UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de Valencia

Desarrollo de un sistema de Reconocimiento Automático del Habla en Rumano para el subtitulado de vídeos educativos

Treball Fi de Grau

Grau en Enginyeria Informàtica

Autor: Beniamin Cristian Colompar

Tutor: Alfons Juan Ciscar

Cotutor: Albert Sanchis Navarro

Director Experimental: Joan Albert Silvestre Cerdà

Curs 2017-2018

Resumen

El reconocimiento automático del habla (ASR, del inglés “Automatic Speech Recognition”) es una de las áreas más activas dentro de la inteligencia artificial y el aprendizaje automático. En esta área viene trabajando activamente el grupo de investigación "Machine Learning and Language Processing"(MLLP) del DSIC, principalmente a través de los proyectos europeos FP7 transLectures (2011-2014) y EMMA (2014-2016), el proyecto MINECO MORE (2016-2018), así como el proyecto europeo H2020 X5gon, el cual se encuentra en su primer año de ejecución (2017-2020). El trabajo desarrollado en estos proyectos ha dado lugar a sistemas avanzados de ASR en diversas lenguas, los cuales están siendo utilizados en el subtitulado automático de vídeos educativos del repositorio oficial de la UPV, “media UPV”, así como de repositorios de otras instituciones académicas como, por ejemplo, la Universidad Carlos III de Madrid, la Universidad de Lisboa y el Instituto Hasso Plattner alemán. Si bien el grupo MLLP ha desarrollado sistemas avanzados de ASR en diversas lenguas de la Unión Europea, tanto mayoritarias como minoritarias, hasta el momento no dispone de un sistema de ASR en Rumano que pudiera aplicarse a “media UP” u otros repositorios educativos. Una de las razones por las cuales no se dispone de dicho sistema es la falta de recursos lingüísticos en Rumano. Ahora bien, recientemente ha sido posible acceder a algunas fuentes de recursos lingüísticos en Rumano, por lo que que ya es posible construir un sistema de ASR en Rumano aceptablemente preciso. El trabajo que se propone, consiste en construir un sistema de ASR en Rumano tan preciso como sea posible, siempre dentro de las condiciones del TFG. Durante el TFG, se han realizado algunos trabajos experimentales que han conducido a la construcción y la evaluación del sistema ASR en Rumano objetivo del TFG.

Palabras clave: Reconocimiento de formas, reconocimiento automático del habla, rumano, TLK

Resum

El reconeixement automàtic de la parla(ASR, del anglès “Automatic Speech Recognition”) es una de de les àrees més actives dins de la inteligència artificial i el aprenentatge automàtic. En aquesta àrea ve treballant activament el grup de investigació “Machine Learning and Language Processing” (MLLP) del DSIC, principalment a través dels projectes europeus FP7 transLectures (2011-2014) y EMMA (2014-2016), el projecte MINECO MORE(2016-2018), així com el projecte europeu H2020 X5gon, que es troba en el seu primer any de execució (2017-2020). El treball desenvolupat en aquests projectes ha donat lloc a sistemes avançats d’ASR en diverses llengües, els quals estàn siguent utilitzats en el subtitulat automàtic de

vídeos educatius del repositòri oficial de la UPV, "media UPV", així com de repositoris de altres institucions acadèmiques com, per exèmple, la Universitat Carlos III de Madrid, la Universitat de Lisboa y el Institut Hasso Plattner alemany. Si bé el grup MLLP ha desenvolupat sistemes avançats d'ASR en divèrses llengües de la Unió Europea, tant majoritàries com minoritàries, fins el moment no disposa d'un sistema d'ASR en Romanés que puguera aplicarse a "media UPV" o altres repositòris educatius. Una de les raons per les quals no disposa d'un sistema semblant és la falta de recursos llingüístics en Romanés. Ara bé, recentment ha sigut possible accedir a algunes fonts de recursos llingüístics en Romanés, per lo que que ja és possible construir un sistema d'ASR en Romanés acceptablement precís. El treball que es proposa consisteix en construir un sistema d'ASR en Romanés tan precís com siga possible, sempre dintre de les condicions del TFG. Durant el TFG, s'ha portat a terme alguns treballs experimentals que han conduït a la construcció y evaluació del sistema ASR en Romanés objectiu del TFG.

Paraules clau: Reconeixement de patrons, Reconeixement de la parla, ASR, català, comparació, TLK, Kaldi

Abstract

Automatic Speech Recognition, or ASR, is one of the most active areas in artifficial intelligence and machine learning. The investigation group "Machine Learning and Language Processing" (MLLP) from DSIC deparment is actively mainly working in this area through the european projects FP7 transLectures (2011-2014) and EMMA (2014-2016), the MINECO MORE project (2016-2018), and the europeanproject H2020 X5gon, which is in it's first year of execution (2017-2020). The work developed in this projects led to advanced ASR systems in different languages, which are used to subtitle educational videos automatically from the official repository of the UPV, "media UPV", and also other repositorys from other institutions such as "Universidad Carlos III" from Madrid, Lisbon University and the german Institute Hasso Plattner. Althought MLLP group has developed advanced ASR systems for different majority and minority languages from the European Union, it does not have any ASR system for Romanian language that can be used for "media UPV" or other educational repositorys. One of the reasons why this system is not available is because of the lack of Romanian linguistic resources. However, recently it was possible to access a few Romanian linguistic resources, so it is already possible to build an Romanian ASR system that is accurate enough. The proposed project consists in building a Romanian ASR system as accurate as possible inside the terms of the TFG. During the TFG, many experimental tasks

have been done that led to the building and evaluating the objective of the TFG which is the Romanian ASR system.

Key words: Pattern recognition, Speech recognition, ASR, catalan, comparison, TLK, Kaldi, TFG

Índice general

CAPÍTULO 1

Introducción

En este trabajo se lleva a cabo la construcción de un sistema de reconocimiento del habla para la lengua rumana para subtitular vídeos educativos. En este capítulo de introducción se introduce la motivación del trabajo y se explican todos los conceptos necesarios para entender el contexto del trabajo.

1.1 Motivación

Los sistemas de reconocimiento del habla son tecnologías del habla que procesan el lenguaje. Este tipo de tecnologías se basan en modelos estadísticos que se construyen a partir de datos en el lenguaje objetivo. Cuantos más datos tengamos a nuestra disposición para construir estos modelos, mejores resultados proporcionará el sistema.

La cantidad de datos públicos de habla contínua que hay para el idioma rumano son muy escasos. La gran mayoría de estos datos son privados y sólo se usan dentro de los grupos de investigación que se dedican a esta tecnología. No obstante, sí que se dispone de gran cantidad de datos textuales en rumano.

Hay varios sistemas de reconociminto del habla LVCSR(del inglés Large-Vocabullary Countinuous Speech Recognition) en rumano, como por ejemplo [1, 2]; pero que se sepa, ninguno de ellos usan redes neuronales profundas.

Sin embargo, en el pasado sí que han habido varias aproximaciones a sistemas ASR con redes neuronales [3, 4]; pero en este caso, son sistemas concebidos para reconocer frases de entre 4 a 6 palabras aisladas y con vocabulario reducido.

En este trabajo, se desarrolla un sistema de reconocimiento del habla en rumano con tecnología actual. Es un sistema híbrido, es decir que los modelos acústicos están compuestos por Modelos ocultos de Markov y redes neuronales profundas. En el artículo [5] se describe un sistema de reconocimiento del habla en rumano con la misma arquitectura que el sistema desarrollado en este trabajo. Es utilizado para reconocer secuencias de dígitos a través del teléfono.

El sistema de reconocimiento de habla rumana desarrollado en este trabajo, cuyo principal objetivo es generar la transcripción escrita de una señal de audio, será utilizado en el marco de la Universidad Politécnica de Valéncia (UPV) ya que se van a poder subtitular vídeos educativos.

1.2 Reconocimiento de patrones

En esta sección se explica de forma general el reconocimiento de patrones y el concepto de clasificación. En la sección siguiente, se explican los mismos conceptos pero aplicados en concreto al reconocimiento del habla. Estas dos secciones (1.2 y 1.3) y las que vendrán a continuación en este capítulo, tienen el objetivo de explicar todos los conceptos utilizados en este trabajo desde el punto de vista teórico.

El reconocimiento de patrones es un campo de investigación en el que se incluye el campo de reconocimiento del habla. El objetivo de un sistema reconocedor o clasificador es predecir el mayor número de veces posible la clase correcta de un objeto. Para ello, todos los clasificadores o reconocedores se pueden expresar mediante funciones discriminantes:

$$g_c : E \rightarrow R \mid 1 \leq c \leq C \tag{1.1}$$

donde E es el espacio de representación donde se representan los objetos, R es el resultado proporcionado por la función(un valor real) y c toma valores entre 1 y el número de funciones discriminantes de las que se dispone.

Tal como muestra la siguiente imagen donde se ve el funcionamiento general de los clasificadores, la representación de las muestras se obtiene mediante técnicas de preproceso y extracción de características. Las características a extraer son las

propiedades deseables de los objetos que proporcionan la capacidad de discriminación entre clase y clase. El sistema reconocedor se entrena a partir de estas características y sus correspondientes etiquetas de clase en el caso en el que sea aprendizaje supervisado. Por el contrario, si el entrenamiento se lleva a cabo sin las etiquetas de clase de cada muestra, se habla de aprendizaje no supervisado.

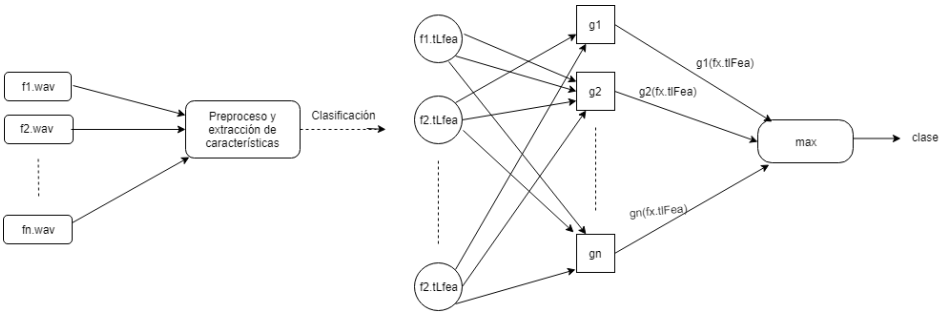

Figura 1.1: Esquema general de un clasificador

A la hora de clasificar, a cada vector de características obtenido se le aplica cada una de las funciones discriminantes y la etiqueta de la clase que se le asigna es la etiqueta de la función discriminante que proporciona el valor más grande:

$$\hat{c} = G(y) = \text{argmáx } g_c(y) \mid 1 \leq c \leq C \tag{1.2}$$

1.3 Reconocimiento automático del habla

El objetivo del reconocimiento automático del habla es proporcionar la transcripción escrita de una señal de audio. En este caso, las clases son todas las posibles frases del lenguaje y los objetos a clasificar son los vectores que contienen las características de la señal acústica. El problema se formaliza de la siguiente manera:

$$\hat{w} = \text{argmáx } p(W|X) \tag{1.3}$$

Siendo $\hat{w}$ la probabilidad estimada, W es la secuencia de palabras que compone la frase($W = w_1, w_2, ...w_n$) y X es el vector de características($X = x_1, x_2, x_3, ..., x_n$).

Aplicando la regla de Bayes, la fórmula se reescribe de la siguiente forma:

$$\hat{w} = \underset{W}{\text{argmáx}} \frac{p(X|W) \cdot p(W)}{p(X)} \tag{1.4}$$

La probabilidad del vector de características $p(X)$ no cambia con respecto a la secuencia de palabras W, por tanto, se puede eliminar de la ecuación y se obtiene el mismo resultado:

$$\hat{w} = \underset{W}{\text{argmáx}}\, p(X|W) \cdot p(W) \tag{1.5}$$

De esta forma, el modelo acústico estima la probabilidad $p(X|W)$ y el modelo de lenguaje estima la probabilidad $p(W)$. Estimando estas dos probabilidades, se logra llegar al objetivo que persigue los sistemas ASR. A continuación, se explica cómo realizan los modelos estas estimaciones con tal de cumplir el objetivo.

1.4 Modelo léxico

El modelo léxico tiene la tarea de traducir las palabras del lenguaje a fonemas. Dicho de una manera más sencilla, el modelo léxico se encarga de encontrar la pronunciación de una palabra, dada su forma escrita.

Para realizar esta tarea, se necesita un diccionario de transcripción fonética para el idioma que se precise, en este caso, el rumano. El diccionario usado en este trabajo es un diccionario llamado NaviRo. Este diccionario contiene más de 13850 palabras del diccionario DexOnline y ha sido construido de forma automática, utilizando una aplicación que se basa en un sistema constituido por 30 redes neuronales artificiales [6]. Las reglas iniciales han sido extraídas del diccionario NaviRo construido manualmente(este diccionario contiene 2383 palabras) sobre las bases de unas transcripciones realizadas por expertos lingüísticos [7].

El problema de este modelo viene cuando necesita encontrar la pronunciación de una palabra que no figura en el diccionario de transcripción fonética. En tal caso, el modelo hace uso de la herramienta sequitur G2P. Esta herramienta es un conversor de grafema a fonema dirigido por los datos y desarrollado en la RWT Aachen University - Department of Computer Science por Maximilian Bisani.

1.5 Modelo acústico

El modelo acústico tiene la función de predecir qué fonema se pronuncia en cada instante de la muestra a reconocer. Esta predicción se realiza estimando la probabilidad $p(x|w)$. Los modelos acústicos están compuestos por un conjunto de modelos ocultos de Markov o HMM(del inglés Hidden Markov Model).

Modelos ocultos de Markov

Los modelos ocultos de Markov son una forma de precisar distribuciones de probabilidad sobre un conjunto de muestras. En sistemas de reconocimiento del habla, estas muestras son los vectores de características.

Un modelo oculto de Markov se puede ver como un grafo que tiene un conjunto de estados y un conjunto de símbolos observables. En cada instante, el HMM está en un estado determinado y emite un símbolo determinado. Hay una probabilidad asociada a cada transición de un estado a otro y a la emisión de un símbolo en cada estado.

De esta forma, en el modelo acústico de este trabajo, los símbolos son los fonemas del sistema. Se dispone de un un modelo oculto de Markov de tres estados por cada fonema y dos estados especiales que son el estado inicial y el final(se representan mediante círculos de línea discontínua y doble línea, respectivamente). El modelo oculto de Markov del silencio difiere del modelo oculto de Markov del resto de fonemas ya que en este caso, sólo tiene un estado. El arco que va directamente del estado inicial al estado final representa que no se ha emitido ningún tipo de característica acústica. A continuación se muestra un ejemplo para los dos casos(figura x para el caso de HMM para un fonema y figura y para el caso de HMM para el silencio):

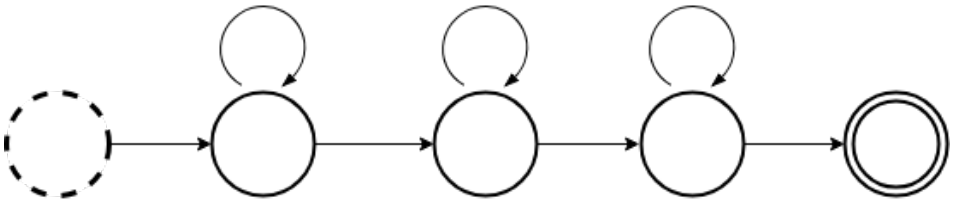

Figura 1.2: HMM de un fonema

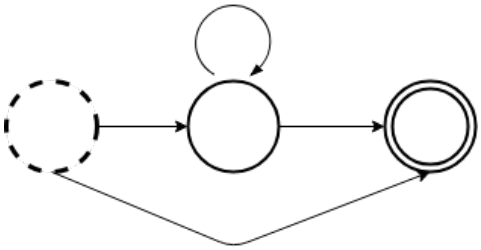

Figura 1.3: HMM del silencio

El primer estado del HMM representa el inicio de la pronunciación del fonema, el segundo estado representa la emisión del fonema y el tercer estado represen-

ta el final de la pronunciación del fonema. Hay dos estados adicionales que son especiales: el estado inicial y el estado final.

Componentes del modelo acústico

El modelo acústico se compone de varios modelos. El entrenamiento de este modelo consiste en un proceso iterativo en el que cada modelo se genera a partir del modelo anterior. Todos ellos se pueden utilizar a la hora de reconocer vídeos o audios, aqunque cada uno proporciona resultados de diferente calidad. A continuación, se explica en qué consiste cada uno de estos modelos:

Modelo de monofonemas:

El entrenamiento de este modelo consiste en generar un modelo oculto de Markov de tres estados para cada fonema del sistema. Estos HMM tienen, en cada estado, una gaussiana, que es la distribución de probabilidad de emisión de ese mismo estado. Estos HMM se entrenan con el algoritmo de *Baum-Welch* [8].

Modelo de trifonemas:

Para entrenar el modelo de trifonemas, se utiliza el modelo anterior para inicializar éste. Se entrenarán todas las posibles combinaciones de trifonemas que han aparecido en los datos de entrenamiento, lo que genera dos problemas.

El primer problema es que habrán muchos modelos pobremente entrenados, es decir, que el modelo tiene poca ocupación. Esto quiere decir que el número de muestras, en término medio, con el que se ha entrenado la gaussiana es pequeño.

El segundo problema que ocasiona el modelo es que entre todos los HMM que se han obtenido, no están todos los que pueden aparecer en el universo, es decir, que puede que haya alguna combinación de trifonemas que no esté modelada en el modelo de trifonemas. Con lo cual, ese trifonema no se podría reconocer debido a que no se dispone de un modelo suyo.

Modelo de trifonemas ligados:

Partiendo del modelo anterior, se entrena este modelo que soluciona los dos problemas que ocasiona el modelo de trifonemas. La estratégia a seguir es que para cada fonema, se crea un cluster para cada estado siendo ese fonema el fonema central.

A continuación, se va bifurcando el conjunto de trifonemas inicial en base a unas reglas predefinidas que se pueden ver como preguntas binarias. El criterio de parada de las bifurcaciones es que es posible que la ocupación del cluster a dividir esté por debajo de un umbral o que el resultado no mejora(el resultado es una forma de medir la calidad del cluster).

De esta forma, se reduce considerablemente el número de HMM que tenemos y si aparece alguno que no se ha visto en el entrenamiento, podemos saber cómo se modela por las sucesivas bifurcaciones que se han hecho(el resultado de las bifurcaciones es un árbol binario).

Modelo con mixturas de gausianas:

El modelo anterior inicializa este modelo. El entrenamiento de este modelo consiste en entrenar los trifonemas ligados de forma que, cuando acabe el entrenamiento, cada estado tendrá mixturas de gaussianas, es decir, que cada estado de los HMM tendrá 64 gausianas.

El entrenamiento es un proceso iterativo donde en cada iteración se duplica el número de componentes, pero esto implica que la ocupación de la componente inicial se divida entre las dos nuevas componentes generadas. Las componentes sólo se duplican y se vuelven a entrenar si la ocupación supera un límite establecido, con lo cual, es posible que no todos los estados tengan 64 componentes.

Redes neuronales profundas

DNN son las siglas de Deep Neural Networks, es decir Redes neuronales profundas en inglés. Las redes neuronales son una combinación de funciones discriminantes lineales con activación agrupadas en N capas ocultas de talla $M1$ y una capa de salida de talla $M2$ mas una capa de entrada de talla d.

En este trabajo, la capa de entrada es el vector de características, las capa de salida tiene un tamño igual al número total de estados de todos los HMM del sistema y las capas ocultas se han definido con un tamaño igual a 2048.

La siguiente imagen muestra un ejemplo del esquema que tiene una red neuronal:

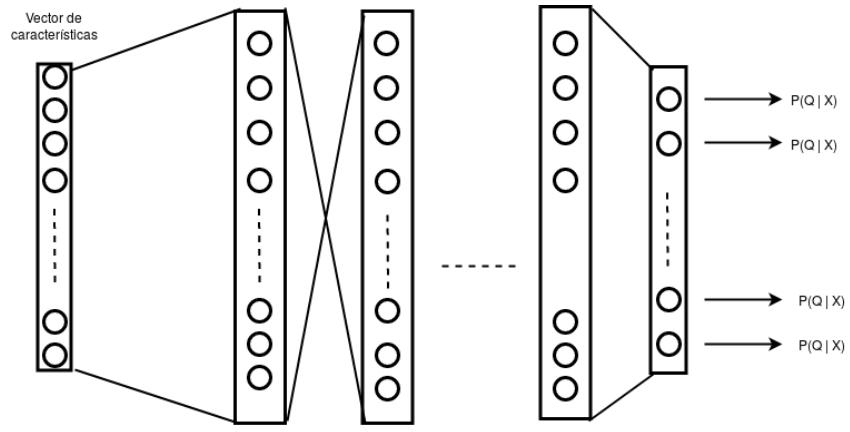

Figura 1.4: Esquema de una red neuronal multicapa

Al modelar las probabilidades de emisión de cada estado de los HMM con redes neuronales profundas, los estados de los modelos ocultos de Markov ya no disponen de mixturas de gaussianas, sinó que utilizan una red neuronal profunda que calcula la probabilidad $p(q \mid x)$ (como se puede apreciar en la imagen 1.4), es decir que el resultado que proporciona es la probabilidad que tiene el vector de pertenecer a un estado.

Para calcular esta probabilidad, se aplica la regla de Bayes:

$$P(x|q) = \frac{P(q|x) \cdot P(x)}{P(q)} \tag{1.6}$$

$P(x)$ se puede quitar igual que en 1.3, ya que su valor es el mismo para todas las muestras, con lo cual quedaría:

$$P(x|q) = \frac{P(q|x)}{P(q)} \tag{1.7}$$

donde $P(q)$ se puede obtener contando cuántas veces se pasa por el estado q durante el entrenamiento.

Una forma sencilla de entender esto es que las redes neuronales profundas tienen el papel de clasificar los vectores de características en estados de los HMM, es decir, que reciben como entrada el vector de características y como salida proporcionan la probabilidad de pertenecer a cada estado del HMM.

El entrenamiento se inicializa con el modelo de trifonemas ligados con mixturas de gaussianas. La diferencia entre el entrenamiento de este modelo y los modelos anteriores es que aquí ya no se calcula la ocupación que tiene cada estado, sino que se realiza un alineamiento.

El proceso de alineamiento es muy similar al proceso de reconocimiento descrito en la sección 4.5. Dada la referencia de cada dato de entrenamiento, se construye un grafo de búsqueda con la misma topología que el grafo de búsqueda del reconocimiento: los nodos del grafo son la história y las transiciones entre los nodos son las palabras que forman la transcripción de la referencia.

A continuación, los arcos de los nodos se convierten en el HMM equivalente. A partir de este punto, mediante el algoritmo de Viterbi se calcula el camino más probable [9, 10]. En la primera iteración se parte del modelo de mixturas de gausianas i a partir de la segunda iteración ya se dispone de DNN.

De esta forma, se consigue tener una etiqueta para cada vector de características, siendo la etiqueta el estado del HMM al que pertenece la etiqueta.

1.6 Modelo de lenguaje

El modelo de lenguaje calcula la probabilidad de una frase $p(W)$ donde W es la secuencia de palabras($W = w_1, w_2, ...w_n$) de la que se desea calcular la probabilidad.

El objetivo de este modelo es ayudar al modelo acústico a predecir mejor la transcripción del audio a reconocer, ya que las frases bien construídas tendrán una probabilidad mayor que las frases mal construídas. Por ejemplo, la frase *Hola casa dia buen* es una frase válida pero no tiene ningún sentido y por tanto tendría una probabilidad muy baja, mientras que la frase *Hola mi nombre es Pedro* es válida, tiene sentido y tendría una probabilidad alta.

El tipo de modelo de lenguaje que se ha entrenado en este trabajo es un modelo de n-gramas. La probabilidad de una determinada palabra del n-grama depende de las palabras que vienen a continuación y de las palabras anteriores(depende de su historia):

$$p(x) = p(x_1^n) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot ... \cdot p(x_n \mid x_1, x_2, ..., x_{n-1}) \quad (1.8)$$

Por tanto, si se entrena un modelo por ejemplo de cuatri-gramas, se dispone de un modelo capaz de asignarle una probabilidad a todas las posibles combinaciones de cuatro palabras de todo el vocabulario(el vocabulario se puede ver como todas las palabras que conoce el sistema), es decir, que el modelo dispondrá de 200000^4 probabilidades diferentes. Esta es la razón por la cual es una buena práctica entrenar n-gramas, porque limita la historia. En caso contrario, el modelo crecería exponencialmente.

De todas las combinaciones, habrán combinaciones que no han aparecido nunca en el corpus final(el resultado del preproceso), lo que equivale a asignarle probabilidad cero a ese n-grama. Pero que no haya aparecido en el corpus no significa que no pueda ser un n-grama probable, es decir, que hay combinaciones que no existen en el corpus, pero que pueden existir fuera del corpus. Este problema se soluciona haciendo un suavizado utilizando la técnica de *Back-off* [8].

1.7 Adaptacion del modelo al locutor

La adaptación del modelo al locutor consiste en adaptar el modelo a un locutor en concreto para que el sistema reconozca mejor muestras de ese locutor. Esto se realiza mediante una técnica que, para explicarlo de una forma sencilla, lo que hace es que todas las muestras se parezcan entre sí llevándolas a otro espacio de representación, es decir, que se normalizan las muestras. Hay dos tipos de adaptación del modelo al locutor, adaptación supervisada y adaptación no supervisada.

En la adaptación supervisada, se dispone de datos de entrenamiento del locutor y de sus transcripciones. En la adaptación no supervisada no se dispone de estos datos.

El problema que plantea esta adaptación en este trabajo es que no se van a reconocer audios o vídeos para el mismo locutor siempre, sino que no se sabe a qué locutor pertenece el video o audio a reconocer, es decir, que el vídeo o audio puede venir de cualquier lado y por tanto, no se dispone de su transcripción. Para solucionar este problema, se entrenan dos modelos. Se entrena un modelo estándard y un segundo modelo que es capaz de reconocer muestras normalizadas. Para entrenar el segundo modelo, se utiliza la técnica CMLLR. Esta técnica no adapta el modelo al locutor, sino que adapa las muestras al modelo.

La técnica CMLLR(del inglés Constrained Maximum Likelihood Linear Regression) se aplica a modelos acústicos de trifonemas ligados con una sola componente, es decir, una gaussiana por cada estado del HMM. Consiste en que se entrenará una matriz CMLLR por cada locutor por medio del algoritmo Baum-Welch [8]. Básicamente se multiplica cada vector de características por la media y la variáncia de cada gaussiana de cada estado. De esta forma, lo que se logra es que es más probable que los estados de los HMM generen los datos adaptados.

1.8 Evaluación de los resultados

Para calcular la tasa de error del sistema, y por consiguiente, para saber cómo de bien funciona, se ha utilizado el concepto WER. El WER(*Word Error Rate*), es el número mínimo de sustituciones, inserciones y borrados que se necesita para obtener la hipótesis, es decir, el resultado que obtiene el sistema a partir de la referencia. La fórmula para calcularlo es la siguiente:

$$WER = \frac{S + I + D}{N} \cdot 100 \tag{1.9}$$

siendo S el número de operaciones de sustitución, I el número de operaciones de inserción, D el número de operaciones de borrado y N el número de palabras que tiene la referencia. El resultado se interpreta de la siguiente forma: de cada 100 palabras, WER están mal reconocidas. Por ejemplo, si WER toma el valor 17, de cada 100 palabras que se han reconocido, 17 de ellas o bien se han sustituído, insertado o borrado.

Para mejorar la tasa de error, se inicia el proceso de reconocimiento con dos parámetros cuyo valor óptimo hay que encontrar, tal como se explica en la sección 4.6. El primer parámetro es el GSF(del inglés, *Grammar Scale Factor*) y el segundo es el WIP(del inglés, *Word insertion penalty*).

El GSF es un parámetro que se utiliza para dar el mismo órden de magnitud a las probabilidades obtenidas del modelo acústico y del modelo de lenguaje. La mejor forma de explicar la razón de la necesidad de utilizar este parámetro al realizar un reconocimiento es por medio de un ejemplo(todas las probabilidades están expresadas en logaritmos):

Vamos a considerar el caso en el que tenemos dos hipótesis diferentes A y B para una misma referencia. En la hipótesis A, el modelo acústico obtiene una probabilidad igual a -50 y el modelo de lenguaje obtiene una probabilidad igual a -0.003. En la hipótesis B, el modelo acústico obtiene una probabilidad igual a -45 y el modelo de lenguaje obtiene una probabilidad igual a -0.3. En total obtendríamos una probabilidad igual a -50.003 para la hipótesis A mientras que en la hipótesis B obtendríamos un total de -45.3. El sistema elegiría la hipótesis con la probabilidad más grande de las dos, que en este caso es la hipótesis A.

Esto ocasiona un problema, ya que la probabilidad obtenida por el modelo de lenguaje en la hipótesis B es 100 veces superior a la probabilidad obtenida por el modelo de lenguaje de la hipótesis A y por tanto es mucho más probable que el sistema se esté equivocando. Para realizar este equilibrio entre los resultados de los dos modelos, se multiplica el valor del GSF por la probabilidad obtenida por el modelo de lenguaje. De esta forma, suponiengo que GSF toma un valor igual a 100, obtendríamos una probabilidad igual a -75 para la hipótesis B y una probabilidad igual a -50.3 para la hipótesis A. Finalmente, el sistema elegiría la hipótesis B ya que es más probable que sea más correcta que la hipótesis A.

El WIP es un parámetro que balancea la cantidad de palabras. Sin este parámetro, los sistemas suelen insertar demasiadas palabras. El objetivo que tiene este parámetro es controlar la compensación entre la inserción y el borrado de palabras. De esta forma, a medida que el WIP se hace más más negativo, ocurren más errores por eliminación y menos errores por inserción.

La búsqueda de los parámetros óptimos se realiza en dos pasos:

- en el primer paso, se realiza un reconocimiento con diferentes valores para GSF y un modelo de lenguaje reducido. La razón de utilizar un modelo de lenguaje reducido es que el reconocimiento se lleva a cabo rápidamente y requiere pocos recursos. El proceso es: reconocimiento con muestras sin adaptar con un gsf pre establecido(GSF0 para diferenciarlo), reconocimiento con muestras adaptadas y con el mismo GSF0. Este paso genera una primera salida no optimizada.

- en el segundo paso se realiza el rescoring"que consiste en que se utiliza la salida del primer paso para aplicar los parámetros y mejorar el resultado.

CAPÍTULO 2

Recursos de habla y de texto

2.1 Introducción

En este capítulo se describen los conjuntos de datos que se han utilizado en el trabajo. Cabe destacar que no ha sido fácil conseguir conjuntos de dados de habla en rumano y, de hecho, todos los conjuntos que se van a describir a continuación se han obtenido con un esfuerzo considerable. El conjunto que tal vez ha sido el más fácil de conseguir és el conjunto que denominamos *RODIGITS*, que se va a describir en la sección 2.2 - en el punto 1.

2.2 Corpus de audio

1: RODIGITS

Este corpus fue creado por el laboratorio de investigación Speech and Dialogue (SpeeD). Las grabaciones fueron hechas bajo diferentes condiciones (varios micrófonos y varios sistemas de grabación de audio), utilizando una aplicación de grabación de audio online desarrollada por el mismo grupo de investigación. Los locutores fueron principalmente estudiantes de la facultad de electrónica, teleco-

municaciones y tecnologías de la información de la Universidad Politécnica de Bucarest(UPB).

El corpus está compuesto por 15389 ficheros de audio donde participan 154 locutores (hablantes nativos rumanos). Cada fichero de audio contiene la emisión de 12 dígitos aleatorios entre el 0 y el 9 en rumano. En general hay 100 ficheros de audio por locutor. Hay unas cuantas excepciones: para 11 locutores, el corpus contiene sólo 99 ficheros de audio por locutor. En total, el corpus contiene 38 horas de habla con una media de 8.7 segundos por emisión.

RODIGITS está dividido en 3 partes: entrenamiento, dev y test: *parte de entrenamiento:* 11120 ficheros - 80 ficheros donde participan 139 locutores. *parte de dev:* 2780 ficheros - 20 ficheros donde participan 139 locutores. *parte de test:* 1489 ficheros - 100 ficheros donde participan 15 locutores.

2: SWARA

Este corpus es un gran conjunto de datos paralelo de lectura en lengua rumana. Es el resultado del proyecto SWARA, fundado por el ministerio rumano de educación bajo el acuerdo garantizado PN-II-PT-PCCA-2013-4 No 6/2014. Está disponible tanto para uso académico como para uso comercial.

El objetivo principal de este corpus es habilitar a personas discapacitadas, especialmente a los que tienen afonia quirúrgica, a uitilizar un sistema de síntesis de texto a voz rápido y personalizado.

El corpus contiene más de 21 horas de grabaciones de alta calidad de 17 locutores diferentes. Los datos están divididos en 19279 emisiones, incluyen sus transcripciones ortográficas y alineaciones semi automáticas a nivel de fonema y están manualmente segmentados a nivel de emisión y semi-automáticamente etiquetados a nivel de fonema. Cada locutor lee entre 921 y 1493 emisiones. De todas estas emisiones hay 880 emisiones comunes a cada locutor junto con más de 16 horas de datos paralelos.

El corpus fue grabado en un estudio con equipo de alta gamma y que además, dispone de una cabina de grabación y una estación externa de monitorización. La cabina de grabación es una habitación completamente isolada a prueba de sonido que sólo contiene el micrófono, un conjunto de auriculares para la comunicación y un monitor de solicitud.

Para consultar más detalles sobre el proceso de grabación del corpus o sobre cómo se ha realizado la segmentación de los datos, consultar el documento [11].

La tabla 2.1 indica la distribución de los locutores en el corpus.

Tabla 2.1: Contenidos del corpus SWARA

Nº.	Speaker ID.	Sexo	Duración	Nº de emisiones
1.	BAS	F	1h 34' 30"	1493
2.	CAU	F	1h 11' 35"	996
3.	DCS	F	1h 50' 01"	1493
4.	DDM	F	1h 09' 18"	996
5.	EME	F	1h 53' 36"	1493
6.	FDS	M	0h 57' 21"	996
7.	HTM	F	1h 06' 27"	981
8.	IPS	M	0h 58' 08"	996
9.	PCS	M	1h 08' 03"	996
10.	PMM	F	1h 01' 53"	921
11.	PSS	M	1h 27' 45"	1486
12.	RMS	M	1h 08' 65"	996
13.	SAM	F	1h 43' 31"	1493
14.	SDS	M	1h 01' 28"	996
15.	SGS	M	0h 55' 22"	994
16.	TIM	F	1h 09' 27"	973
17.	TSS	M	1h 01' 54"	996

2.3 Corpus de texto

1: Europarl

Europarl es un corpus de texto paralelo, es decir, texto emparejado con su traducción a un segundo lenguaje. Ha sido coleccionado por un grupo de investigación de la escuela de informática de la Universidad de Edinburgo de Escocia, a partir de las actas del Parlamento Europeo que están publicadas en la web. Dicho grupo realizó la recolección de este corpus para ayudar a su investigación de traducción estadística automática.

La adquisición de semejante corpus cuyo objetivo es ser usado en un sistema de traducción estadística automática típicamente requiere 5 pasos: obtener los datos en formato raw(proceso de rastrear la red), extraer y mapear pedazos paralelos de texto(alineación de los documentos), separar el texto en frases(separación), preparar el corpus para este tipo de sistemas(normalización) y por último, alinear las frases de un idioma con las frases del idioma a traducir(alineación de las frases). A continuación se describirán los 5 pasos con más detalle.

El proceso de rastrear la web se realiza con una herramienta que empieza en una página índice y sigue ciertos enlaces basándose en reglas de inclusión y exclusión.

La página de inicio es una página de la web del Parlamento Europeo. Esta web proporciona las actas del Parlamento Europeo en forma de ficheros HTML que contienen, cada uno, las emisiones de un mismo locutor.

El siguiente paso, que es el paso de alineación de los documentos, consiste en identificar los textos que pertenecen a cada tema que se trata en la web y agruparlos por idiomas. Luego, se requiere extraer sólo el texto relevante de los ficheros HTML citados en el paso anterior, ya que contienen muchas fuentes de errores, como puede ser estándares que han cambiado de formato debido a que estos datos han sido creados durante un período de varios años.

Acto seguido, llega el paso de separar el texto en frases. Un problema de esta fase es la ambigüedad del punto "." ya que se puede usar tanto para denotar el final de una frase como para denotar una abreviación. Para solucionar este problema, se pueden utilizar técnicas como por ejemplo máxima entropía. A continuación, viene la fase de normalización, donde se prepara el texto para la tarea específica de entrenar un sistema de traducción estadística automática.

Finalmente, se llega al paso de alineación de las frases que es realizado utilizando una implementación del algoritmo de Gale y Church. Las frases alineadas se almacenan en un fichero por día por cada idioma.

Para más información sobre este corpus, sobre cómo fue generado o sobre el tipo de sistemas para los que fue concebido, se puede consultar [12].

2: EUbookshop

El corpus fue recolectado y convertido a un formato útil para el procesamiento natural de lenguaje. Las personas que recolectaron el corpus lo utilizaron para entrenar sistemas estadísticos de traducción estadística automática.

Este corpus fue recolectado rastreando el sitio web de EUbookshop utilizando una herramienta de reastreo de la web personalizada. El sitio web de EUbookshop tenía una funcionalidad de búsqueda que devolvía una lista con todos los documentos y, aprovechando esta funcionalidad, la herramienta recorría todas estas páginas y creaba una lista completa de publicaciones disponibles del sitio web. A continuación, la herramienta descargaba y guardaba los títulos de todas las publicaciones disponibles en aquel momento y luego los procesaba para obtener una lista de *URL's* de los documentos PDF de todas las publicaciones, para luego ser usada para obtener dichos documentos mediante el comando *wget*.

Una vez obtenidos estos documentos PDF, se extrajo el contenido textual de todos ellos por medio de 4 pasos principales: convertir los documentos PDF a texto plano o XML, limpiar y filtrar los datos, preproceso lingüístico y alineación de las frases. Estos pasos fueron necesarios para poder solucionar una serie de problemas, como

por ejemplo que en algunos idiomas no se podía detectar bien los límites de las palabras o de los párrafos.

El resultado final incluye un corpus con más de 40 idiomas diferentes con alineación de frases para todos los posibles pares de idiomas. También incluye corpus monolingües para todos los lenguajes para hacer más sencilla la tarea de entrenar modelos de lenguaje, que de hecho, es ésta la finalidad con la que se ha utilizado este corpus en este trabajo.

Para más información sobre el corpus, sobre los detalles de cómo ha sido recolectado, sobre la evaluación de su cualidad como corpus manual o sobre la evaluación de cómo de adecuado es el corpus para ser usado para traducción automática estadística, ver la referencia [13].

3: Opensubtitles

Este corpus, como su nombre indica, fue generado a partir de subtítulos de películas y de la televisión. Estas fuentes son fuentes literarias bastante ricas, ya que contienen desde lenguaje coloquial hasta discursos expositivos y narrativos.

Para originar este corpus, se han seguido una serie de pasos similares a los pasos que se han seguido para originar los demás corpus de texto usados en este trabajo de fin de grado: un primer paso de preproceso de los ficheros de cada subtítulo que se divide a su vez en cuatro sub pasos. El primer paso de la fase de preproceso es obtener la mejor codificación para los ficheros; el segundo paso es dividir los ficheros en bloques de sentencias tokenizadas. Cada bloque proporciona el instante de inicio y fin. El tercer paso es corregir los errores de ortografía y los errores del reconocimiento óptico de caracteres, ya que se utiliza esta técnica para extraer de forma automática una cantidad importante de los subtítulos del corpus. Por último, se generan los metadatos relacionados con cada fichero de subtítulos.

El segundo paso se lleva a cabo con el fin de obtener un corpus paralelo(en el punto x se explica qué es un corpus paralelo). Se trata de alinear los subtítulos de un idioma con otro idioma. Para consultar información detallada sobre cómo se realiza este paso, o sobre el paso anterior, consultar el artículo [14].

En este trabajo, sólo se ha utilizado el conjunto de datos monolingüe del idioma rumano para entrenar el modelo de lenguaje que se explica en el punto/sección 4.3. Los autores del artículo citado en el párrafo anterior han utilizado el corpus, tanto los datos monolingües como los bitexts(el resultado del alineamiento de los subtítulos de un idioma con otro, como por ejemplo inglés-español), para entrenar un sistema estadístico de traducción automática cuyo objetivo fue estimar el valor de los alineamientos.

CAPÍTULO 3

Herramientas para ASR utilizadas

3.1 Introducción

En este capítulo se van a describir las herramientas que se han utilizado para entrenar los modelos acústicos y el modelo de lenguaje del trabajo.

3.2 TLK

Esta herramienta fue desarrollada por el grupo de investigación MLLP del DSIC. Mediante esta herramienta, se puede hacer todo lo necesario para llevar a cabo la construcción de un sistema ASR y también permite utilizar dicho sistema.

TLK se ha usado para construir varios sistemas como por ejemplo: los sistemas de reconocimiento del habla para la lengua inglesa, castellana, portuguesa, catalana, alemana, italiana, eslovena y francesa desarrollados por el grupo MLLP, al igual que otras tarea ligadas a ASR como por ejemplo adaptación de los modelos de lenguaje o revisión de traducción automática. También se ha usado para llevar a cabo el sistema de transcripción automática transLectures de los repositorios de vídeos de poliMedia y VideoLectures.NET [15, 16].

La herramienta incluye una librería(libTLK), una serie de órdenes de consola básicas y unas cuantas órdenes para el preproceso de los corpus y para realizar el entrenamiento y el reconocimiento. Estas tres órdenes son las siguientes: tLtask-preprocess(como su nombre indica, sirve para preprocesar los corpus), tLtask-train(para entrenar el sistema) y tLtask-recognize(para utilizar el sistema que se ha entrenado con la órden anterior para reconocer audios o vídeos).

3.3 SRILM

SRILM es una herramienta que fue desarrollada en Estados Unidos, en un laboratorio de California: Speech Technology and Research Laboratory SRI International. Fue concebida tanto para producir como para utilizar modelos de lenguaje para sistemas de reconocimiento del habla además de otras aplicaciones. Es una herramienta libre para uso no comercial y se puede utilizar en sistemas operativos basados en Unix o en Windows.

Esta herramienta es una colección compuesta por librerías en C++, una serie de programas ejecutables y scripts de ayuda. Todos estos componentes permiten llevar a cabo diferentes tareas como por ejemplo desarrollar y probar el funcionamiento de diferentes tipos de modelos de lenguaje basados en n-gramas, o sistemas de traducción automática entre otras. Para más información sobre las funcionalidades de la herramienta, o cómo fue diseñada, consultar [17].

CAPÍTULO 4

Sistema ASR en Rumano

4.1 Introducción

En este capítulo se describe con detalle el desarrollo, utilizando la herramienta TLK, de un sistema ASR y cómo se utiliza para reconocer el habla. Para empezar, se explicará el trabajo previo necesario para preprocesar los datos(sección 4.2), a continuación, en la sección 4.3 se explica cómo se costruye y cómo se entrena el modelo de lenguaje. Seguidamente en la sección 4.4 se describen todos los pasos para construír el modelo acústico. Para finalizar, la sección 4.6 explica cómo se evalúa el sistema.

4.2 Preproceso de los datos

Para empezar con el desarrollo del sistema, lo primero que se necesita es realizar un preproceso de los datos, tanto los datos necesarios para el entrenamiento del modelo de lenguaje como los datos de entrenamiento del modelo acústico.

Preproceso de los datos de texto

El preproceso de los datos de texto consiste en limpiar todos estos datos. En este proceso de limpieza, se transliteran todos los números a su forma escrita; se borran todos los caracteres numéricos; se pasan todos los caracteres a minúsculas(esto es necesario ya que para el modelo de lenguaje, las palabras Hola y hola son diferentes) y por último se borran todos los signos de puntuación, todos los paréntesis, todos los corchetes y todas las llaves.

Esta tarea se realiza por medio de un script que se encarga de ello. La órden para hacerlo es la siguiente:

```
$ 02-preprocess.sh --ro --translit-num --erase-int-chars --no-hesitation \
        --lowercase --erase-punc --tokenize --erase-parenthesis \
        --erase-brackets --erase-curly --join-accents --corpus $corpus \
        --out $outdiress.sh --ro --translit-num --erase-int-chars \
        --no-hesitation --lowercase --erase-punc --tokenize \
        --erase-parenthesis --erase-brackets --erase-curly \
        --join-accents --corpus $corpus --out $outdir
```

El script recibe como argumentos una serie de opciones que le indican qué tiene que limpiar y también el corpus a preprocesar(*corpus*) y el directorio de salida del corpus limpio(*outdir*).

Preproceso de los datos de audio

A la hora de preprocesar los datos de audio, lo primero que hay que tener en cuenta es que se necesita que todas las referencias(tanto las de entrenamiento como las de test), es decir, todas las etiquetas de los ficheros de audio, tienen que estar limpias. Para ello, hay que recorrer todas estas referencias una por una y analizarlas. El análisis se hace mediante el script `parse.perl`.

Una vez finalizada esta tarea, se necesita crear para todos los corpus de entrenamiento, una lista con todas las muestras de entrenamiento y otra lista con todas las etiquetas de las muestras, de forma que la línea o la posición que ocupa cada muestra en su lista coincide con la misma línea que ocupa su etiqueta en su respectiva lista. La siguiente imagen muestra el resultado final de esta parte del preproceso:

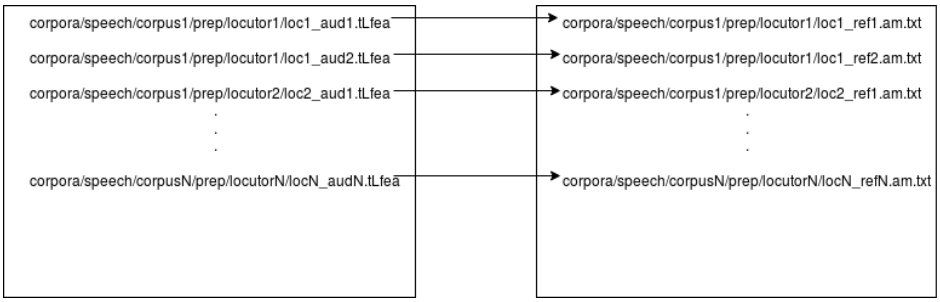

Figura 4.1: Esquema de las listas de entrenamiento

4.3 Entrenamiento del modelo de lenguaje

Para entrenar el modelo de lenguaje, lo primero que se necesita son recursos de texto. Los recursos que se han utilizado para entrenar el modelo de lenguaje del sistema son los descritos en 2.3. Estos datos necesitan un preproceso(se explica en el punto **Preproceso de los datos de texto**: 4.2), porque para el modelo de lenguaje, por ejemplo, las palabras Hola y hola son diferentes.

Finalizado el preproceso, se han juntado todos los corpus limpios en un solo corpus que los contiene a todos y se ha construido un vocabulario con un tamaño de 200000 palabras(se ha asumido que este tamaño es un buen tamaño para el vocabulario). Utilizando el corpus final y el vocabulario, se entrena el modelo de lenguaje por medio de la herramienta SRILM. Para ello, se ejecuta el script `train-nglm.sh` que recibe como argumentos el conjunto de todos los corpus, el vocabulario.

Por último, se quitan todos los cuatri-gramas que estiman una probabilidad inferior a un umbral(en este caso se ha elegido $10E^{-05}$), porque hay n-gramas que no tienen ningún sentido y es muy poco probable que aparezcan, como por ejemplo *dia hola buen*.

4.4 Entrenamiento del modelo acústico

Entrenamiento de los HMM con mixturas de gaussianas

Cuando tenemos generadas las listas con las muestras 4.1, se procede a crear un léxico de entrenamiento. Para ello, primero se necesita un vocabulario que se genera recorriendo, palabra a palabra, todas las entradas de la lista de etiquetas. Cada palabra que se recorre se guarda en una estructura de datos, de forma que al final se dispone de una estructura con las diferentes palabras(sin repeticiones) que figuran en las muestras de entrenamiento. Luego, utilizando el vocabulario, se crea

el léxico que consiste en obtener una traducción de cada palabra del vocabulario a fonemas. Para ello se utiliza el modelo léxico descrito en 1.4.

En este punto, ya se dispone de todo lo necesario(las listas con las muestras de entrenamiento, las listas con las transcripciones de los datos de entrenamiento y el léxico de entrenamiento) para entrenar la primera parte del modelo acústico. Como se ha dicho en la sección 1.5 en el punto **Componentes del modelo acústico**, el entrenamiento consiste en un proceso iterativo en el que en cada iteración se genera un modelo nuevo obtenido a partir del modelo anterior. Para ello, se utiliza tLtask-trainghmm de la siguiente forma:

```
$ tLtask-trainghmm confs/train.conf --log-folder logs
```

Esta herramienta recibe como argumento un fichero de configuración($confs/train.conf$) y un fichero de logs($logs$) donde imprime los posibles errores que puedan surgir y más detalles sobre la ejecución. El fichero de configuración contiene varios parámetros. Los más importantes para la tarea de entrenamiento del modelo son:

samples = lists/train_feas.lst es la ruta relativa(desde donde se lanza la órden de reconocimiento) de las muestras de entrenamiento.

transcriptions = lists/train_sents.txt es la ruta relativa(desde donde se lanza la órden de reconocimiento) de las transcripciones de las muestras de entrenamiento.

models-dir = models es la ruta relativa de la carpeta donde están los modelos.

states = 3 indica cuántos estados se desea que tengan los HMM.

tiedphoneme-mixture-components = 64 indica cuántas componentes se desea que tenga la mixtura de gaussianas del entrenamiento.

Adaptación al locutor

El siguiente paso que se ha dado para entrenar el sistema es realizar la adaptación de los modelos al locutor. Para ello, lo primero que se necesita es un enlace simbólico de los corpus de habla en un directorio nuevo de trabajo. A continuación, se necesita, por cada corpus, un fichero que contiene todos los ficheros con los vectores de características de todas las muestras, un fichero que contiene el locutor de cada una de las muestras del fichero anterior, un fichero que es una lista de todos los locutores y un fichero con las transcripciones de los aurdios. Por último, se vuelve a crear un vocabulario igual como se ha procedido al principio(en el primer párrafo).

Una vez se tiene todo lo necesario, se lanza el siguiente comando:

```
$ tLtask-cmllr --log-folder logs conf/cmllr.conf \
      Lists_standard-samples=lists/feas_mfcc_$NAME.lst \
      Lists_clusters=lists/mfcc_$NAME-clusters.lst \
      Lists_clusters-samples=lists/mfcc_$NAME-cluster-samples.lst \
      Lists_transcriptions=lists/mfcc_$NAME.trans \
      General_tmp-dir=tmp/$NAME \
      General_prefix-name=$NAME \
      Align_lexicon=lexicons/mono_$NAME.lex
```

Los argumentos que recibe son todos los ficheros que se han generado en este paso. Con esto lo que se consigue es que se adaptan las muestras al modelo. En la sección 1.7 se explica toda la parte teórica de qué hace TLK para adaptar las muestras al locutor.

Entrenamiento con DNN

La etapa final del entrenamiento del modelo acústico es utilizar redes neuronales profundas que tienen el papel de predecir a qué estado de todos los modelos ocultos de Markov que se dispone pertenece cada vector de características. Esta etapa se divide en dos partes. En la primera parte se entrena la primera red neuronal con muestras sin adaptación al locutor y en la segunda etapa se entrena otra red neuronal pero con muestras adaptadas al locutor.

Para entrenar la red neuronal de la primera etapa, primero se necesita realizar un alineamiento(en la sección 1.5 en el punto **Redes neuronales profundas** se explica en qué consiste). Para ello, se ejecuta la siguiente orden:

```
tLtask-align --log-folder logs confs/align.conf \
      Lists_samples = lists/feas_mfcc_$NAME.lst \
      Lists_transcriptions = lists/mfcc_$NAME.trans \
      General_tmp-dir = tmp/$NAME \
      General_prefix-name = $NAME \
      Align_lexicon = lexicons/mono_$CORPUS.lex
```

esta orden recibe como argumentos un fichero de configuración($confs/align.conf$),
una lista de vectores de características de las muestras de entrenamiento($Lists\_samples$),
las transcripciones de las muestras($Lists\_transcriptions$), y el léxico de entrenamiento($Align\_l$

Terminado el alineamiento, los datos que se necesitan para realizar el entrenamiento son: los corpus de entrenamiento, una lista con todos los vectores de características para cada corpus y los labels del align(con un pequeño preproceso que consiste en quitar la primera línea de los ficheros y cambiar la ruta). La red se entrena por medio del siguiente comando que recibe como argumentos los datos necesarios:

```
tLdnn-train \
      -read $CONF \
      -tar "$TARS" -buffer_size 209920 \
      -tsamples $TRAINC -vsamples $DEVC
```

donde $-read\$CONF$ es el fichero de configuración, $\$TARS$ son las listas, $\$TRAINC$ son los labels de entrenamiento y $\$DEVC$ son los labels del conjunto dev(es un conjunto muy parecido al conjunto de muestras de test. La diferencia es que la exploración de los parámetros óptimos se realiza sobre éste conjunto y el reconocimiento del conjunto test se lleva a cabo con los parámetros obtenidos en los experimentos sobre el conjunto dev).

4.5 Reconocimiento

Todos los modelos descritos en la sección 1.5 se pueden utilizar para reconocer audios o vídeos, pero el que mejor resultado proporciona es el modelo con dos redes neuronales con adaptación al locutor. A la hora de reconocer, el proceso es diferente de cuando se usan redes neuronales a cuando no se usan. Por tanto, en el siguiente punto se explica cómo es el proceso de reconocimiento con gaussianas y en el punto que vendrá después se explica el proceso de reconocimiento con redes neuronales.

Reconocimiento con gaussianas

Para reconocer, se utilizará uno de los modelos acústicos con gaussianas y el modelo de lenguaje. La estraégia que se sigue es la siguiente: se construye un grafo de búsqueda que calcula el camino más probable a partir de un fichero de audio.

Cada nodo del grafo es la historia de los n-gramas del modelo de lenguaje y la transición entre cada nodo es una palabra. En las transiciones están también las probabilidades que proporciona el modelo de lenguaje. Las palabras de los arcos se traducen a fonemas por medio del modelo léxico y cada fonema se modela de acuerdo con el modelo acústico que se va a utilizar. El resultado será un modelo oculto de Markov a gran escala que recibirá como entrada el vector de características de la/las muestras a reconocer.

La búsqueda se realiza por medio del algoritmo de Viterbi [9, 10]. Un ejemplo muy sencillo de cómo se calcula un camino es el siguiente: Suponiendo que el audio original se ha transformado en un vector con 3 vectores de características, que a su vez, son vectores de 48 dimensiones: la probabilidad se calcularía de acuerdo con la siguiente fórmula:

$$n = p_{Ia} \cdot p_a(x_1) \cdot p_{ab} \cdot p_b(x_2) \cdot p_{bc} \cdot p_c(x_3) \cdot p_{cF} \cdot p(w \mid H_1) \tag{4.1}$$

donde n es la probabilidad del camino, $p(w \mid H_1)$ es la probabilidad de la palabra w dada la história H_1, p_{Ia} es la probabilidad de transitar del estado inicial al estado a, p_{ab} es la probabilidad de transitar del estado a al estado b, $p_a(x_1)$ es la probabilidad de emitir el vector x_1 en el estado a, y así sucesivamente.

Reconocimiento con DNN

Cuando se reconoce con los modelos con DNN, tanto con el modelo estándard como con el modelo con adaptación al locutor, se sigue la misma estratégia. La diferencia está en que para realizar la adaptación al locutor, se realiza un reconocimiento en dos pasos. El primer paso consiste en que se realiza un primer reconocimineto con la red estándard. Utilizando este reconocimiento, se realiza la transformación CMLLR y se vuelve a reconocer con la segunda red.

En ambos casos, para calcular el camino más probable, se utiliza el grafo de búsqueda generado en el paso anterior y se calcula igual.

4.6 Evaluación

Para evaluar el sistema y ver la tasa de acierto que se ha logrado, se ha explorado el valor óptimo que deben tomar los parámetros GSF y WIP. Estos experimentos se han realizado sobre el conjunto dev de los corpus para luego reconocer el conjunto de test con los valores óptimos de los parámetros.

La primera exploración del error obtenido se ha realizado teniendo en cuenta sólo la primera parte del modelo acústico, es decir, que se ha reconocido sólo con el modelo acústico de trifonemas ligados con mixturas de gaussianas. En este caso, no se ha explorado cuáles son los valores óptimos del GSF y del WIP ya que el reconocimiento se ha realizado con la finalidad de ver si se han obtenido resultados con sentido.

Bajo estas condiciones, se ha obtenido un WER igual a 27.18 en el conjunto de dev del corpus SWARA. Este resultado es moderadamente malo a priori, pero, teniendo en cuenta las limitaciones(los pocos datos de entrenamiento y las características del modelo acústico), es un resultado mediocre acercándose a bueno.

A continuación, la prueba del sistema se ha realizado añadiendo al modelo acústico las redes neuronales (los detalles de su funcionamiento se explican en el punto 1.5 en el punto **Redes neuronales profundas**). En este caso tampoco se han buscado los parámetros óptimos. Los resultados que se obtienen en este experimento son: WER que se obtiene con muestras estándard sin rescoring, WER que se obtiene con muestras adaptadas(con reconocimiento en dos fases, explicado en la sección 4.5 en el punto **Reconocimiento con DNN**) sin rescoring y wer que se obtiene

con muestras adaptadas con rescoring. En la siguiente tabla se pueden observar los resultados obtenidos:

Tabla 4.1: WER conjunto dev SWARA

WER
20.48
18.17
17.24

Por último, se presentan dos casos en los que sí que se buscan los parámetros óptimos. En estos casos las redes neuronales que se utilizan son redes neuronales entrenadas con datos en castellano también.

Ya que son dos lenguas latinas, la gran mayoría de los fonemas son iguales a excepción de unos pocos casos y se ha asumido que utilizándolas, el sistema daría mejores resultados al no tener la limitación de recursos.

La asunción que se ha realizado es que todas las capas menos la última sirven de una manera para extraer características. Como estas redes han sido entrenadas con datos suficientes, se ha asumido que se podrían aprovechar las capas correspondientes para la tarea en rumano, ya que realizarían mejor la extracción de características. Para poder utilizarla, se ha tenido que entrenar la última capa con los datos disponibles para el rumano, ya que es ésta la que se encarga de clasificar.

La exploración que se ha realizado es la siguiente:

- valores de GSF(GSF0 para diferenciarlo con el otro) del primer paso: 9, 10, 11, 12
- valores de GSF del segundo paso: 7, 8
- valores del WIP: -8, -6, -4, -2, 0, 2, 4
- de esta forma, se realizará rescoring de todas las combinaciones posibles.

Esta es la razón por la que la exploración se realiza en dos pasos, como se explica en la seccion 1.8. En el segundo paso, el GSF0 es común y por tanto no es necesario realizar el reconocimiento para todas las combinaciones.

El primer caso es el caso donde se reconoce el conjunto Dev del corpus SWARA. Es este el caso donde se realiza la exploración de los parámetros óptimos. Los resultados se pueden consultar en la tabla 4.2. Habiendo encontrado los parámetros óptimos, se reconoce a continuación el conjunto de test del corpus RODIGITS con

los parámetros óptimos. Si se reconoce con muestras estándard, se obtiene un WER de 3.80 y si se realiza la conversión CMLLR, se obtiene un WER de 3.07. La razón por la que se obtienen resultados tan buenos es que esta tarea es extremadamente simple. El vocabulario está compuesto por 10 palabras, ya que en el corpus se pronuncian secuencias de dígitos del 0 al 9, con lo cual, es muy fácil reconocer las palabras.

Tabla 4.2: WER en función de los parámetros GSF y WIP en el conjunto Dev-SWARA

GSF0	GSF	WIP	WER
10	7	0	16.56
10	7	-2	16.47
10	8	-4	16.30
10	8	4	16.68
11	7	-6	16.90
11	8	4	17.11
12	7	4	17.99
9	7	2	16.59
9	8	-8	15.98

CAPÍTULO 5

Conclusiones

Este trabajo será utilizado por la Universidad Politécnica de Valencia para subtitular vídeos educativos en rumano y así traducirlos. En este trabajo se ha llevado a cabo el desarrollo y la evaluación de un sistema de reconocimiento del habla mediante la herramienta TLK.

En esta memoria se han explicado en general los conceptos de clasificación y reconocimiento de patrones para luego dar una explicación de los mismos en concreto para el campo de reconocimiento del habla. Se han descrito todos los modelos por los que están compuestos los sistemas ASR y se ha explicado también cómo se pueden evaluar dichos sistemas.

Se han descrito los corpus que se han utilizado para entrenar el sistema, tanto los de habla como los de texto, y las herramientas que se han utilizado para llevar a cabo la construcción del sistema.

A continuación, se han descrito todos los pasos necesarios para entrenar cada uno de los modelos del sistema. Se ha comprobado que cada uno obtiene resultados mejores que el anterior: El primer modelo(modelo acústico con trifonemas ligados con mixturas de gaussianas) obtiene un WER de 27.18(se ha medido sobre el conjunto dev del corpus SWARA). Al incluir las redes neuronales, se observan mejoras importantes: reconociendo con muestras estándard sin rescoring", se obtiene un WER de 20.48; si se realiza la adaptación de las muestras, se obtiene un WER de

18.17 y si además se reconoce con los parámetros óptimos, se obtiene un WER de 17.24 4.1.

El modelo final, donde las redes neuronales están entrenadas con datos en castellano y la capa de salida con datos en rumano es el modelo que mejores resultados obtiene. De esta manera, evaluando el sistema con los parámetros óptimos sobre el conjunto dev del corpus SWARA, el mejor resultado qeu se consigue es con un WER de 15.98 siendo los parámetros óptimos GSF(en el primer paso) igual a 9, GSF igual a 8 y WIP igual a -8.

Cabe destacar que estas pruebas se han realizado con datos grabados en buenas condiciones(en estudios de grabación con material de buena calidad), es decir que los datos no tienen ruido de fondo y la calidad de las señales de audio son muy buenas.

Se ha realizado una test del sistema con un corpus de la red, grabado por usuarios en sus casas. Es un corpus de mala calidad ya que hay archivos de audio que se escucha muy poco, archivos que se escucha la voz distorsionada, archivos que no se escucha ni una sola palabra, etc. Por eso, se ha decidido no incluir este corpus en el entrenamiento, ya que lo más probable es que ocasionaría que el sistema empeorara.

La versión del sistema que se ha utilizado para reconocer este corpus es la versión sin adaptación al locutor con la DNN entrenada con datos en rumano. En este caso, se han reconocido las muestras una a una y los resultados que se obtienen son WERS con valores como 100.0, 166.67, 10.0, 0.0, 36.36, etc.

A juzgar por los resultados, hay muestras buenas y muestras malas. Las muestras donde se obtiene un WER con un valor mayor que 100 es porque se reconocen mal todas las palabras de la referencia y además se insertan palabras que están mal en la hipótesis.

Los resultados del reconocimiento de este corpus(RASC) ha llevado a realizar un experimento con tal de comprobar si se sacan mejores resultados. Esta es la razón por la que se ha entrenado un sistema con redes multilingües. Como se ha explicado en el capítulo 4, este experimento ha dado lugar a que el sistema proporcione mejores resultados(se logra llegar a un sistema con un WER de 15.98 con datos buenos).

Bibliografía

[1] H. Cucu y col. "Recent improvements of the SpeeD Romanian LVCSR system". En: *2014 10th International Conference on Communications (COMM)*. Mayo de 2014, págs. 1-4. DOI: 10.1109/ICComm.2014.6866659 (vid. pág. 2).

[2] B. Tarján y col. "Broadcast news transcription in Central-East European languages". En: *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*. Dic. de 2012, págs. 59-64. DOI: 10.1109/CogInfoCom.2012.6421940 (vid. pág. 2).

[3] Mihaela Oprea y Daniela Schiopu. "An Artificial Neural Network-Based Isolated Word Speech Recognition System for the Romanian Language". En: () (vid. pág. 2).

[4] Daniela Schiopu y Mihaela Oprea. "Using Neural Networks for a Discriminant Speech Recognition System". En: () (vid. pág. 2).

[5] Valentin Enescu Inge Gavat Matei Zirra. "A Hybrid NN-HMM System For Connected Digit Recognition Over Telephone In Romanian Language". En: () (vid. pág. 2).

[6] Todorean Gavril Domokos József Buza Ovidiu. "Automated Grapheme-to-Phoneme Conversion System for Romanian". En: Cluj-Napoca, Romania (vid. pág. 4).

[7] Todorean Gavril Domokos József Buza Ovidiu. "100K+ WORDS, MACHINE-READABLE, PRONUNCIATION DICTIONARY FOR THE ROMANIAN LANGUAGE". En: Cluj-Napoca, Romania (vid. pág. 4).

[8] Lawrence Rabiner y Biing-Hwang Juang. *Fundamentals of speech recognition.* Prentice-Hall, 1993 (vid. págs. 6, 9, 10).

[9] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". En: *Information Theory, IEEE Transactions* 13.2 (1967), págs. 260-269 (vid. págs. 8, 26).

[10] Frederick Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, 1997 (vid. págs. 8, 26).

[11] Adriana Stan y col. "The SWARA Speech Corpus: A Large Parallel Romanian Read Speech Dataset". En: *Proceedings of the 9th Conference on Speech Technology and Human-Computer Dialogue (SpeD).* Bucharest, Romania, jul. de 2017 (vid. pág. 14).

[12] Jörg Tiedemann. "Parallel Data, Tools and Interfaces in OPUS". Inglés. En: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12).* Ed. por Nicoletta Calzolari (Conference Chair) y col. Istanbul, Turkey: European Language Resources Association (ELRA), mayo de 2012. ISBN: 978-2-9517408-7-7 (vid. pág. 16).

[13] Raivis Skadiņš y col. "Billions of Parallel Words for Free: Building and Using the EU Bookshop Corpus". En: *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014).* Reykjavik, Iceland: European Language Resources Association (ELRA), mayo de 2014 (vid. pág. 17).

[14] Pierre Lison y Jörg Tiedemann. "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". En: (vid. pág. 17).

[15] Miguel Angel del-Agua y col. "The MLLP ASR Systems for IWSLT 2015". En: (2015) (vid. pág. 19).

[16] A. Martínez-Villaronga y col. "Advances in Speech and Language Technologies for Iberian Languages: Second International Conference, IberSPEECH 2014, Las Palmas de Gran Canaria, Spain, November 19-21, 2014. Proceedings". En: ed. por Juan Luis Navarro Mesa y col. Cham: Springer International Publishing, 2014. Cap. Language Model Adaptation for Lecture

Transcription by Document Retrieval, págs. 129-137. ISBN: 978-3-319-13623-3. DOI: 10.1007/978-3-319-13623-3_14 (vid. pág. 19).

[17] Andreas Stolcke y col. "SRILM-an extensible language modeling toolkit." En: *INTERSPEECH*. Vol. 2002. 2002, pág. 2002 (vid. pág. 20).

Índice de figuras

Índice de tablas