UNIVERSITAT POLITÈCNICA DE VALÈNCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Adaptación de modelos acústicos para el reconocimiento interactivo de contenidos educativos

Trabajo fin de grado - Grado de ingeniería informática

Autor:
Francisco Elías Serrano Martínez-Santos

Tutor:
Dr. Alfons Juan Císcar

8 de julio de 2014

A mi hermano.

Índice general

Índice de figuras

Índice de tablas

Resumen

Es difícil negar que la información es esencial en estos tiempos. Los avances de la ciencia y la existencia de internet han permitido el acceso a esta misma en todo el mundo.

Esta gran masa de información, en concreto las vídeo-charlas, como por ejemplo, cursos online o clases de universidades, están empezando a distribuirse a gran escala. Tanto universidades como páginas web independientes han comenzado a ofrecer gratuitamente contenidos educativos de diversas temáticas.

Pese a que estos contenidos estén al alcance de cualquiera, están limitados por el idioma y a personas con problemas auditivos. Es en este momento, donde el reconocimiento automático del habla juega un papel fundamental rompiendo estas limitaciones, gracias al cual se procesa la señal de audio y se transcribe la voz, mostrándola como subtítulos y ofreciendo la posibilidad de traducirlo a otros idiomas.

Actualmente las transcripciones automáticas aun no son perfectas, éstas rondan desde un 10 % al 40 % de error dependiendo de las condiciones. Por ejemplo, si hay ruido de fondo o si la recepción es de mala calidad, dificulta la transcripción automática, o también si el diálogo es espontáneo.

Para poder generar los sistemas de transcripción automáticos es muy importante la cantidad de datos anotados que se tengan, pero no resulta sencillo tener grandes conjuntos de ellos y típicamente, resultan caros de generar. Una manera de generar datos de forma barata es utilizar datos sin anotar, cuya anotación se obtiene automáticamente. Esto genera un problema que es el de seleccionar que parte de estos recursos tiene mayor calidad.

Un ejemplo de un repositorio de datos de gran tamaño, es el caso del repositorio poli[Media], contiene una gran cantidad de vídeos sin transcribir, de los que se querría hacer uso para mejorar los actuales sistemas de reconocimiento.

El objetivo de este trabajo es crea, describir y validar empíricamente una aplicación para seleccionar datos sin transcribir que puedan enriquecer el conjunto de datos de entrenamiento de un sistema de reconocimiento.

Este documento se ha dividido en cuatro partes bien diferenciadas:

Introducción Se introduce el concepto de reconocimiento de formas, en concreto el del habla, y se describe brevemente la teoría estadística sobre la que se asienta,

así como los algoritmo y técnicas utilizadas. También se enumeran los programas que se han empleado para realizar este trabajo.

Repositorio poli[Media] Se describe la base de datos que se ha utilizado para el trabajo, su origen, cómo se ha creado, qué cantidad de datos la forman y cómo se han generado los subtítulos.

Selección datos de alta confianza Este apartado explica cómo se seleccionan los datos de alta confianza de un conjunto mediante medidas de confianza y se presenta un análisis empírico sobre su eficacia.

Caso de estudio en transLectures En este capítulo se describe la utilización de la aplicación creada en este trabajo para mejorar el sistema de reconocimiento actual en poli[Media], desarrollado en el proyecto transLectures, y se discutirán los resultados obtenidos.

Por último se cerrará el documento con una conclusiónes sobre del trabajo realizado.

Capítulo 1

Introducción

1.1. Reconocimiento de formas

El área de la informática, y más concreto la rama de la inteligencia artificial, es la encargada de desarrollar sistemas que reconozcan objetos y los etiqueten de acorde a sus características. El reconocimiento automático se basa en, dado un objeto, se clasifica en su clase correspondiente, por ejemplo, un audio que contiene una frase de un profesor, sería etiquetado con sus distintas palabras o dicho de otra manera, con su transcripción.

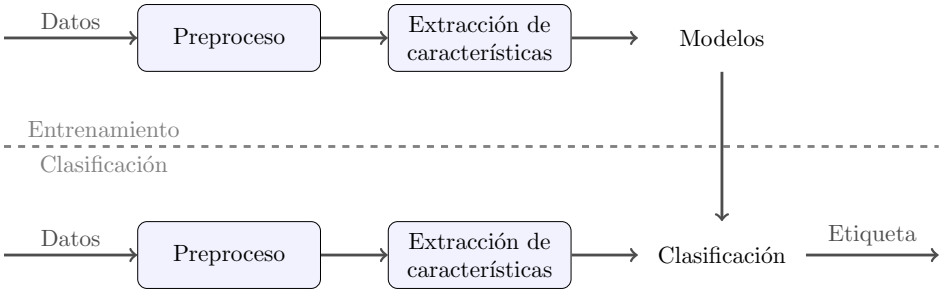

Figura 1.1: Sistema reconocedor

En la Figura 1.1 tendríamos un sistema de reconocimiento de formas, y enfocándonos en la parte de la clasificación, el proceso se divide en:

- **Preproceso:** Dado un objeto, se procesa para eliminar ruido y prepararlo para el siguiente paso, en el caso del ejemplo anterior, habrá que adquirir su señal, tras eso se filtra para eliminar el ruido para solo quedarnos con la parte que se necesita.

- **Extracción de características:** Se adquiere la información relevante para el sistema y se crea una representación numérica del objeto, para extraer toda la información discriminante que sea posible. Se generan coeficientes cepstrales en las frecuencias de Mel o MFCC[1] que lo que hacen es filtrar el sonido según el tracto vocal, incluido la lengua, dientes, etc. Esto determina que sonido se ha dicho.

- **Clasificación:** Con los sistemas de reconocimiento de formas previamente entrenados, asignamos una etiqueta a cada objeto, que sera la salida de la clasificación.

1.2. Reconocimiento automático del habla

Dentro del reconocimiento de formas, el campo que se trata en este trabajo es el del reconocimiento automático del habla o ASR[2]. A continuación presentamos el problema del ASR y como se puede solucionar mediante un sistema de reconocimiento del habla.

El objetivo del sistema ASR es el de, dada una señal acústica, procesarla y obtener la transcripción en texto de aquello que se ha dicho. Formalizaremos este problema como, dado una serie de T observaciones acústicas $\boldsymbol{x} = x_1, \ldots, x_T$, se puede obtener la secuencia de N palabras $\boldsymbol{w} = w_1, \ldots, w_N$ que corresponden a su transcripción, obteniendo aquella $\hat{\boldsymbol{w}}$ que maximiza la probabilidad a posteriori $p(w|x)$ [GY07], siempre que se cumpla que las muestras sean independientes e idénticamente distribuidas; o IID[3], o expresado matemáticamente lo podemos ver en la Fórmula 1.1.

$$\boldsymbol{w} = \arg\max_{w} \; p(\boldsymbol{w}|\boldsymbol{x}) \tag{1.1}$$

Así que $\boldsymbol{w}$ será la salida deseada. El problema es que, como $p(\boldsymbol{w}|\boldsymbol{x})$ es difícil de modelar directamente, aplicaremos la ley de Bayes para transformar (1.1) en un problema equivalente de búsqueda:

$$\boldsymbol{w} = \arg\max_{w} \; p(\boldsymbol{x}|\boldsymbol{w}) \; p(\boldsymbol{w}) \tag{1.2}$$

La probabilidad $p(\boldsymbol{x}|\boldsymbol{w})$ expresa la probabilidad de una muestra acústica sea generada dada una frase, esta parte se estimaría usando un *modelo acústico*. Por otra parte, la probabilidad de que una frase haya sido dicha. Esta probabilidad se estima mediante un *modelo de lenguaje*. En este caso, si para un lenguaje dado conociéramos estas probabilidades para todas las palabras posibles, el problema estaría resuelto. Pero, como podemos imaginar, este caso no se da en la realidad. Por lo tanto, el objetivo del ASR será aproximar estas probabilidades mediante el entrenamiento con

[1] del inglés, *Mel Frequency Cepstral Coefficients*
[2] del inglés, *Automatic Speech Recognition*
[3] del inglés, *independent and identically distributed*

datos previamente etiquetados. En concreto, en este trabajo:

Modelo acústico Se hace uso de un modelo híbrido de modelos de Markov de capa oculta o HMM[4] y redes neuronales profundas o DNN[5]. El entrenamiento de estos modelos requiere que las muestras de audio sean procesadas. En las Secciones 1.2.1 y 1.2.2 se verá con mayor profundidad.

Modelo de lenguaje En este paso se usan modelos de n-gramas, que se estiman mediante transcripciones de textos. Podemos verlo con mayor profundidad en la Sección 1.4.

Como ya hemos visto en la Figura 1.1, se necesita una fase de entrenamiento para la obtención de modelos a partir de muestras. Que este entrenamiento se haga con las técnicas adecuadas es igual o más importante que los datos usados para el mismo.

Para entrenar el modelo acústico se usa el algoritmo EM[6] para los HMMs y luego, a partir de éstos, se estiman las DNN mediante descenso por gradiente estocástico. Estos pasos los podremos ver con más detalle en la Sección 1.3. En lo que respecta al modelo de lenguaje basado en n-gramas, se hará a partir de estimación de n-gramas (Sección 1.4.1) y suavizado (Sección 1.4.2). Además, si se disponen de diferentes fuentes y/o de distintos dominios, se entrenan los modelos individuales para cada corpus y después se usan técnicas de interpolación para combinar las salidas (Sección 1.4.3).

En la Figura 1.2 podemos ver un diagrama que recoge el proceso de entrenamiento de un sistema ASR.

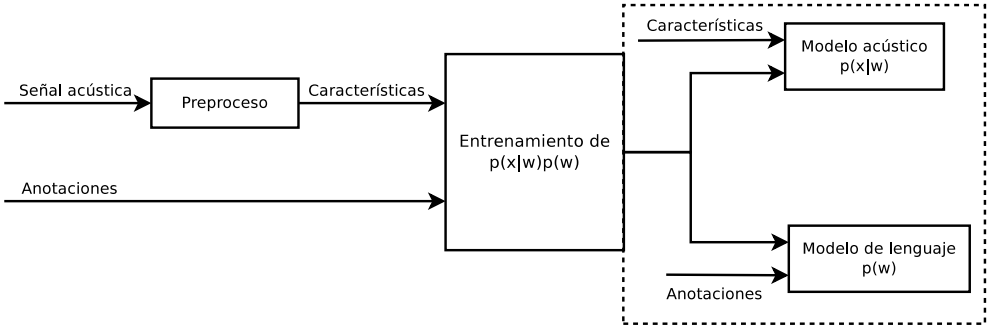

Figura 1.2: Sistema ASR

[4]del inglés, *Hidden Markov Model*
[5]del inglés, *Deep Neural Network*
[6]del inglés, *Expectation-Maximmisation*

1.2.1. HMM

Como ya hemos hablado en el apartado anterior, para calcular la probabilidad dada una secuencia de T características $\boldsymbol{x} = x_1, \ldots, x_T$ generadas por las transcripciones de las palabras N $\boldsymbol{w} = w_1, \ldots, w_N$, habría que aplicar $p(\boldsymbol{x}|\boldsymbol{w})$. Para simplificar, vamos a considerar que sólo usamos una palabra, por lo que la probabilidad quedará como $p(\boldsymbol{x})$. Después, para modelar la probabilidad de una frase, concatenaremos varios HMMs correspondientes a las distintas palabras.

Estimar directamente $p(\boldsymbol{x})$ no es posible, ya que deberíamos comprobar todas las posibles segmentaciones de $\mathbf{x}$ en la correspondiente transcripción. Para solucionar esto suponemos que cada elemento x_i de $\boldsymbol{x}$ ha sido emitido por un diferente estado q_i por un conjunto de estados finitos Q. Al igual que x_i, los elementos q_i siguen el mismo orden secuencial de $1, \ldots, T$. Una secuencia de diferentes estados puede representar una letra o una palabra. Si calculamos la probabilidad de $\boldsymbol{x}$ con $\boldsymbol{q}$

$$p(\boldsymbol{x}) = \sum_{q} p(\boldsymbol{x}, \boldsymbol{q}) \tag{1.3}$$

donde el último termino puede ser expandido usando la regla de la cadena de probabilidad

$$p(\boldsymbol{x}, \boldsymbol{q}) = \prod_{t=1}^{T} p(x_t, q_t | x_1^{t-1}, q_1^{t-1}) \tag{1.4}$$

Ahora hacemos dos suposiciones para aproximar el último término. Primero, suponemos que la probabilidad de emitir x_t sólo depende de q_t. Segundo, Suponemos que el estado q_t sólo depende del estado anterior q_{t-1}. De este modo, tenemos que:

$$p(x_t, q_t | x_1^{t-1}, q_1^{t-1}) = p(x_t|q_t)\ p(q_t|q_{t-1}) \tag{1.5}$$

En la la Ecuación 1.5 por una parte $p(x_t|q_t)$ corresponde a la probabilidad de emisión, que es la probabilidad de generar x_t del estado q_t. Por otro lado, $p(q_t|q_{t-1})$ expresa la probabilidad de transmisión del estado q_{i-1} a q_i.

Los HMMs son modelos generativos, que modelan la emisión de secuencias de vectores de características $\boldsymbol{x}$. Sin embargo, sólo se ve la emisión de secuencias $\boldsymbol{x}$, mientras que la secuencia de estados $\boldsymbol{q}$ permanece "oculta".

Dado que las transiciones sólo dependen del estado anterior, hay que definir qué estado se puede alcanzar desde uno anterior. Este problema se soluciona definiendo un autómata de estados finitos estocástico [VTdlH+05], donde cada estado q_i corresponde a uno dentro de Q, y cada enlace representa la transición del estado q_i a q_j

$$p(q_t = q_j | q_{t-1} = q_i) = a_{i,j}, \forall_i \sum_j a_{ij} = 1 \tag{1.6}$$

donde a_ij representa la probabilidad de transición, y la probabilidad de salir del estado suma

$$0 \leq a_{ij} \leq 1, \sum_j a_{ij} = 1 \tag{1.7}$$

El autómata expuesto en este trabajo puedes ser clasificado con diferentes tipos de topologías. La que se ha usado, comúnmente se llama topología de izquierda a derecha. Desde un estado q_i sólo podemos hacer 2 transiciones, sobre sí mismo o al estado siguiente. Podemos ver un ejemplo en la Figura 1.3.

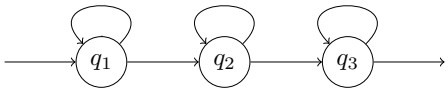

Figura 1.3: Ejemplo de la topología de izquierda a derecha

Ahora que ya hemos expuesto lo que es un HMM pasaremos a ver sus dos principales problemas, que son el problema del aprendizaje y al de la decodificación.

El problema del aprendizaje

El problema del aprendizaje en los HMMs es el de estimar los parámetros más verosímiles, dada una estructura, un conjunto de entrenamiento y sus correspondientes transcripciones. Los parámetros más verosímiles se obtienen, en reconocimiento de formas, usando el criterio de la log-verosimilitud (ML[7]) [SDHS01]:

$$L = \sum_{n=1}^{N} \log p(x_n|w_n) \tag{1.8}$$

En el caso del HMM esta ecuación es muy compleja y hace que no se pueda estimar, pero el uso de la descomposición, Ecuación 1.5, hace que sí se pueda. En ese caso se usa el algoritmo EM, que este se encarga de encontrar los estimadores máximo-verosímiles [DLR77] deseados.

El problema de la decodificación

El problema de la decodificación en un HMM es el de poder encontrar el camino más probable de los estados de la secuencias $\boldsymbol{q}$. El estado de la secuencia más probable $\hat{q}$ se calcula como:

$$\hat{q} = \arg\max_{q} \prod_{t=1}^{T} p(x_t|q_t)p(q_t|q_{t-1}) \tag{1.9}$$

que puede escribirse como

$$\hat{q} = \arg\max_{q1} \left\{ \arg\max_{q2} \left\{ \cdots \arg\max_{qT} \left\{ p(x_T|q_T)p(q_T|q_{T-1}) \right\} \cdots \right\} \right\} \tag{1.10}$$

Definimos la función recursiva de Viterbi [Vit67] de un estado j de la siguiente forma

$$v_j(j) = \arg\max_{i, i \in Q} v_{t-1}(i)p(x_i|q_i)p(q_j|q_i) \tag{1.11}$$

[7] del inglés, *Maximum Likehood*

que se calcula eficientemente usando el algoritmo de Viterbi

$$v_t(j) = \begin{cases} a_{0j} b_{j1} & t = 1 \\ \{\text{máx}_j \, v_{t-1}(i)\} a_{ij} b_{jt} & \text{los demás} \end{cases} \tag{1.12}$$

Al final, el estado que maximiza la función v_t se obtiene extrayendo el camino más probable de $\hat{q}$.

1.2.2. DNN

En los últimos años, las DNN han supuesto una revolución en el campo del reconocimiento de formas y por lo tanto en el campo ASR. Básicamente, una DNN recibe como entrada un vector numérico de tamaño fijo y computando una serie de transformaciones numéricas en sus distintas capas devuelve la probabilidad a posteriori de pertenecer a una etiqueta. En concreto la salida de una DNN está formada por tantos números como etiquetas haya.

La estimación de una DNN es compleja y requiere una serie de pasos. Se comienza por una red neuronal clásica con una sola capa oculta y se entrena mediante descenso por gradiente. Al acabar se elimina la capa de salida y se apila una capa oculta adicional y la de la salida y se vuelve a estimar de nuevo. Así tantas veces como sea necesario. De ahí, su nombre *deep*.

Su utilización se hace conjuntamente con modelos de Markov ya que las DNN no pueden segmentar la señal en sus distintas palabras.

Los detalles matemáticos de la estimación y clasificación se pueden ver en el artículo [DYDA12].

1.3. Proceso de entrenamiento del modelo acústico

En este apartado describiremos cómo hacer el proceso de entrenamiento del modelo acústico de un sistema ASR actual. Para poder entrenar el modelo acústico, dado un conjunto de vídeos, se necesita tener la extracción de características de sus muestras y su anotación fonética. El entrenamiento de este sistema es un proceso complicado que se divide en los siguientes pasos:

1. **Modelo estándar HMM:** Primero entrenamos un modelo estándar que servirá como un modelo acústico general. Para entrenarlo haremos:
   1.1. **Fonemas 1 mixtura HMM:** Se entrena el modelo usando un HMM por fonema de tres estados de modo que, cada uno de ellos modela la probabilidad de emisión con una mixtura de gausianas de una sola componente.
   1.2. **Tri-fonemas 1 mixtura HMM:** Partiendo del modelo anterior, podemos estimar un modelo más complejo que funcionará mejor, ya que los

fonemas de las palabras están relacionados con sus adyacentes, es decir, que por propiedades de la garganta y lenguaje solo podemos ir de uno a otro. Entrenaremos un HMM por cada tri-fonema (formado por el fonema anterior, el actual y el siguiente) de tres estados cada uno que se modela la probabilidad de emisión con una mixtura de gausianas de una sola componente.

1.3. **CART al modelo tri-fonemas para obtener fonemas atados:** Esto se hace porque en el caso del español, con 24 fonemas, al hacer tri-fonemas nos quedamos con un total de hasta 24^3 tri-fonemas, que es un conjunto muy grande. El algoritmo de CART se encarga de unir varios tri-fonemas.

1.4. **Entrenar hasta 128 mixturas:** Una vez tenemos los fonemas atados, pasamos a entrenar desde 1 a 128 componentes por mixtura de gausianas, ya que como los datos no siguen una distribución normal simple, necesitamos hacer uso de una mixtura de gausianas para aproximar distribuciones complejas.

2. **Modelo de CMLLR con DNN:** Con el paso anterior obtenemos un modelo acústico general, pero interesa adaptarnos a los locutores. Para ello se suele hacer una transformación a CMLLR[8] [PY11], que se encarga de adaptar las muestras originales a cada locutor y después, a partir de éstos, crea un nuevo modelo acústico adaptado.

   2.1. **Repetir pasos 1-4 del estándar para el CMLLR:** Hacemos los mismos pasos pero con las muestras adaptadas.

   2.2. **Entrenar los CMLLR usando DNN:** El último paso se encarga usar el modelo CMLLR de 128 componentes por mixtura de gausianas para entrenar la DNN que será la nueva encargara de modelar la probabilidad de emisión de los HMMs.

En este trabajo, todo esto se lleva a cabo mediante el software TLK [tUT], que lo veremos con más profundidad en la Sección 1.7.1.

1.4. Modelo de lenguaje

Como hemos dicho anteriormente, para calcular la probabilidad dado el conjunto de N palabras $\boldsymbol{w} = w_1, \ldots, w_N$ tendremos que aplicar $p(\boldsymbol{w})$. El objetivo de usar un modelo de lenguaje o LM[9], sirve para mejorar las transcripciones de una frase. Éste se encargará de asignar la probabilidad de que nuestro modelo genere esa frase. En concreto tendríamos que cada palabra tendría una probabilidad de aparición respecto a la anterior.

Suponemos que la frase $\boldsymbol{w}$ está compuesta por N palabras $w_1 w_2 \ldots w_N$, podemos calcular la probabilidad $p(s)$ como

[8] del inglés, *Constrained Maximum Likelihood Linear Regression*
[9] del inglés, *Language Model*

$$p(\boldsymbol{w}) = \prod_{i=1}^{N} p(w_i|w_1^{i-1}) \tag{1.13}$$

Intentar aplicar esta fórmula directamente implicaría suponer que la última palabra de la frase es dependiente de todo el resto de las palabras de dicha frase. Para ello necesitaremos un número de parámetros y muestras muy elevados para estimarlo. Una mejor aproximación sería considerar solamente un rango de palabras limitado, por lo que tendríamos solamente que cierta palabra depende de un número de palabras posteriores.

Los modelos de n-gramas limitan la ocurrencia de palabras. Por lo tanto, ahora sólo tendremos dependencia sobre un rango, por lo que tendríamos que la palabra en la posición i solo dependerá de las n anteriores, siendo n un parámetro fijo del modelo. Por lo tanto, el cálculo de la probabilidad de una frase podremos calcularla cómo

$$p(\boldsymbol{w}) \approx \prod_{i=1}^{n} p(w_i|w_{\text{máx}(1,i-n+1)}^{i-1}) \tag{1.14}$$

1.4.1. Estimación

Vista la idea de los modelos de n-gramas, hay que explicar cómo poder obtener uno de un lenguaje en concreto. Para ello necesitamos un corpus de entrenamiento, es decir, frases que, suponemos, están bien construidas a partir del lenguaje objetivo. Una vez obtenido, podemos aproximar la probabilidad deseada como

$$p(w_i|w_{i-n+1}^{i-1}) \approx \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} \tag{1.15}$$

Donde c es la cuenta de veces que aparece un n-grama en el corpus. Esta aproximación se conoce como estimación por máxima verosimilitud. Por ejemplo, la probabilidad de "la casa azul" se obtendría de calcular la probabilidad de que se genere "la casa azul" de la probabilidad de que se tenga "la casa".

1.4.2. Suavizado

El principal problema que surge de la aproximación por máxima verosimilitud es que se produce una distribución de probabilidades que asigna cero a cada caso que no se ha producido en el corpus de entrenamiento. Como la probabilidad de una frase es el producto de la probabilidad de los n-gramas que la forman, basta que sólo uno sea igual a cero para que la probabilidad de la frase sea cero. Como podemos imaginar, es muy poco probable que, aunque asumamos un n pequeño, los datos posean todos los n-gramas del lenguaje.

Las técnicas de suavizado se usan para solventar este problema. Se encargan de suavizar la distribución de las probabilidades con el fin de eliminar aquellas que son

nulas. Un estudio de las distintas técnicas disponibles para usarlo con los n-gramas se puede ver en [CG99]. En este trabajo empleamos un suavizado llamado *Modified Knesser-Ney* [CG99].

1.4.3. Interpolación

Cuando se disponen de diferentes recursos de texto, correspondientes a distintos dominios, se deberían combinar de manera eficiente. Esta técnica nos permite crear modelos independientes con datos que provengan de distintas fuentes y fusionarlos para obtener un modelo apto para el reconocimiento.

Las técnicas más usadas y efectivas para interpolar modelos de lenguaje es la interpolación lineal. Para cada uno de ellos le asignaremos un peso en la interpolación. Utilizamos el algoritmo EM para estimar los pesos.

1.5. Proceso de reconocimiento de un fichero de audio

Para poder hacer un reconocimiento necesitamos haber entrenado el sistema ASR. En concreto estará formado por dos partes: sistema estándar, formado por el modelo acústico de HMMs estándar y el modelo de lenguaje de n-gramas; y el sistema híbrido CMLLR que esta formado por modelo acústico híbrido de HMMs/DNN del CMLLR y el modelo de lenguajes de n-gramas. De modo que, si ya tenemos ésto, pasaremos a reconocer un conjunto de vídeos. Dado un vídeo ya segmentado, el proceso de reconocimiento es el siguiente:

- **Reconocer las muestras usando el modelo estándar:** Cada segmento de vídeo se reconoce con el modelo estándar. Con esto obtendremos una primera transcripción que usaremos como base para la siguiente fase.
- **Reconocer usando el modelo CMLLR y DNN:** Adaptamos el vídeo al modelo usando CMLLR y las transcripciones obtenidas en el primer paso. Tras eso le pasamos el vídeo adaptado al sistema híbrido CMLLR. La transcripción obtenida es la resultante del sistema.

1.6. Evaluación de los resultados

Para evaluar los resultados se usa el *Word Error Rate* (WER) [Wikb]. Intuitivamente, mide el número medio de operaciones de edición básicas necesarias para convertir la transcripción reconocida en su referencia, por ejemplo, un 10 de WER implicaría aproximadamente, cambiar un 10 % de la frase. El WER se deriva de la distancia de Levenshtein.

La forma de calcularlo es obteniendo el número mínimo de inserciones, sustituciones y borrados de una palabra a otra, que se necesitan para que está se transforme en la otra referencia. Teniendo N palabras que comparar, calcularemos el WER como:

$$WER = \frac{I + S + D}{N} \tag{1.16}$$

El software encargado de hacer esto en este trabajo es el "wer++", vistos con más profundidad en la sección siguiente.

1.7. Herramientas informáticas

A lo largo de la memoria se ha usado un *toolkit* informático para el reconocimiento del habla transLectures-UPV *toolkit* (TLK).

1.7.1. TLK

El TLK es un *toolkit* desarrollado por la Univèrsitat Politècnica de València, en el proyecto transLectures, para el reconocimiento automático del habla. Consiste en una serie de herramientas, utilidades y programas sobre una librería base en C. Es de código abierto, licenciado en ApacheV2 y para sistemas UNIX.

Los programas y scripts del TLK que se han usado han sido:

tLtask-train Este programa se encarga de realizar el entrenamiento del sistema ASR. Para ello hay que tener, en la carpeta que se va a trabajar, tres listas que tendrán que tener el mismo número de líneas y han de corresponderse cada línea con los demás archivos. Estas listas serán: una de las muestras, otra de los vídeos de los segmentos y otra de los fonemas de los segmentos. También habrá que poner otra lista extra que corresponderá a los clústers, que es lo mismo que la lista de los vídeos pero sin repetirse. Todo esto estará incluido en un fichero de configuración, en el que también podremos elegir si queremos entrenar por redes neuronales o no, entre otras características.

tLtask-recognise Este programa, al igual que le anterior, realizar el reconocimiento del sistema ASR. Para poder reconocer, en la carpeta donde trabajamos, hemos de tener una carpeta con los modelos de lenguaje y una lista de los vídeos a reconocer. También tendremos un fichero de configuración, donde tendremos las mismas listas que necesita el *tLtask-train*, pero éstas harán referencia a los vídeos a transcribir. En este fichero se podrá elegir si se quiere entrenamiento por redes o no, elegir los parámetros de búsqueda y si se quieren generar *wordgraphs*.

Para la evaluación de los resultados se han usado las aplicaciones "wer++" y "wgraph".

wer++ Este programa se encarga de comparar dos ficheros de texto y calcular el ratio de error de palabra o WER[10], así como el número de palabras que debería haber reconocido (inserciones), que sobran (borrados) y que debería sustituir por otra muy parecida (sustitución). Aparte de eso, también se puede generar la salida de la comparación, donde mostrará con colores cada una de las inserciones, borrados y sustituciones, para de este modo comprobar el tipo de errores que se obtienen.

wgraph Este programa permite hacer un reconocimiento en base a los archivos generados por el ASR. Con esto se pueden hacer pruebas individuales, para poder probar diferentes parámetros y generar transcripciones provisionales.

[10]del inglés, *Word Error Rate*

Capítulo 2

Repositorio poli[Media]

2.1. Introducción

Para llevar a cabo el trabajo se ha usado el repositorio poli[Media] de la Universitat Politècnica de València. En la siguientes secciones explicaremos qué es, qué contiene, para que sirve y cómo se ha usado.

2.2. poli[Media]

La UPV creó el sistema poli[Media] en 2007. Se trata de una plataforma web para la producción y distribución de contenidos educativos. Cada elemento de poli[Media] se compone de un vídeo donde el autor explica cierto tema con la ayuda de diapositivas o programas que podremos visualizar a un lado del mismo. El sistema también ha sido usado en otras universidades, como la Universidad de la Laguna o la Universidad de Sao Paulo [pdV] [Wika].

El portal poli[Media] recibe una media de 58.000 visitas al mes, alcanzando un total de visitas, desde junio del 2013 a junio del 2014, de 810.241. Estas visitas proceden de un total de 131 países diferentes, siendo España el más predominante.

En la Figura 2.1 podemos ver el tercer vídeo más visto de todo poli[Media]. Se ha escogido éste ya que esta transcrito automáticamente, obteniendo un total de visitas de 6.248 y una media de 550 al mes. Un total de 47 países lo han reproducido. También podemos ver que el vídeo más reproducido tiene un total de 11.288 visitas y el segundo 9921. Con ésto, y los datos totales dados antes, se puede ver que poli[Media] es un recurso muy utilizado y a tener en cuenta. Por eso resulta muy interesante que el ASR mejore para poder hacer las transcripciones mejores y se pueda extender todavía más.

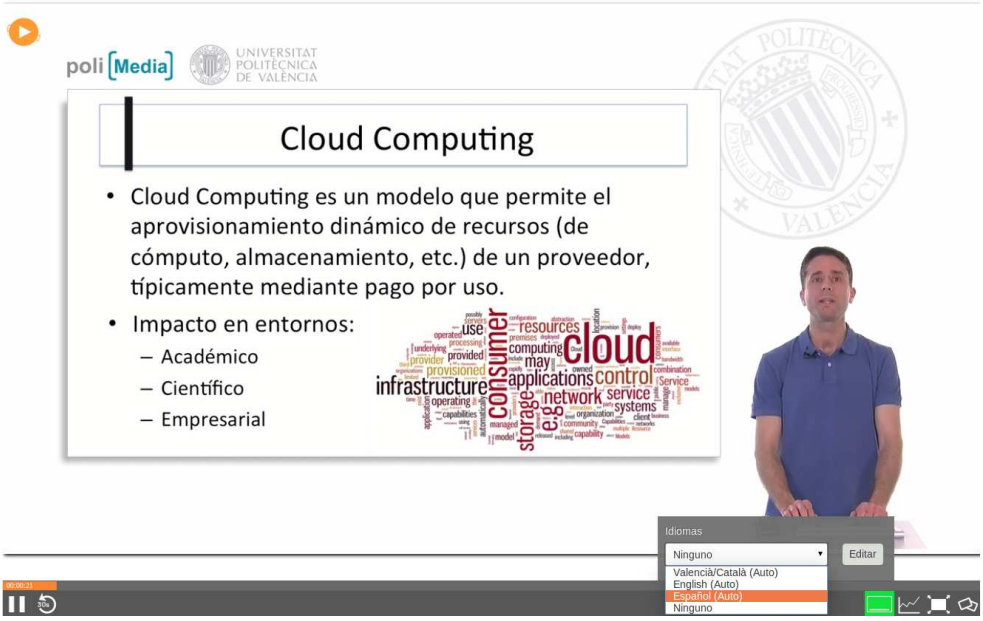

Figura 2.1: Curso Online de *Cloud Computing* con AWS

2.3. transLectures

El objetivo de transLectures es desarrollar soluciones innovadoras y eficientes para producir transcripciones y traducciones precisas en repositorios de charlas educativos, como poli[Media] o VideoLectures. Este último es un portal web, gratuito y de libre acceso a contenidos educativos, en su mayoría en inglés.

El sistema de ASR desarrollado en transLectures que es utilizado en este trabajo y descrito en el Capítulo 1, está obteniendo resultados muy competitivos y se puede considerar entre los más avanzandos.

2.4. Los subtítulos en poli[Media]

Cada vídeo cuenta con la posibilidad de activar subtítulos en tres idiomas: español, valenciano/catalán e inglés. Estos subtítulos pueden haber sido transcritos manualmente o por el sistema ASR. Los vídeos que tienen sus subtítulos transcritos automáticamente están marcados con la etiqueta *Auto*. En la Figura 2.2 podemos ver el detalle de la selección en la Figura 2.1.

En la Tabla 2.1 se presentan las estadísticas más relevantes del repositorio, que

Figura 2.2: Selector de idioma

son: el número de vídeos, la duración total de ellos, la longitud media de cada uno y el número de autores diferentes. Como podemos observar hay una gran cantidad de vídeos de una multitud de autores diferentes, lo que hará, como veremos en el Capítulo 4, que un sistema ASR general obtenga resultados modestos.

Tabla 2.1: Estadísticas del repositorio poli[Media]

Vídeos	9.222
Duración total (horas)	2.102
Duración media (minutos)	13
Autores	1.302

Solo una parte pequeña, el 5 %, ha sido transcrito manualmente. Con esta parte se ha entrenado el sistema ASR de transLectures, que es el que se ha empleado para transcribir el resto del repositorio. En las siguientes secciones se describirán detalladamente estas dos partes: la transcrita y la no transcrita.

2.4.1. Vídeos transcritos de poli[Media]

De todo poli[Media] sólo se ha transcrito manualmente una parte pequeña ya que, el coste en tiempo y económico de anotar vídeos es alto. Concretamente, el tiempo medio necesario para anotar un vídeo es de aproximadamente 10 veces la duración del mismo. Sin embargo, como se ha descrito en el Capitulo 1, para entrenar el sistema hace falta vídeos junto con sus anotaciones manuales. De este modo, en el proyecto transLecture se anotaron aproximadamente 114 horas con tal de generar un sistema ASR. Estos datos se repartieron en tres conjuntos: *training*, *development* y *test*. En la Tabla 2.2 podemos ver la cantidad específica que tiene cada conjunto, cuantos autores diferentes hay, la duración total de los vídeos, el número total de palabras, palabras y vocabulario. Hay que resaltar que los conjuntos son independientes en términos de locutor; esto es, no se repite ninguno en más de un conjunto. De este modo, se simulan las condiciones reales de este sistema; esto es, se asume que llegarán vídeos para transcribir de locutores que no hayan sido vistos por el sistema.

Tabla 2.2: Estadísticas de los conjuntos de *training*, *development* y *test*

	Training	*Development*	*Test*
Vídeos	655	26	23
Autores	73	5	5
Horas	107	3.8	3.4
Frases	39.2K	1.3K	1.1K
Palabras	936K	35K	31K
Vocabulario	26.9K	4.7K	4.3K

Pese a que parecen pocos datos, en total 114 horas, entrenando un sistema con el conjunto de *training* y ajustando los parámetros necesarios en *development*, se obtienen cifras de error de error de 13,5 de WER. Este es un resultado muy favorable, ya que como hemos descrito en el Capítulo , los sistema ASR estado del arte están entre el 10 al 40 de WER.

2.4.2. Vídeos no transcritos de poli[Media]

Este grupo se caracteriza por su gran cantidad de datos. Lamentablemente, éstos no han sido utilizados para estimar el sistema de ASR. En la Tabla 2.3 podemos ver las estadísticas de los vídeos. El número de vídeos es 13 veces más alto, lo que hace que haya casi 19 veces más horas y un total de 1156 autores más.

Tabla 2.3: Estadísticas del contenido sin transcribir

Vídeos	8567
Autores	1229
Horas	1995
Frases (estimado)	512.71K
Palabras	8.6M

En la Tabla 2.3 también se muestra el número de frases y palabras. Se ha hecho una estimación sobre estos valores debido a que, al no estar transcrito, se desconoce el valor exacto de ellos. Por lo tanto, se he aplicado una simple regla de tres respecto a los valores del conjunto transcrito, para visualizarlos y hacernos una idea sobre la cantidad de datos que queda sin utilizar.

2.4.3. Conclusión

En este trabajo se quiere hacer uso de la parte no transcrita de poli[Media] y la amplia cantidad de datos que se quedan sin utilizar. Específicamente en poli[Media]

cada ciertos tiempo, los vídeos son retranscritos y el sistema intenta medir el acierto que ha obtenido al transcribir la parte que no tiene referencia. La idea básica sería utilizar las partes que sabemos que se han obtenido automáticamente y que corresponde a la transcripción correcta. Esto lo veremos en el capítulo siguiente.

Capítulo 3

Selección de muestras de alta confianza

3.1. Introducción

En este capítulo vamos a abordar el tema de la selección óptima de muestras sin transcribir para usarlas en un entrenamiento de un sistema ASR. Las muestras destinadas al entrenamiento son un elemento clave para el reconocedor. Por ello, como ya hemos visto en la Sección 1.2, se necesitan muestras de audio junto con sus transcripciones. Típicamente, se usan datos transcritos manualmente, pero trabajos recientes sobre aprendizaje semi-supervisado [Zhu06] sugieren que se pueden emplear las transcripciones generadas por un sistema ASR para mejorar el reconocimiento.

3.2. Unidades de selección

El primer punto a estudiar es elegir qué tipo de unidad vamos a elegir para añadir al entrenamiento. Con tipo de unidad nos referimos a un conjunto de muestras. Por ejemplo, en nuestro caso, la unidad más pequeña que reconocemos son palabras y la unidad más grande son vídeos (un conjunto de palabras). Así pues, tenemos, tres tipos de unidades a estudiar: palabras, segmentos y vídeos. Básicamente, lo que haremos será elegir de las unidades reconocidas no anotadas que tenemos, las que serán elegidas para añadirlas al entrenamiento.

Cabe resaltar que en el caso de segmentos y vídeos nos referimos a una serie de palabras consecutivas. Típicamente un segmento corresponde a una muestra, que en el caso de poli[Media] suelen tener una duración media de 10 segundos, suele corresponder con frases en el discurso. Y, en el caso de vídeos, de manera similar a todas las palabras dichas en un vídeo.

A continuación mediremos, dependiendo de la unidad de selección, la confianza

que tiene el sistema en el reconocimiento de esa unidad. Ésto, se conoce como medida de confianza para esa unidad. Ésta es un número que va del 0 al 1 indicando si, probablemente, esté mal o bien reconocida, respectivamente.

En las siguientes secciones, describiremos cómo calcular esta medida de confianza para cada unidad de selección; palabra, segmento o vídeo; describiremos su aplicación en la base de datos poli[Media] y evaluaremos sus resultados.

3.3. Medidas de confianza a nivel de palabra

Dada una muestra reconocida, la medida de confianza o CM[1] es la puntuación que el sistema ASR calcula según su confianza en el reconocimiento. En este trabajo se han usado las del proyecto transLectures y que se basan en las propuestas por Wessel y Ney en [WN05]. Lo que proponen es usar la probabilidad a posteriori de cada palabra en la Ecuación 1.1 como CM. Esta probabilidad se considera una buena medida de confianza, ya que representa directamente la probabilidad de que según el modelo se haya reconocido dicha palabra.

Aun así, presenta dos problemas. Primero, hay un problema de segmentación de la palabra en la muestra y la transcripción, dado que habría que considerar todas las segmentaciones de esa palabra en esa dada. Esto se soluciona asumiendo que la CM de una palabra es la probabilidad a posteriori de esa palabra en un segmento en la mejor hipótesis. Dada una muestra representada por un vector de características $\boldsymbol{x}$ y una palabra w reconocida en el instante s a t en $\boldsymbol{x}$, su probabilidad a posteriori se calcularía como:

$$p(w|\boldsymbol{x}_s^t) = \frac{p(\boldsymbol{x}_s^t|w)p(w)}{p(\boldsymbol{x})} \tag{3.1}$$

Por último, otro problema es que en la Ecuación 3.1 tenemos el término $p(\boldsymbol{x})$, el cual es muy costoso de calcular. Para aliviar esta dificultad lo que se hace es descomponerlo como:

$$p(\boldsymbol{x}) = \sum_{w} p(\boldsymbol{x}, \boldsymbol{w}) = \sum_{w} p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w}) \tag{3.2}$$

De esta forma, hay que calcular la probabilidad de todas las transcripciones posibles. Aun así, calcular la probabilidad de todas las transcripciones posibles es una tarea complicada, además de que la probabilidad de la mayoría de las secuencias es 0. Por lo tanto lo que se hace es solo usar un conjunto pequeño de $\boldsymbol{w}$ ya que se obtienen los mismos resultados y de forma más rápida. Este conjunto corresponde al conjunto de transcripciones más probables y se puede obtener durante el reconocimiento *word-graphs*.

[1] del inglés, Confidence Mesure

3.3.1. *Wordgraphs*

Un *wordgraph* representa, de forma compacta, un conjunto de transcripciones. Cada nodo, de una muestra específica, muestra un instante de tiempo de $\boldsymbol{x}$ dada una historia, y cada arco la probabilidad de generar una palabra de un nodo a otro. Por lo tanto, tendremos un conjunto de nodos unidos por enlaces donde habrá tantos enlaces como palabras posibles se puedan reconocer.

En la Figura 3.1 tenemos el *wordgraph* de una frase. En este caso la transcripción más probable es "que existen que se den en todas", junto con otros caminos más probables, que como podemos imaginar es una versión podada de todas las posibles. El número de caminos que hay de un instante a otro refleja la calidad del reconocimiento. Si se tienen muchos enlaces quiere decir que el sistema no tiene mucha confianza, debido a que duda entre muchas posibilidades. Mientras que en el caso de que haya pocos sería al contrario, el sistema no ha tenido ninguna duda.

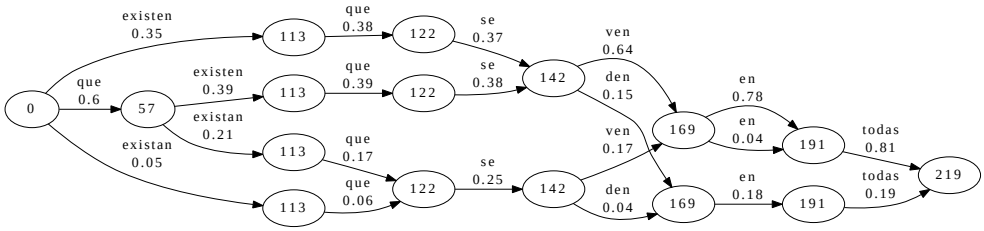

Figura 3.1: *Wordgraph* de un segmento

Por ejemplo, tomaremos como nodos y enlaces los de la Figura 3.1. En el caso de encontrarnos en un cierto nodo, calcularíamos la probabilidad posteriori de ese nodo como la probabilidad de todos los caminos que contienen ese arco, dividida entre la probabilidad de todos los otros caminos. La probabilidad a posteriori de la palabra "existen" del nodo 57 al 113, se calcularía como la probabilidad de todos los que llegan al nodo 57 por la probabilidad de todos los caminos que salen del 113 dividida entre la probabilidad de todos. Para mayor comodidad, los números que acompañan a las palabras son las probabilidades a posterior ya calculadas. El algoritmo que se encarga de hacer esto se llama *forward-backward*.

Por último, mejoraremos las probabilidades a posteriori que tenemos aplicando el suavizado Naive-Bayes propuesto en [SJV03].

3.3.2. Evaluación

A continuación describiremos de un experimento para medir la efectividad de las medidas a nivel de palabra. Para ello tomaremos todas las palabras de poli[Media] como hemos descrito en la sección anterior. Para la parte anotada y no anotada, ordenaremos sus palabras por medida de confianza. En el caso de la parte anotada de

la primera palabra a la última iremos contando el número de palabras erróneas. Si ordenamos las palabras de mayor confianza a menor confianza, de ahora en adelante MCF[2], este error nos indicará cuántos errores se han producido al seleccionar cierta porción de datos de mayor confianza. De manera similar, si ordenamos de menor a mayor, de ahora en adelante LCF[3], obtendríamos cierta porción de los datos de menor confianza. Estos dos criterios nos sirven para evaluar como de efectivas son seleccionando palabras correctas o incorrectas.

En la parte izquierda de la Figura 3.2 podemos observar cómo evoluciona el porcentaje de error total al ir aumentado el número de palabras seleccionadas. Por ejemplo, si seleccionamos un 10 % de las palabras, el porcentaje total de los errores que habríamos elegido. En nuestro caso nos interesa que este número sea el menor posible. En este caso evaluamos dos criterios: el criterio MFC y el criterio aleatorio. Este último sería el resultante de haber seleccionado las palabras al azar y se considera el peor método de selección, ya que los errores van apareciendo conforme la media de error en este conjunto. Como podemos observar, el primer tramo de palabras son las que más CM tiene y menos error se da, pero según vamos aumentando la porción seleccionada, el error se incrementa. Como podemos observar nuestras medidas de confianza son mejor que el criterio aleatorio, ya que para todo porcentaje de palabras que seleccionamos, siempre se obtiene menor cantidad de error. Por otra parte, podemos observar que la selección es más efectiva cuando se elige una porción pequeña.

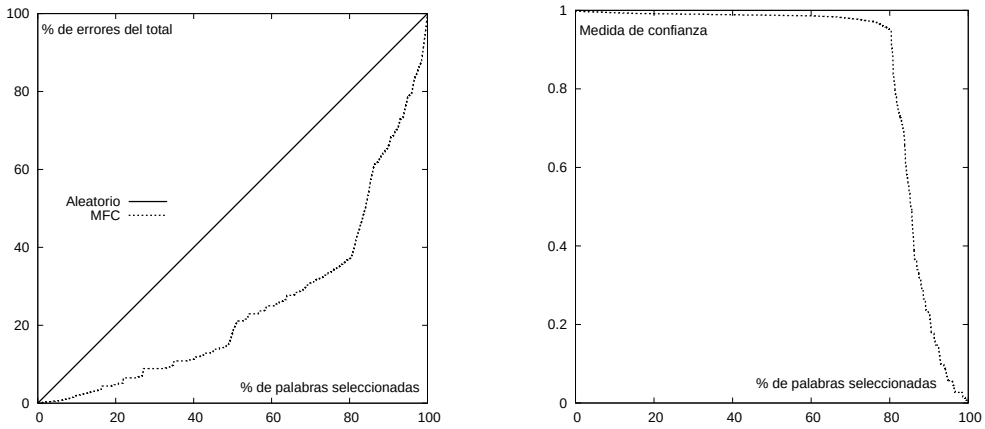

Figura 3.2: Error y medida de confianza a nivel de palabra del *development* y *test*

En la parte derecha de la figura, podemos ver, según el criterio MFC, el valor de la CM para todas las palabras del conjunto. Como hemos ordenado de mayor a menos, este valor va de 0 a 1. También podemos observar que la mayoría de palabras tienen una CM alta. Observamos que cuando baja de cierto valor pierde su capacidad de

[2]del inglés, *Most Confidence First*
[3]del inglés, *Least Confidence First*

reconocer palabras correctas. Esto es interesante porque en la parte no anotada no podemos evaluar el comportamiento del error, con lo cual las CM nos servirán para relacionar la parte anotada y no anotada y saber que porción es conveniente elegir.

En la parte no anotada, como vemos en la Figura 3.3, nos encontramos con una curva muy parecida al de la parte derecha de la Figura 3.2, por lo que nos va dando una idea de que el ASR ha hecho un reconocimiento más efectivo para las palabras de mayor confianza, y con ello podríamos añadir más datos al entrenamiento.

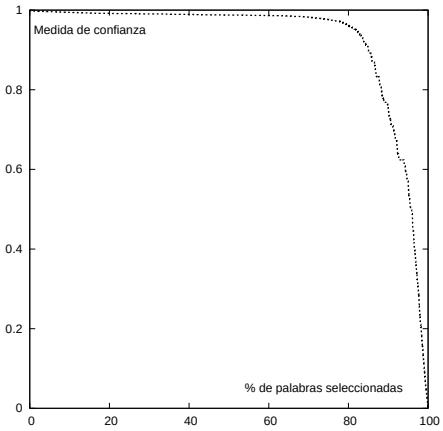

Figura 3.3: Medidas de confianza a nivel de palabra parte no transcrita

3.4. Medidas de confianza a nivel de segmento

Otra forma de seleccionar las muestras sería por segmentos, que es el caso de seleccionar un conjunto de palabras. Con esto creamos muestras un poco más grandes y reducimos así el número de muestras totales. Para calcular la CM de los segmentos, combinamos las medidas de las palabras de cada segmento. Proponemos cuatro formas distintas de hacerlo. Para ellos se harán los diferentes cálculos y luego se elegirá el mejor. Este se ha realizado usando diferentes métodos: media, media geométrica, máximo y umbral. Si tenemos el conjunto de T CMs a nivel de palabra $\boldsymbol{x} = x_1, \ldots, x_T$, la CM a nivel de segmento s para el caso de la media se calcularía como

$$s = \frac{1}{T} \sum_{i=1}^{T} x_i \tag{3.3}$$

en el caso de la media geométrica

$$s = \left(\prod_{i=1}^{T} x_i \right)^{1/T} \tag{3.4}$$

para el máximo hacemos

$$s = \max \boldsymbol{x} \tag{3.5}$$

y para calcular el umbral, definimos un valor mínimo u para que pueda puntuar y entonces tenemos

$$s = \frac{1}{T}\sum_{i=1}^{T} x_i, x_i = \begin{cases} 0 & x_i \geq u \\ 1 & x_i < u \end{cases} \tag{3.6}$$

Este valor u se tiene que optimizar en un conjunto de validación.

3.4.1. Evaluación

En la Figura 3.4 podemos observar cómo evoluciona el porcentaje de error total al ir aumentado el número de segmentos seleccionados con la CM calculada usando la media. En este caso evaluamos tres criterios: MFC, LFC y aleatorio. Como podemos observar el primer tramo de palabras son las que más CM tiene y menor error, pero según vamos incrementado la porción seleccionada, el error se incrementa. En el caso del MFC, las medidas de confianza son mejor es que el aleatorio, al contrario que el LFC que siempre se mantiene superior. Esto es debido a que esta técnica está orientada a detectar errores. Por otra parte, la selección más efectiva se da cuando se elige una porción pequeña.

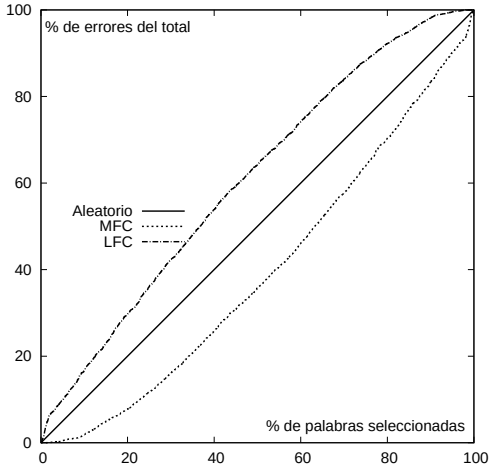

Figura 3.4: Media a nivel de segmento de la parte no anotada

La Figura 3.5 corresponde al cálculo de la CM en el caso del máximo. De manera similar, el criterio MFC sigue siendo el mejor pese a mantenerse paralelo al caso aleatorio en un 10 %. Comparado con la media, el máximo es la peor medida de confianza. Esto es debido a que, en un mismo segmento, hay muchas palabras con CM muy bajo y esté sólo se queda con el máximo, por lo que se debería de haber descartado.

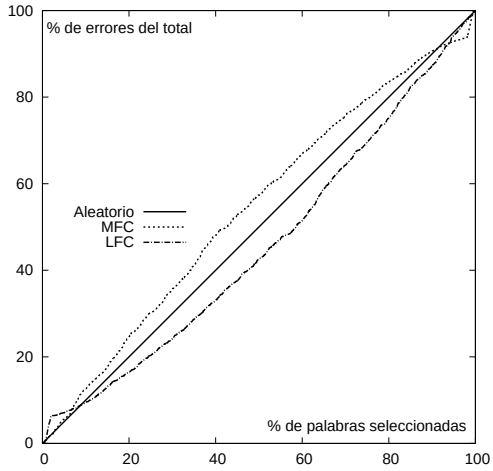

Figura 3.5: Media y máximo a nivel de segmento de la parte no anotada

En la Figura 3.6 se muestra el cálculo del CM usando la técnica del umbral, previamente optimizado. Como podemos observar el comportamiento de esta medida de confianza es prácticamente igual que la media.

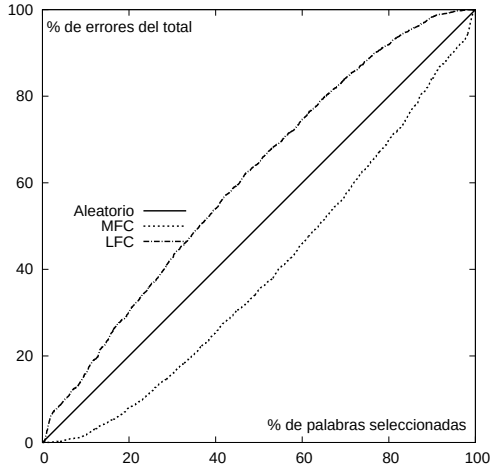

Figura 3.6: Umbral a nivel de segmento de la parte no anotada

En la Figura 3.7 se ha usado el cálculo de la media geométrica y, al igual que en las demás, el MFC sigue siendo el mejor. Podemos ver que el comportamiento es bastante similar al de la media, pero en este la curva se encuentra un poco más alejada del caso aleatorio.

Comparando todas las gráficas, las que mejores resultados ofrecen son la de la

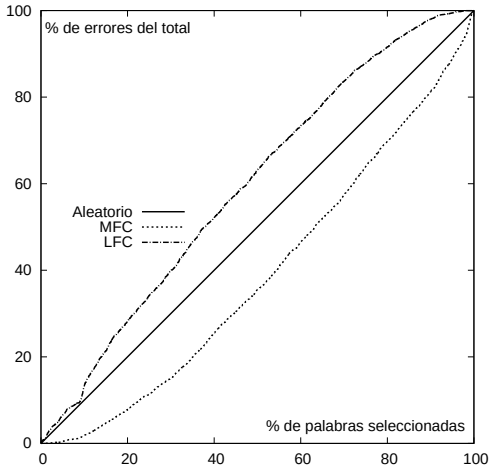

Figura 3.7: Media geométrica a nivel de segmento de la parte no anotada

media y media geométrica, que son bastante similares. Aun así, la media geométrica es mejor, ya que el error está más alejado del aleatorio. Por lo tanto, la medida de confianza elegida para los segmentos es la media geométrica.

En la Figura 3.8 podemos ver la evolución de la CM de los segmentos para el criterio MFC para la parte anotada (izquierda) y no anotada (derecha) conforme vamos seleccionando los segmentos. Podemos ver, en la parte anotada, la evolución del CM no empieza a descender hasta pasados el 10 %. En la parte no anotada la curva se mantiene más alta durante un periodo más largo. Con ello podemos ver que el error se comportará de manera similar o incluso ser un poco menor, ya que no tiene ese descenso y ésto implica que se mantenga el error próximo a 0 durante un periodo mayor.

3.5. Medidas de confianza a nivel de vídeo

La última técnica que se ha considerado para seleccionar las muestras consiste en basarnos directamente en los vídeos. Estos vídeos pueden verse como un conjunto de segmentos. Estas muestras son las más grandes y también las que forman el conjunto más pequeño. Como en el paso anterior, las calcularemos combinando las palabras que las forman, y de manera similar probaremos distintas técnicas para combinarlas. Los distintos casos que vamos a probar son los mismos que para los segmentos; esto en, la media, máximo, umbral y media geométrica, pero utilizando todas las palabras del vídeo.

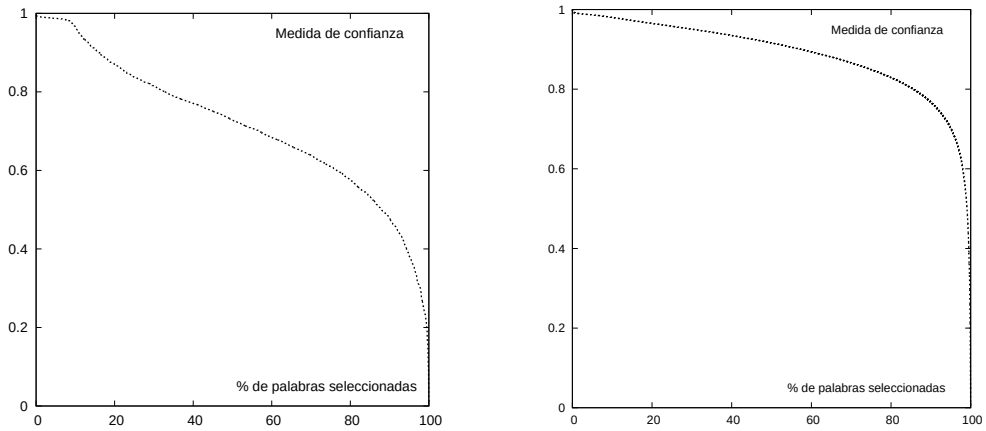

Figura 3.8: Medidas de confianza parte anotada (izquierda), parte no anotada (derecha) a nivel de segmentos

3.5.1. Evaluación

En la Figura 3.9 podemos observar cómo evoluciona el porcentaje de error total al ir aumentando el número de vídeos seleccionados, en cuanto a CM calculado usando la media. En este caso evaluaremos dos criterios: MFC y aleatorio. Como podemos observar, el error se mantiene casi siempre por encima del caso aleatorio, y entre el de un 70 % y el 80 % desciende ligeramente por debajo de este.

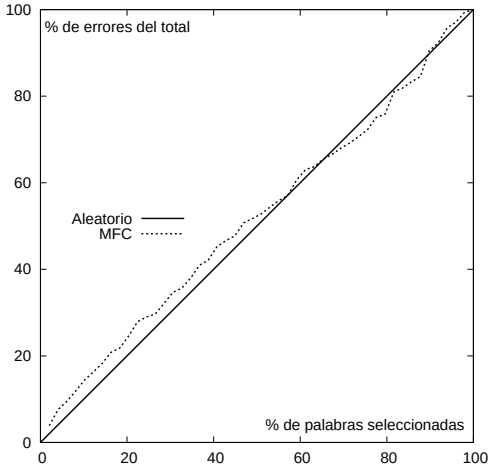

Figura 3.9: Media a nivel de segmento de la parte no anotada

La Figura 3.10 corresponde al cálculo del CM en el caso del máximo. De manera similar, el criterio MFC se mantiene siempre superior. Este resultado y el anterior

dan tan malos resultados ya que a la hora de seleccionar entre la cantidad de vídeos que hay, casi todos estos vídeos tienen un error aproximado, por lo que el error sería muy similar al aleatorio.

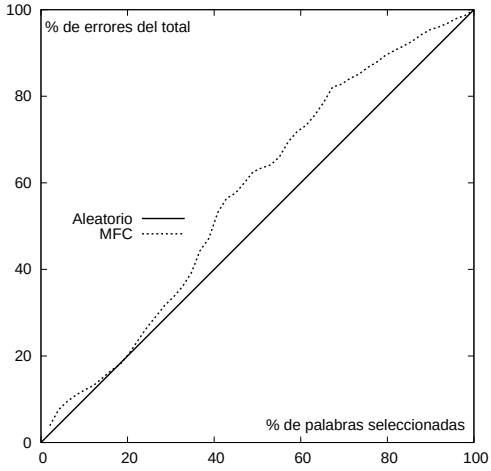

Figura 3.10: Media y máximo a nivel de vídeo de la parte no anotada

En la Figura 3.11 muestra el cálculo del CM usando la técnica del umbral, previamente optimizado. Como podemos observar se mantiene superior al caso aleatorio hasta el 65 % y tras eso desciende levemente manteniéndose por debajo del mismo.

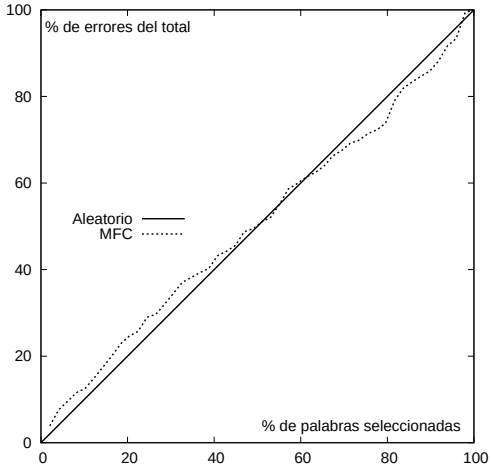

Figura 3.11: Umbral a nivel de vídeo de la parte no anotada

En la Figura 3.12 se ha usado el cálculo de la media geométrica, que, como podemos ver se mantiene en todo momento por debajo del caso aleatorio incluso descendiendo

poco a poco según va avanzando. Esto es debido a que el cálculo de la media geométrica tiene en cuenta las muestras que peores resultados tienen. De todas maneras, para la parte que nos interesa el error es similar al aleatorio.

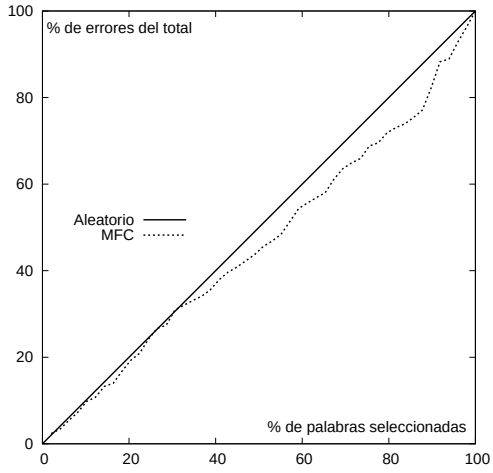

Figura 3.12: Media geométrica a nivel de vídeo del *development* y *test*

Comparando todas las gráficas, la que mejores resultados tiene es la media geométrica. Por lo tanto, la medida elegida para los vídeos es la media geométrica.

En la Figura 3.13, podemos ver la evolución de la CM de los vídeos, para la parte anotada (izquierda) y no anotada (derecha) conforme vamos seleccionando los vídeos. Podemos observar, en la parte anotada, el valor inicial del CM es casi 0,8 dado que es muy difícil que se hayan reconocido vídeos a la perfección. Este valor va descendiendo poco a poco. En la parte no anotada, la curva se mantiene más alta pero, al igual que la anotada, va descendiendo de la misma manera pero más lentamente.

3.6. Selección de datos

Una vez analizados cada nivel y eligiendo la media geométrica, como la mejor forma de calcular la probabilidad a nivel de segmento y vídeo, hemos observado que la mejor forma de seleccionar los datos sería usando las muestras a nivel de palabra o segmento. Comparando los dos, el nivel de palabra es un poco mejor que el de segmento, pero obtener las muestras a nivel de palabra requeriría obtener los marcos de tiempo de cada palabra en cada vídeo, dado que es una tarea muy costosa. Aun así, para el pequeño porcentaje que vamos a escoger, ya que si elegimos mucho introduciríamos error, por simplicidad, se escogen segmentos, ya que se pueden escoger directamente y al elegir palabras implicaría un trabajo adicional porque habría que recortarlas.

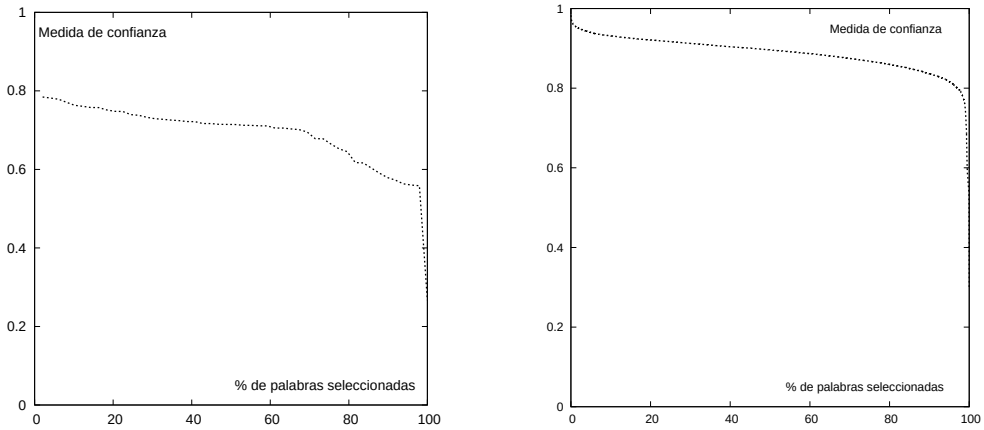

Figura 3.13: Medidas de confianza parte anotada (izquierda), parte no anotada (derecha) a nivel de vídeo

Para ir por un camino seguro, se ha decidido elegir un porcentaje inferior al 6 % de los segmentos no transcritos, ya que por lo que hemos visto en los CM de la parte anotada no hay error y con ellos nos aseguramos de añadir muestras muy posiblemente correctas. En el siguiente capítulo aplicaremos la selección elegida a un caso de estudio real y evaluaremos su impacto en la transcripción de nuevos vídeos.

Capítulo 4

Caso de estudio en transLectures

4.1. Introducción

En este capítulo describiremos el estudio aplicado a un caso real. En esta aplicación, dado un conjunto de datos sin anotar, se seleccionan aquellos de mayor calidad para formar parte de un nuevo entrenamiento del sistema ASR. En nuestro caso hemos evaluado esta aplicación en la tarea de prueba del proyecto transLectures sobre poli[Media]. En concreto, se trata de una evaluación empírica sobre la parte transcrita de poli[Media]. Entrenaremos un sistema ASR con los datos transcritos y añadiremos un porcentaje de la selección de nuestra aplicación. Después, compararemos los resultados obtenidos con el sistema ASR de transLectures. Por último, discutiremos estos resultados y presentaremos unas conclusiones.

El capítulo se encuentra dividido en cuatro secciones: en la primera hablaremos de cómo extraer los datos nuevos; luego hablaremos de cómo procesarlos para ser entrenados; tras eso cómo hacer un reconocimiento y optimizarlo; y, por último, cerraremos con los resultados obtenidos. El capítulo se cerrara con una breve conclusión.

4.2. Selección de nuevos datos para el entrenamiento

Para poder seleccionar los nuevos datos, se hizo un estudio del error y la medida de confianza de la parte anotada y no anotada. De ese modo, se puede ver cómo evolucionaban dichos parámetros. Se observó que a nivel de palabra y segmento es como mejor resultados se obtenía. Se descartó el nivel de palabra, dada su complejidad para obtener las muestras de ellos.

Para ir por camino seguro, de todas las muestras que se pueden obtener de los segmentos, sólo tomaremos una parte pequeña, para asegurarnos que introducimos

palabras correctamente reconocidas. Concretamente, tomaremos un 2 % de éstas, dado que no debería haber prácticamente errores en ellas y en total es una cantidad muy significativa.

En la Tabla 4.1, tenemos el número de vídeos, la cantidad de autores, y el total de horas de los vídeos, tanto de la parte que esta anotada, como la parte anotada más el 2 % de los mejores segmentos de la parte sin anotar.

Tabla 4.1: Estadísticas del contenido transcrito

	Anotado	Anotado + 2 %
Vídeos	655	5.013
Muestras	39K	55K
Horas	107	147

Al final, de este 2 % obtenemos un conjunto de vídeos y muestras muy alto, y añadimos un total de 40 horas más para entrenar. A continuación se describirá los pasos de entrenamiento y reconocimiento usando la parte anotada más el conjunto extra nuevo.

4.3. Entrenamiento

Para este paso entrenaremos el sistema ASR haciendo uso de un nuevo modelo acústico y usando el mismo modelo de lenguaje que había. El proceso de entrenamiento es muy costoso, y al añadir más muestras al modelo acústico hace que se complique más. En nuestro caso, el entrenamiento se ha realizado en un clúster de cálculo que se compone de 120 núcleos. Hacer los pasos previos al entrenamiento del DNN (entrenamiento de los HMM de mixturas de gausianas), cuesta alrededor de una semana. Por otra parte, el entrenamiento de la DNN es muy complejo, e incluso utilizando tarjetas gráficas de gama alta, para poder hacer los cálculos rápidamente, el tiempo necesario es de 3 días. Como podemos ver, lanzar un entrenamiento requiere mucho tiempo.

4.3.1. Problemas

Al obtener las nuevas muestras basadas en los segmentos de más alta confianza, se tomaba sólo una parte de los segmentos de cada vídeo. Por lo tanto, las muestras nuevas sólo eran un conjunto pequeño de sus respectivos vídeos. Esto genera un problema, y es que a la hora de adaptar las muestras a cada vídeo (CMLLR), si sólo tenemos un conjunto pequeño de éste, provoca que se sobre-entrenen. Por lo tanto, al sobre-entrenarse el sistema ASR, este empeora.

Para solucionar este problema, lo que se hizo fue tomar todas las muestras de los nuevos vídeos que se iban a adaptar, independientemente de si estaban en el porcentaje elegido o no. De este modo, cuando se termina el proceso de adaptación de las muestras, tomando las muestras que anteriormente queríamos usar, y así éstas no se habrán sobre-entrenado.

4.3.2. Evaluación

Al finalizar el entrenamiento, obtenemos un modelo acústico adaptado a partir de más muestras, el cual obtiene mejor error corregido de validación por parte de la DNN que el del proyecto transLectures (M30).

4.4. Reconocimiento

A la hora de hacer el reconocimiento, es interesante optimizar primero ciertos parámetros en el conjunto de *development* de poli[Media]. Por lo tanto, obtenemos los *wordgraphs* y con ellos haremos las pruebas necesarias. Los parámetros a optimizar son el GSF[1] y el WIP[2]. El GSF se encarga de asignar un peso al modelo de lenguaje, ya que las probabilidades del modelo acústico y modelo de lenguaje no están en la misma escala. De esta modo, se ajustan a que sean similares. Si el GSF es muy bajo aparecen palabras muy cortas. Si éste es muy alto, entonces aparecerán palabras más largas. El WIP se encarga de penalizar la inserción de las palabras, si esta es positiva, se penaliza la introducción de palabras nuevas. Al contrario beneficia la inserción de mismas.

Una vez optimizados los parámetros, pasamos a reconocer el conjunto de *test* con los parámetros optimizados y así obtener los resultados a nivel de WER. Como podemos observar en la Tabla 4.2 vemos qué valores se usaron en los parámetros de GSF y WIP y que WER se obtuvo.

Tabla 4.2: Comparación del reconocimiento

	transLectures	TFG
GSF	12	12
WIP	18	18
WER	13,5	13,7

[1] del inglés, *Grammar Scale Factor*
[2] del inglés, *Word Insertion Penalty*

4.4.1. Conclusiones

El mejor resultado a nivel de WER que se obtuvo fue usando los mismos parámetros que los de transLectures, ya que al optimizarlo se obtuvieron los mismos valores. Finalmente, podemos comprobar que la diferencia no es significativa, ya que empeorar solo un 0,2 de WER no supone un cambio muy grande. Aun así, este resultado sólo se obtiene usando el conjunto del *test*, por lo que aun habría que probar el sistema ASR obtenido para entrenar vídeos nuevo, dado que al haber usado parte no anotada de poli[Media], se puede concluir que el sistema obtendría mejores transcripciones. También sería interesante probar añadiendo un conjunto de datos seleccionando un 1 % y 3 % para poder observar la curva del WER. Sin embargo, como ya hemos dicho, el proceso de entrenamiento y reconocimiento es computacionalmente muy alto.

Capítulo 5

Conclusión

En este documento hemos descrito los sistemas de reconocimiento automático del habla y su uso actualmente. Estos sistemas necesitan datos anotados para poder llevar a cabo su entrenamiento, por lo que se ha creado una aplicación que obtiene nuevos datos de recursos sin transcribir y los añade al sistema.

Se ha desarrollado una aplicación que selecciona las muestras de alta calidad de datos sin transcribir. Esta aplicación funciona mediante el uso de medidas de confianza a nivel de segmentos. Después de pruebas exhaustivas se eligió la media geométrica de las palabras como medida de confianza el segmento. Una evaluación empírica en poli[Media] validó la utilidad de esta aplicación de selección.

Hemos estudiado su aplicación en la base de datos poli[Media], donde seleccionamos un porcentaje de los datos que no están anotados. Con estos nuevos datos hemos entrenado de nuevo el modelo acústico y se ha usado para hacer unas pruebas de reconocimiento que al final han dado resultados similares del sistema del proyecto transLectures. Aun así, haría falta hacer más pruebas. Sin embargo, al ser computacionalmente muy costosas, no se han podido lanzar más ya que escapaban del marco temporal de este trabajo.

Bibliografía

[CG99] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[DYDA12] George Dahl, Dong Yu, Li Deng, and Alex Acero. Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing (receiving 2013 IEEE SPS Best Paper Award)*, 20(1):30–42, 2012.

[GY07] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, 2007.

[pdV] Universitat politècnica de València. ¿que es polimedia? http://www.upv.es/entidades/ASIC/catalogo/522359normalc.html. [Online; accedido 21-Junio-2014].

[PY11] Daniel Povey and Kaisheng Yao. A Basis Method for Robust Estimation of Constrained MLLR. In *ICASSP*, May 2011.

[SDHS01] David G Stork, Richard O Duda, Peter E Hart, and DG Stork. Pattern classification. *IO/Il7*, 2001.

[SJV03] A. Sanchis, A. Juan, and E. Vidal. Improving utterance verification using a smoothed naive bayes model. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–592–I–595 vol.1, 2003.

[tUT] The transLectures UPV Team. The translectures-upv toolkit (tlk). http://translectures.eu/web/tlk.

[Vit67] Andrew J Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[VTdlH+05] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R.C. Carrasco. Probabilistic finite-state machines - part i. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, 2005.

[Wika] Wikipedia. Polimedia — Wikipedia, the free encyclopedia. http://es.wikipedia.org/wiki/Polimedia. [Online; accedido 21-Junio-2014].

[Wikb] Wikipedia. Word error rate — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Word_error_rate. [Online; accedido 2-Julio-2014].

[WN05] Frank Wessel and Hermann Ney. Unsupervised training of acoustic models for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(1):23–31, 2005.

[Zhu06] Xiaojin Zhu. Semi-supervised learning literature survey, 2006.