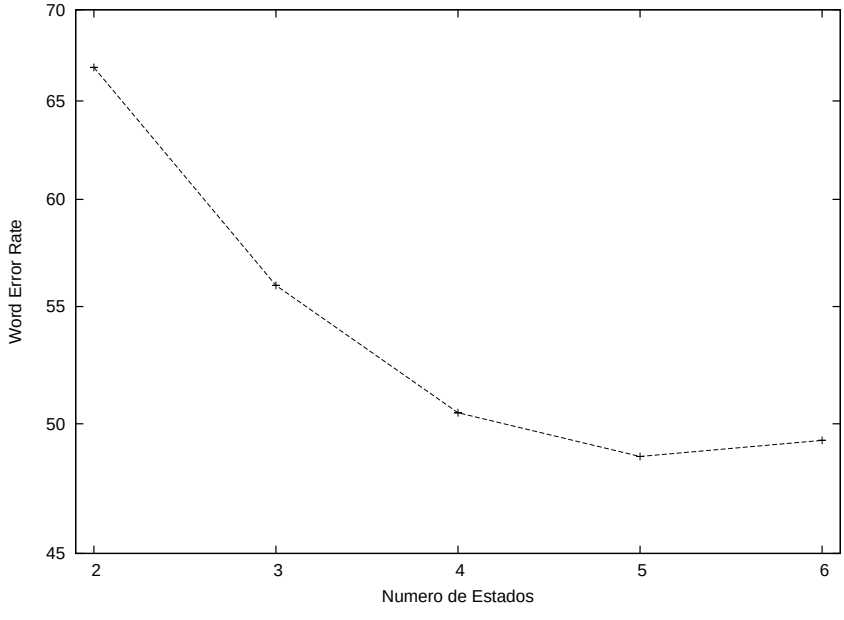

Figura 4.2: Ajuste del Número de Estados con Monofonemas

A continuación se ha realizado una exploración del número de mixturas de gausianas óptimo para el modelo de monofonemas, fijando el número de estados al óptimo encontrado previamente de 5 estados por fonema. La figura 4.3 muestra los resultados de la exploración que oscila entre 48.69 y 44.55 en el valor óptimo de 2048 mixturas de gausianas.

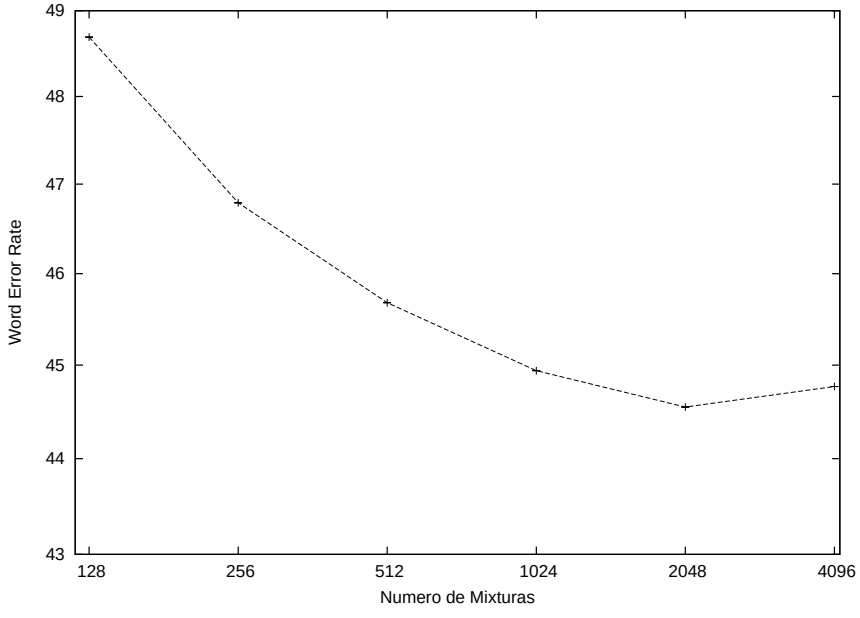

Figura 4.3: Ajuste del Número de Mixturas con Monofonemas

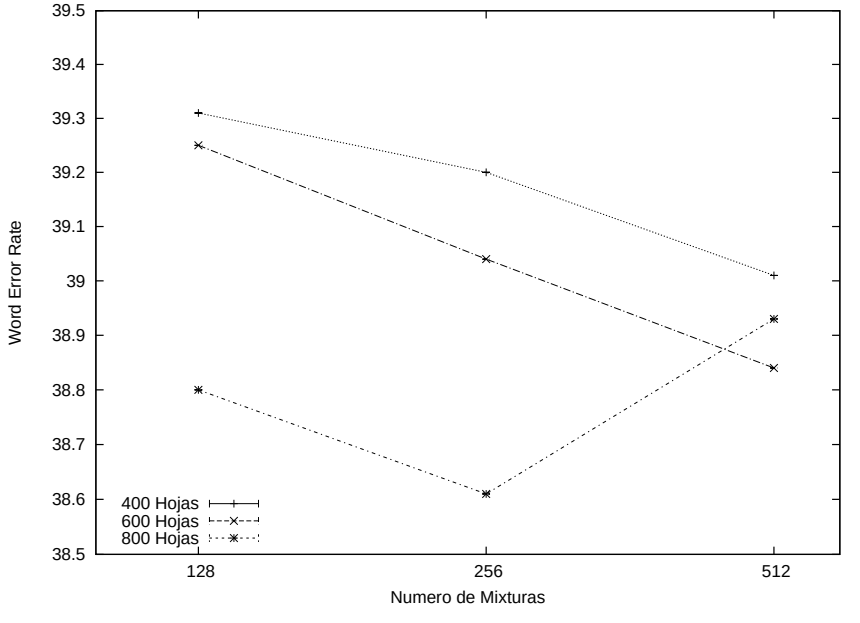

Figura 4.4: Resultados del Ajuste de Trifonemas

Finalmente, se ha pasado a utilizar un modelo de trifonemas, y utilizando el número de estados óptimo y la zona óptima de mixturas de gausianas obtenidas a partir de la exploración de monofonemas, se ha realizado una exploración conjunta del número de hojas del árbol de trifonemas y el número de mixturas óptimo en cada caso. Los resultados pueden observarse en la figura 4.4, que nos indica claramente el óptimo en 800 hojas y 256 mixturas con un WER de 38.61.

Los resultados de este primer experimento ratifican en primer lugar, que es mucho mejor utilizar un sistema basado en trifonemas que uno basado en monofonemas, ya que funcionan mucho mejor. Además, este experimento pone de manifiesto la complejidad de la tarea de Reconocimiento del Habla (en términos generales), y de los resultados de la misma.

Destacar que es necesario optimizar el GSF y el WIP [Uni12c] en cada una de las pruebas, ya que estos factores determinan que el WER alcance un mínimo global o se dispare de forma muy abrupta. Así pues, cada uno de los puntos de la gráfica representar el valor de GSF y WIP optimo encontrado para cada configuración de parámetros concreta.

Así pues, en este primer experimento hemos conseguido un WER óptimo de 38.61 con modelos de trifonemas de 5 estados sin repetición, y 2^8 mixturas de gausianas en 4 refinamientos sucesivos, con 800 hojas máximas, un GSF de 10 y un WIP de 12. Destacar finalmente que estos resultado nos ofrecen una transcripción legible, aunque con bastantes errores sintácticos, pero que sin embargo es un buen punto de partida para esta compleja tarea del Reconocimiento del Habla Continua.

Tabla 4.2: Resultados Optimos de la Experimentación Preliminar

Tipo de Modelo	WER Optimo
Monofonemas	44.55
Trifonemas	38.61

Capítulo 5

Experimentación

5.1. Particiones Utilizadas

Para la experimentación definitiva de este proyecto, y una vez realizada una primera toma de contacto en la sección 4.1, se han utilizado las particiones definidas sobre el proyecto europeo de investigación TransLectures [dV12a]. La especificación de estas particiones se puede observar con más detalle en las tablas 5.1 y 5.2, y nos garantiza en gran medida que los resultados no estén sesgados.

Tabla 5.1: Resumen de los videos de validación de TransLectures

Conjunto	Duración	Dominio	Genero
00501-Profesores_POLIMEDIA_II/M46	01:24:00	Leyes	Hombre
Profesores_POLIMEDIA/M74	00:48:00	Estadística	Mujer
Profesores_POLIMEDIA/M26	00:42:00	Gráficos	Mujer
00501-Profesores_POLIMEDIA_II/M54	00:36:00	Geolocalización	Hombre
Profesores_POLIMEDIA_I/M40	00:18:00	Botánica	Hombre

Tabla 5.2: Resumen de los videos de test final de TransLectures

Conjunto	Duración	Dominio	Genero
Profesores_POLIMEDIA_I/M47	01:06:00	Arquitectura	Mujer
Profesores_POLIMEDIA/M62	00:48:00	Marketing	Hombre
00505-Profesores_Alcoy/M03	00:42:00	Medio Ambiente	Hombre
Profesores_POLIMEDIA/M67	00:30:00	Informática	Hombre
Profesores_POLIMEDIA/M47	00:18:00	Leyes	Mujer

Este proyecto de investigación europeo define una partición de validación de 3 horas y 48 minutos, y otra para el test final de 3 horas y 24 minutos. El resto de videos descritos en la sección 2.3 que no están incluidos en ninguno de estos conjuntos, y que no han presentado ningún problema en el software de reconocimiento, se han utilizado para realizar el entrenamiento de los modelos (con un total de 89 horas).

5.2. Resultados Base

La optimización inicial se ha llevado a cabo siguiendo el proceso definido en la sección 4.2, que consiste en una exploración exhaustiva de parámetros sobre el conjunto de validación para obtener el mejor WER posible.

Debido al enorme tamaño del corpus de entrenamiento, esto se ha llevado a cabo partiendo de la base obtenida el la sección 4.3 de forma que se ha restringido dicha exploración a los parámetros que mejor han funcionado previamente.

Se ha restringido la exploración a modelos de trifonemas de tres a cinco estados sin repeticiones, explorando el número de hojas y mixturas en un rango muy acotado (600 y 800 hojas con 2^7, 2^8 y 2^9 mixturas de gausianas), y probando modelos de lenguaje de bigramas, trigramas y cuatrigramas. El resultado es una configuración de parámetros muy similar al obtenido en la sección 4.2, pero al utilizar más cantidad de audio se han elevado el número de mixturas de gausianas necesarias a 2^9, que es el óptimo encontrado.

Una vez obtenidos estos parámetros optimos, hemos pasado a realizar una prueba final sobre el conjunto de test especificado en la sección 5.3. Para ello se han empleado los parámetros obtenidos sobre el conjunto de validación en el reconocedor, obteniendo un WER final de 39.37 puntos (ver mas detalles en la tabla 5.3). Como podemos ver el resultado es muy similar al experimento de la sección 4.3, por lo que realizar el ajuste sobre un subconjunto de datos ha sido una decisión acertada (por el ahorro de tiempo que conlleva, y la validez de los resultados).

Tabla 5.3: WER Óptimo del Baseline de la Experimentacion

Modelo de Lenguaje	WER Óptimo
2-Gramas	40.62
3-Gramas	39.37
4-Gramas	39.44

Así pues, partiendo de estos resultados base iniciales, vamos a proceder a realizar una mejora sustancial en el proceso de reconocimiento para mejorar este aceptable resultado inicial de forma significativa.

5.3. Mejora del Reconocimiento

Los resultados obtenidos hasta ahora son una muy buena base inicial, pero se pueden intentar mejorar aún más utilizando diversas técnicas más avanzadas. Como objetivo en este proyecto nos hemos planteado utilizar la combinación de diversos modelos de lenguaje para mejorar estos resultados de forma significativa. Para ello, hemos empleado como modelo de lenguaje externo los conocidos Google NGrams [MSA+] disponibles de forma pública a través de su propia página web.

Los Google NGrams es el resultado que ha obtenido Google de aplicar técnicas de OCR sobre una gran cantidad de libros que datan desde el 1800 hasta el 2009. Una vez aplicado este OCR se han computado los unigramas, bigramas, trigramas, cuatrigramas y 5-gramas de todas las obras separando el resultado por años.

El resultado final es un compendio de cuentas divididas por años de una cantidad ingente de libros, que Google ha puesto a libre disposición, y que se van a utilizar en este proyecto para generar un modelo de lenguaje que complementará al extraido de las transcripciones manuales de los videos del conjunto de entrenamiento.

El primer paso ha sido unificar todas las cuentas que están divididas por años en una única gran cuenta que contemple todos los años a la vez. A continuación se ha extraido un modelo de lenguaje de n-gramas utilizando como base esta gran cuenta generada en el paso previo. Paralelamente se ha extraido un modelo de lenguaje de las transcripciones del conjunto de entrenamiento, y se han combinado linealmente los dos modelos de lenguaje en uno único, ponderando los dos iniciales para obtener la menor perplegidad posible sobre las transcripiones del conjunto de validación. Esta combinación se realiza partiendo de un vocabulario base, e incluyendo todos los n-gramas que se encuentran en este vocabulario de los dos modelos al modelo final unificado, ponderando los pesos de forma óptima.

Así pues, este proceso se va a repetir para modelos de lenguaje de bigramas, trigramas y cuatrigramas con el objetivo de encontrar el modelo de lenguaje óptimo. Además, se van a utilizar a la hora de realizar la combinación lineal tres vocabularios diferentes:

1. Únicamente el vocabulario exclusivo de poliMedia, sin añadir palabras de las cuentas de Google NGrams.

2. Todo el vocabulario de poliMedia y las 20.000 palabras más frecuentes de las cuentas de Google NGrams.

3. Todo el vocabulario de poliMedia y las 50.000 palabras más frecuentes de las cuentas de Google NGrams.

Para realizar este proceso nos hemos a servido de la herramienta SRILM NGram que se ha presentado en la sección 3.3, que nos permite llevar a cabo este proceso de forma muy sencilla, indicándole los textos de las transcripciones de poliMedia, las cuentas de Google NGrams, y el vocabulario que se desea emplear. Una vez finalizada la combinación lineal de cada uno de los modelos de lenguaje, se han vuelto a encontrar los valores de GSF y WIP óptimos con el nuevo modelo de lenguaje sobre el conjunto de validación. Los resultados obtenidos han sido los que se muestran en la grafica 5.1.

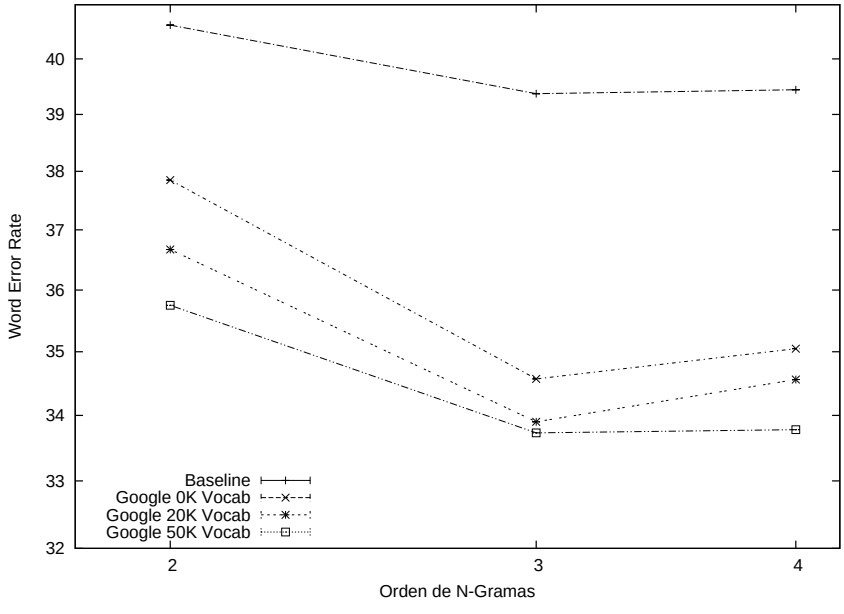

Figura 5.1: Evolución del WER en función del Orden de N-Gramas

Así pues, el caso en el que mejor se comporta es cuando utilizamos el vocabulario de poliMedia y las 50.000 palabras más frecuentes del modelo de lenguaje que hemos generado a partir de las cuentas de Google NGrams. Utilizando esta configuración hemos obtenido finalmente un 33.73 de WER sobre el conjunto de test lo que supone una mejora de un increible 5.64 puntos de WER. Esta mejora se ve reflejada en la tabla 5.4.

Tabla 5.4: Resultados Optimos de la Experimentación Final

Sistema Empleado	WER Trigramas	WER Cuatrigramas
Baseline	39.37	39.44
50K Google	33.73	33,78
Mejora Obtenida	5,64	5,66

Capítulo 6

Conclusiones Finales

6.1. Resumen del Trabajo

En el presente proyecto, se ha diseñado un sistema de reconocimiento del habla continua, optimizando los modelos y parámetros para obtener los mejores resultados posibles sobre un corpus especifico.

Como software básico destacar que hemos utilizado el RWTH ASR System y el SRILM NGram para el entrenamiento de modelos y el reconocimiento del audio, como programas relevantes y fundamentales para el presente proyecto.

El corpus empleado han sido los videos de libre distribución de la plataforma Polimedia de la Universidad Politécnica de Valencia, que se ha dividido en tres particiones (entrenamiento, validación y test final) según las particiones definidas en el proyecto europeo de investigación TransLectures [dV12a].

El proceso seguido ha consistido en primer lugar en realizar una exploración exhaustiva de parámetros y modelos, para obtener un punto de partida inicial aceptable, que ha sido de 39.37 puntos de WER.

Sobre este resultado inicial se ha planteado y ejecutado una mejora relevante, consistente en la fusión del modelo de lenguaje inicial con el proporcionado por Google con el objetivo de mejorar el modelo inicial. La mejora final obtenida ha sido de 5.64 puntos de WER obteniendo un resultado final de 33.73 puntos de WER.

Finalmente, estos modelos se han empleado en la construcción del reconocedor del habla automático para el sistema final, generado como resultado de este Proyecto Final de Carrera. Se trata de un programa que a partir de un directorio con segmentos de audio genera una transcripción aceptable de los mismos de forma completamente automática.

6.2. Mejoras Futuras

Durante el transcurso del presente proyecto, se han encontrado diversos aspectos a tener en cuenta para posibles mejoras futuras. De esta forma probablemente los

resultados finales que se pueden obtener sean mejores. La lista de estas mejoras es la siguiente:

1. Utilizar únicamente las palabras de los años más recientes de Google NGrams en lugar de todas las disponibles (años 1800-2009).

2. Combinar con más modelos de lenguaje externos, y no únicamente con uno que aunque extenso, es susceptible de errores al haberse extraido de forma automática mediante OCR.

3. Ampliar el vocabulario externo que se ha incluido a 100.000 y 200.000 palabras para observar el impacto directo que tiene sobre el WER.

4. Aplicar esta técnica al modelo acústico y no únicamente al modelo de lenguaje, para mejorar el modelo acústico siguiendo un proceso similar al planteado para el de lenguaje.

Así pues, quedan fijadas diversas lineas de actuación para mejorar o ampliar el presente proyecto final de carrera, que se explorarán en un futuro.

Bibliografía

[BGWL00] C. Barras, E. Geoffrois, Z. Wu, and M. Liberman. Transcriber: development and use of a tool for assisting speech corpora production. Speech Communication special issue on Speech Annotation and Corpus Tools, Vol 33, No 1-2, January 2000.

[CG98] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Tech. Report TR-10-98, August 1998.

[dA11] Junta de Andalucía. *Manual de atención al alumnado con necesidades específicas de apoyo educativo derivadas de discapacidad auditiva.* Consejería de Educación, 2011.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* PhD thesis, 1973.

[dV12a] Universidad Politécnica de Valencia. Proyecto europeo translectures. http://www.translectures.eu/, 2012.

[dV12b] Universidad Politécnica de Valencia. Web oficial del repositorio de polimedia. http://polimedia.blogs.upv.es/, 2012.

[HCS] J. M. Huerta, S. J. Chen, and R. M. Stern. The 1998 carnegie mellon university sphinx-3 spanish broadcast news transcription system. DARPA Broadcast News Transcription and Understanding Workshop, March, 1999, Herndon, Virginia.

[Jel98] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts London, England, 1998.

[KN95] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, II:181–184, May 1995.

[LGH+07] J. Lööf, C. Gollan, S. Hahn, G. Heigold, B. Hoffmeister, C. Plahl, D. Rybach, R. Schlüter, and H. Ney. The rwth 2007 tc-star evaluation system for european english and spanish. Interspeech:2145–2148, 2007.

[Mag05] Laura Sebastián Magaña. *Medios audiovisuales y prácticas performativas en el arte contemporáneo.* PhD thesis, Departamento de Arte, Universidad de Castilla - La Mancha, 2005.

[Mar10] Nora Cecilia Gómez Marín. *Promoción turística a través de los medios audiovisuales, Caso Medellín, Colombia.* Universidad Internacional de Andalucía, 2010.

[Med02] Juan José Hernández Medina. *De la palabra impresa al medio audiovisual: "Die Blechtrommel"de Günter Grass a la adaptación cinematográfica de Volker Schlöndorff.* PhD thesis, Departamento de Filología Alemana, Universidad Complutense de Madrid, 2002.

[MSA+] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. Science 14 January 2011: 331 (6014), 176-182.

[Per02] Telma López Perdomo. *La utilización de medios audiovisuales y ayudas didácticas para el aprendizaje del idioma inglés.* PhD thesis, Factultad de Humanidades, Universidad Francisco Marroquín, Abril 2002.

[RG05] Francisco Ignacio Revuelta and M. Cruz Sánchez Gómez. *El proceso de transcripción en el marco de la metodología de investigación cualitativa actual.* PhD thesis, 2005.

[RG06] José Luis Oropeza Rodríguez and Sergio Suárez Guerra. Algoritmos y métodos para el reconocimiento de voz en español mediante sílabas. *Computación y Sistemas*, 9(3):270–286, 2006.

[RGH+09] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney. The rwth aachen university open source speech recognition system. Interspeech:2111–2114, 2009.

[RJ] L. Rabiner and B. Juang. Fundamentals of speech recognition. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[Seg92] Héctor Miguel Rulot Segovia. *Un algoritmo de Inferencia Gramatical mediante Corrección de Errores.* PhD thesis, Grupo de Investigación en Reconocimiento del Habla, Universidad de Valencia, 1992.

[Ser05] Suárez Guerra Sergio. ¿100 % de reconocimiento de voz? trabajo inedito. 2005.

[SK83] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983.

[Sto02] A. Stolcke. Srilm - an extensible language modeling toolkit. Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado, September 2002.

[TW05] V. Tyagi and C. Wellekens. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE International Conference*, ICASSP 05(1):529–532, 2005.

[Uni12a] RWTH AACHEN University. Manual y documentación oficial de bliss lexicon. http://www-i6.informatik.rwth-aachen.de/rwth-asr/manual/index.php/Lexicon/, 2012.

[Uni12b] RWTH AACHEN University. Manual y documentación oficial del toolkit asr. http://www-i6.informatik.rwth-aachen.de/rwth-asr/manual/, 2012.

[Uni12c] RWTH AACHEN University. Presentación y notas de versión del toolkit asr. http://www-i6.informatik.rwth-aachen.de/rwth-asr/, 2012.

[Vit67] Andrew Viterbi. Error bounds for convolutional codes and a asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[YEG+06] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006.

Índice de figuras

Indice de Tablas