



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica



Pattern Recognition and Human Language Technology

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Reconocimiento del habla en un sistema de ayuda a la traducción

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Ignacio Pérez Muñoz

Tutor: Francisco Casacuberta Nolla

Supervisión: Vicente Alabau Gonzalvo

21 de septiembre de 2014

Quiero dar las gracias a Vicente porque sin él este trabajo no habría sido posible, a mi tutor por ofrecerme la oportunidad de realizarlo en el PRHLT, y a mi familia y mi novia por soportarme durante todo el desarrollo de este TFG.

Resumen

La traducción automática es un campo muy complejo que difícilmente puede ser automatizado. Es prácticamente imposible crear un sistema que no cometa errores, por ello, para ayudar a corregir dichos errores se ha desarrollado la Traducción Asistida por Computador o CAT (por sus siglas en ingles, Computer-Assisted Translation). Estos nuevos sistemas CAT funcionan como un traductor automático pero con la diferencia de que el usuario puede interactuar con el sistema para corregir el texto mal traducido, de forma que se aprovecha la retroalimentación del usuario para mejorar el resto de la traducción automática

Actualmente el centro de investigación Pattern Recognition and Human Language Technology (PRHLT) ha desarrollado una herramienta de traducción CAT de próxima generación, fruto del proyecto CasMaCaT, con el cual el usuario puede corregir la traducción mediante el uso del teclado y el ratón. Este proyecto ha sido creado para mejorar la productividad y la calidad en el mundo de la traducción, aún así, el trabajo por parte de la persona encargada de la corrección, puede ser agilizado si en lugar de tener que escribir la parte mal traducida pudiera directamente dictarla al sistema.

El objetivo de este trabajo fin de grado consiste en integrar un sistema de reconocimiento del habla en el sistema CAT que dispone el PRHLT. El sistema es capaz de mejorar una traducción realizada del inglés al castellano apuntando con el ratón sobre la parte mal traducida y corrigiendo la traducción mediante la voz. Para ello se ha ajustado la interfaz web de CasMaCat para que sea capaz de capturar el audio del micrófono desde un navegador y se ha creado un servidor capaz de reconocer el audio que recibe mediante WebSockets.

Palabras clave: traducción, automática, reconocimiento, habla, corrección, CasMaCat, iATROS, WebSockets, MediaStream API

Índice general

Resumen	2
1. Introducción	5
1.1. Motivación	5
1.2. Estado del arte	6
1.3. Objetivos	6
2. Traducción asistida por computador	8
2.1. Traducción automática	8
2.1.1. Memoria de traducción	9
2.1.2. Traducción estadística	9
2.2. Predicción interactiva de la traducción	10
2.3. Interacción con el sistema	11
3. Reconocimiento automático del habla	13
3.1. Componentes del reconocedor	13
3.1.1. Preprocesado del audio	14
3.1.2. Modelo Acústico	15
3.1.3. Modelo de lenguaje	16
4. Aplicaciones utilizadas	17
4.1. WebSocket	18
4.1.1. Negociación del protocolo	18
4.1.2. WebSocket en el navegador	19
4.2. MediaStream API	20
4.3. iATROS	21
4.4. CasMaCat	22
4.4.1. Interfaz de CasMaCat	23
4.4.2. Componentes	25

5. Implementación del trabajo	26
5.1. Captura del audio	28
5.1.1. Captura del audio del navegador	28
5.1.2. Envío del stream de audio	28
5.1.3. Guardado del audio en el servidor	28
5.2. Reconocimiento del audio y retorno del resultado	29
5.2.1. Generación de Coeficientes Cepstrales	29
5.2.2. Reconocimiento del audio	30
5.2.3. Envío del texto desde Python al navegador	31
5.3. Integración con CasMaCat	31
5.3.1. Comprobación de los tokens seleccionados	31
5.3.2. Inserción de la transcripción en el navegador	32
5.4. Explotando la información contextual para mejorar el reconocimien- to	32
6. Evaluación del sistema	34
7. Conclusiones	37
Índice de códigos	38
Índice de figuras	39
Índice de cuadros	40
Bibliografía	41

Capítulo 1

Introducción

En un mundo conectado globalmente, las personas puede comunicarse y relacionarse con gente de todo el planeta. Las empresas de hoy en día pueden tener clientes de cualquier país, y uno de los mayores inconvenientes que tienen es la casi obligación de traducir todos sus textos a diferentes lenguas. En el caso de textos de grandes volúmenes, como por ejemplo manuales de usuario, puede resultar una tarea pesada y lenta incluso para un traductor experto.

Los sistemas de ayuda a la traducción surgen para solventar este problema y ayudar al traductor a realizar su trabajo más rápido. Partiendo de esta premisa, y con el objetivo de reducir aún más el tiempo de traducción surge este trabajo, en el que se integrará un sistema de reconocimiento del habla en un sistema de ayuda a la traducción.

Las herramientas utilizadas durante este trabajo están pensadas para que el usuario pueda utilizar el sistema sin la necesidad de tener que instalar ningún software adicional. El *front-end* del sistema ha sido diseñado para ser usado desde el navegador y éste viene instalado en cualquier ordenador moderno.

1.1. Motivación

Se estima que la velocidad de entrada [1], medido en palabras por minuto o ppm, del ciudadano medio para escribir a mano es de unas 30 ppm, es capaz de escribir a máquina mediante un teclado a 50 entre 80 ppm, y es capaz de hablar a una velocidad de entre 250 y 350 ppm. Como se puede apreciar, mediante el habla se puede agilizar el proceso de entrada de información de un sistema y hacer que sea más rápido de usar.

Debido al afán de mejorar la velocidad con la que el usuario de CasMaCat

corrige una frase, se ha decido utilizar el habla para mejorar las traducciones y reducir así el tiempo de traducción.

1.2. Estado del arte

Gracias a la facilidad que ofrece el reconocimiento del habla para manejar un sistema, y a la llegada de nuevos dispositivos cada vez más pequeños que carecen de teclado, el reconocimiento automático del habla está cada día más en auge y se está implantando cada vez en más aparatos. Actualmente existen multitud de herramientas que permiten realizar la transcripción del audio. Una de las herramientas más populares es el servicio de reconocimiento de voz online de Google, sistema que además integra en todos los dispositivos que llevan Android instalado.

En cuanto a la traducción asistida por computador, resulta una tarea enfocada a las empresas, no se inventó para ayudar al usuario de a pié. Dado que es una tarea menos común y está enfocado a un público más objetivo, es más difícil encontrar herramientas que desempeñen esta utilidad, pero existen diversos proyectos como por ejemplo *MateCat* (proyecto con el cual CasMaCat trabaja de forma colaborativa) o el proyecto open source *OmegaT*.

Existen pocas alternativas de sistemas que integren el reconocimiento del habla en sistemas de traducción asistida por computador. Un ejemplo de ellas es memoQ, que puede ser integrada con el reconocedor de voz Dragon [2]. La diferencia de estas herramientas con respecto a CasMaCat es que éste realiza una integración más sofisticada, ya que el sistema CAT aprovecha la información contextual para mejorar el reconocimiento y además, utiliza la retroalimentación del usuario para mejorar la traducción restante. Otra de las ventajas de CasMaCat es el hecho de que haya sido implementado bajo la tecnología Web estándar y esto permite poder ser usado sin la necesidad de instalar ningún software adicional.

1.3. Objetivos

Los objetivos principales marcados para realizar este trabajo han sido:

1. Capturar el audio del navegador y ser capaz de guardarlo sin distorsiones ni ruidos.
2. Crear un servidor que transcriba el audio utilizando iATROS y devuelva el resultado.

3. Modificar CasMaCat para que pueda capturar el audio y sea capaz de corregir la traducción mediante el texto devuelto por el servidor.
4. Hacer uso de la información contextual para mejorar el reconocimiento.

Capítulo 2

Traducción asistida por computador

La traducción de documentos de un idioma a otro no resulta una tarea fácil. Aunque muchos idiomas comparten una gran número de similitudes, existen grandes diferencias a la hora de, por ejemplo, realizar frases hechas o conjugar los verbos. Esto hace de la traducción automática una tarea prácticamente imposible, dado que no se pueden simplemente traducir palabras y esperar a que el texto sea correcto.

Debido a esta problemática y con el objetivo de reducir el tiempo de traducción surge la Traducción Asistida por Computador o CAT (por sus siglas en inglés, Computer-Assisted Translation), consistente en la traducción de documentos mediante la ayuda de *software* específico para la tarea.

En un sistema de traducción asistida por computador el sistema recoge una frase y la busca en la memoria de traducción o la traduce mediante traducción automática, después la frase original y la traducida son mostradas al usuario para que este pueda realizar una corrección manual de la traducción. A esta fase de corrección de la traducción se le conoce como post-edición y es la clave de la traducción asistida por computador.

2.1. Traducción automática

La traducción es una actividad que se lleva haciendo desde la antigüedad. Desde la época del antiguo Egipto ya se pueden encontrar indicios de traducciones gracias a la *Piedra de Rossetta*, la cual contiene tres inscripciones cada una en una lengua diferente.

Con la llegada del ordenador en los años 40, se trató de automatizar este proceso, pero pronto se dieron cuenta de la dificultad que suponía traducir

correctamente un texto de manera automática, y ya por entonces se dijo que nunca se llegaría a alcanzar con un ordenador el nivel de traducción realizado por un hombre.

Pero no es hasta los años 60 cuando se empieza hablar de traducción automática o MT (por sus siglas en inglés, Machine Translation). El almacenamiento de corpus en varias lenguas en grandes bases de datos, permite reutilizar texto ya traducido en distintos idiomas. A estas bases de datos se les conoce como memorias de traducción.

La traducción automática mediante métodos estadísticos (SMT por sus siglas en inglés, Statistical Machine Translation) está teniendo mucho auge debido a que se ajustan más a las necesidades de los clientes y son más fáciles de mantener.

También existen herramientas de traducción automática basados en otras tecnologías como las basadas en reglas (RBMT, Rule-Based Machine Translation), pero estas son menos usadas debido a que tienen un mantenimiento más costoso.

2.1.1. Memoria de traducción

Una memoria de traducción consiste en una base de datos que contiene frases o palabras en varios idiomas diferentes. Además estos textos suelen venir alineados para poder encontrar más fácilmente el texto a traducir. Debido a esta alineación a las memorias de traducción también se les conoce como *corpus* paralelos.

Para realizar una traducción, primero se segmenta el documento a traducir en partes y se comprueba si en el corpus se encuentran las frases extraídas. Si se encuentra en el *corpus*, se devuelve la traducción de ese segmento, y en este caso el grado de similitud será del 100% dado que hay una relación exacta. En el caso de que no se encuentre ninguna coincidencia, se utiliza la distancia de edición para encontrar el texto más parecido y así poder realizar de nuevo la búsqueda.

Estas memorias de traducción son utilizadas en la traducción automática para entrenar los modelos de traducción estadísticos.

2.1.2. Traducción estadística

Dado un texto en inglés x , si se quiere traducir al español y , el sistema de traducción automática deberá encontrar la secuencia y que más se aproxime

a x . Es decir la traducción \hat{y} vendrá dado por:

$$\hat{y} = \arg \max_y P(y|x) \quad (2.1)$$

Partiendo de la formula 2.1 y aplicando la regla de Bayes se obtiene la ecuación fundamental de la traducción automática:

$$\hat{y} = \arg \max_y P(y) \cdot P(x|y) \quad (2.2)$$

Donde $P(y)$ es un modelo de lenguaje e indica la probabilidad de que una frase se diga en español y $P(x|y)$ es un modelo de traducción cuyas probabilidades se estiman en base a un corpus o memoria de traducción [5].

2.2. Predicción interactiva de la traducción

Los sistemas más sofisticados incluyen además del sistema de post-edición sistemas de predicción de la traducción interactivos o ITP (por sus siglas en ingles, Interactive Translation Prediction) mediante el cual el sistema calcula de nuevo la traducción ayudándose de un prefijo que el usuario ya ha validado y marcado como correcto.

Los sistemas que incluyen ITP mejoran el proceso de post-edición creando nuevas traducciones a partir del prefijo ya validado mediante un proceso iterativo. En cada iteración el sistema intentará devolver el mejor sufijo \hat{s} , dado el prefijo p para la frase x . La concatenación de p y s dan como resultado la traducción y utilizada para la ecuación 2.1.

$$\hat{s} = \arg \max_s P(s|x, p) \quad (2.3)$$

Utilizando la fórmula 2.3 y aplicando la famosa regla de Bayes se consigue llegar a la formula:

$$\hat{s} = \arg \max_s P(x|p, s) \cdot P(s|p) \quad (2.4)$$

Donde la concatenación de p y s es la traducción completa de x . De esta manera se intenta minimizar el trabajo del usuario dado que se realiza una búsqueda de un nuevo sufijo completo. Este proceso funciona de forma similar al autocompletado, pero tiene en cuenta la información contextual apropiada

En el cuadro 2.1 se puede ver un ejemplo de la interacción del usuario en un sistema CAT. Además se muestran las iteraciones realizadas hasta traducir correctamente la frase x . En la primera iteración el sistema devuelve la frase traducida automáticamente como sufijo \hat{s} y el usuario selecciona *clíc* como

	x	Click OK to close the print dialog
iteración 0	p	
iteración 1	\hat{s}	Haga clic para cerrar el diálogo de impresión
	a	Haga clic
	k	en
	p	Haga clic en
iteración 2	\hat{s}	ACEPTAR para cerrar el diálogo de impresión
	a	ACEPTAR para cerrar el
	k	cuadro
	p	Haga clic en ACEPTAR para cerrar el cuadro
iteración 3	\hat{s}	de diálogo de impresión
	a	de diálogo de impresión
	k	(#)
	p	Haga clic en ACEPTAR para cerrar el cuadro de diálogo de impresión

Cuadro 2.1: Ejemplo de traducción de un sistema CAT sacado de [7]

la última palabra correcta y almacena en a las palabras que hay hasta dicha palabra. Además el usuario ha escrito mediante el teclado (k) que la siguiente palabra es k . Juntando el resultado de a y k se obtiene en prefijo p , con el que se calculará el mejor sufijo \hat{s} . El procedimiento se repite hasta que la sentencia es correcta y el usuario introduce una palabra de escape, que en nuestro caso es (#).

2.3. Interacción con el sistema

En el ejemplo mostrado en la figura 5.2 el usuario interactúa con el sistema CAT mediante el ratón y el teclado, pero además de estos medios básicos también hay sistemas que permiten que el usuario pueda escribir y realizar gestos mediante una tableta digitalizadora, o incluso mediante un tablet o móvil con la pantalla lo suficientemente grande como para escribir cómodamente. De este modo la persona que utiliza el sistema puede corregir las frases de una forma más natural.



Figura 2.1: Tableta digitalizadora. Fuente [8]

También existe la posibilidad de usar la voz para corregir las traducciones haciendo la corrección aún más sencilla e intuitiva. Este último método es sobre el cual está basado este trabajo final de grado y más adelante se explica el procedimiento mediante el cual se ha integrado un reconocedor de voz en un sistema CAT.

Capítulo 3

Reconocimiento automático del habla

Mediante un sistema de reconocimiento automático del habla o ASR (por sus siglas en inglés, Automatic Speech Recognition) se intenta transcribir una sentencia de audio en texto. Esta es una tarea que se ha ido desarrollando desde los años 60 hasta la actualidad, tomando cada vez más fuerza y llegando a formar parte, hoy en día, de un gran número de dispositivos y programas.

3.1. Componentes del reconocedor

El reconocimiento del habla moderno está fuertemente ligado con la estadística. La mayoría de los componentes de un reconocedor se basan en modelos estadísticos mediante los cuales se busca la mayor probabilidad de que una secuencia de voz corresponda con un texto.

De este modo, el reconocimiento del habla puede ser formulado de la siguiente manera [9]: siendo a el vector de características correspondiente a la secuencia hablada, y w un conjunto de palabras perteneciente al vocabulario del lenguaje, la mejor secuencia de palabras \hat{w} correspondiente a a queda denotado por:

$$\hat{w} = \arg \max_w P(w|a) \quad (3.1)$$

La fórmula 3.1 es difícil de modelar directamente. Para poder realizar un reconocedor más preciso se deduce la fórmula 3.1 aplicando la regla de Bayes, y dado que la probabilidad de que se pronuncie la secuencia acústica $P(a)$ es

independiente de la secuencia w se obtiene la ecuación:

$$\hat{w} = \arg \max_w P(a|w) \cdot P(w) \quad (3.2)$$

En la que $P(w)$ es la probabilidad de que la frase w sea pronunciada y viene determinado por el modelo de lenguaje, y $P(a|w)$ es la probabilidad de que a sea una pronunciación correcta de w y este valor es proporcionado por el modelo acústico.

3.1.1. Preprocesado del audio

El audio no puede ser utilizado tal cual es recogido por el micrófono debido a que contiene demasiada información. Por ello antes de ser procesado por el sistema ASR se extraen las características principales de la grabación.

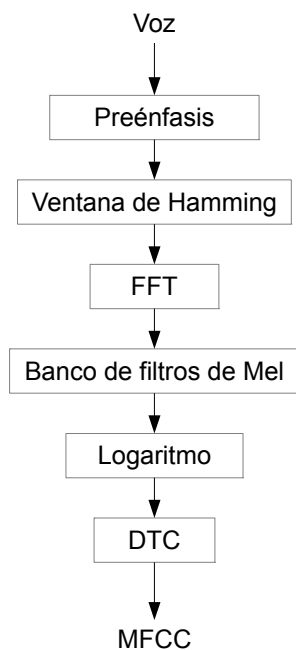


Figura 3.1: Esquema del preproceso del audio

Para poder hacer un mejor uso de los modelos acústicos se utiliza una representación continua del audio, para ello se representan mediante los llamados coeficientes cepstrales.

En el preprocesado primero se aplica el proceso de *preénfasis*, el cual consiste en aplicar un filtro paso alto al audio adquirido. A la nueva señal obtenida se le aplica una *Ventana de Hamming* devolviendo así una secuencia

de “subseñales” que se consiguen al desplazar una ventana de un tamaño determinado por la muestra. Después a cada marco o *frame* que se ha obtenido se le aplica la *transformada rápida de Fourier* o FFT (por sus siglas en inglés, *Fast Fourier Transformation*) y a continuación se pasa por un *banco de filtros de Mel* mediante el cual se filtran las frecuencias de la señal imitando la percepción humana [10]. Por último se aplica el *logaritmo* a cada uno de los vectores anteriores para reducir la sensibilidad de los sonidos fuertes y débiles, y se reducen de dimensión mediante la *transformada discreta del coseno* o DCT (del inglés Discrete Cosine Transform) para permitir un mejor reconocimiento. El resultado de todo este proceso son los *coeficientes cepstrales en la frecuencia de Mel* o MFCC (por sus siglas en inglés, *Mel Frequency Cepstral Coefficients*). De este modo las señales acústicas (a en la ecuación 3.2) quedan reflejadas como secuencias de vectores en $a \in \mathbb{R}^n$. Típicamente se suelen calcular 13 o 11 coeficientes cepstrales y además se le aplica la derivada de primer y segundo orden para modelar mejor la dinámica de la voz.

3.1.2. Modelo Acústico

Partiendo de la formula 3.2, el reconocedor debe de ser capaz de determinar el valor de $P(a|w)$, que como se ha dicho anteriormente, es la probabilidad de cuando el sistema devuelve w el usuario haya pronunciado a . Para obtener dicho valor, se utiliza un modelo acústico estadístico que modela la pronunciación de cada uno de los fonemas que contiene el vocabulario del lenguaje que se desea traducir. El Modelo Oculto de Markov o HMM (por sus siglas en inglés, *Hidden Markov Model*) es empleado normalmente para realizar los modelos acústicos aunque también existen otros posibles modelos basados en otras técnicas, como por ejemplo los modelos basados en *redes neuronales* descritos en [11] y [12].

Además de los HMM también se utiliza un modelo léxico en el que se indica que secuencia de fonemas corresponde a cada una de las palabra del vocabulario.

Modelos Ocultos de Markov

En el reconocimiento del habla se asume que los vectores de características obtenidos de la secuencia acústica observada corresponden con un *modelo de Markov*. Un *modelo de Markov* es un autómata finito estocástico, es decir, las transiciones tienen asociada una probabilidad. Además cada estado genera un vector de características como puede verse en el ejemplo de la figura 3.2.

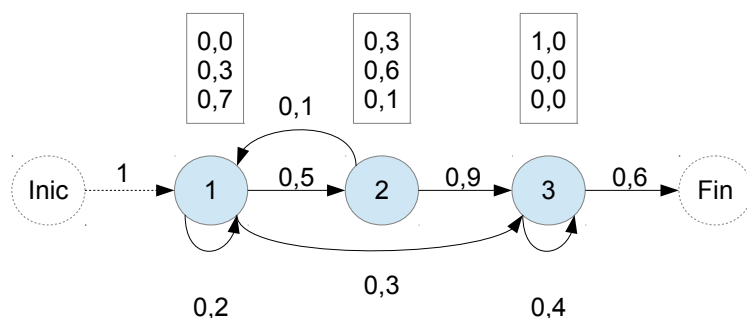


Figura 3.2: Ejemplo de un modelo oculto de markov

El objetivo del *modelos ocultos de markov* es determinar que fonemas están asociados con la secuencia acústica observada.

3.1.3. Modelo de lenguaje

La fórmula 3.2 requiere también que seamos capaces de computar para cada cadena de palabras w la probabilidad a priori $P(w)$ de que el usuario desee pronunciar w .

En un idioma cualquiera, encontrar todas las posibles combinaciones de palabras de su vocabulario resulta una tarea enorme, por no decir de lo costoso que resultaría además buscar su probabilidad a priori. Por ello, para modelar el lenguaje se utilizan gramáticas de *n-gramas* con un número n no muy grande (generalmente no más de tres) con la probabilidad asociada de que este sea utilizado en el lenguaje.

Estos diccionarios suelen obtenerse utilizando textos con temática similar a la del audio que se desea reconocer para que, en sistemas específicos en los que no se va a tener un lenguaje muy general, la respuesta sea más rápida y precisa.

Capítulo 4

Aplicaciones utilizadas

El desarrollo de aplicaciones web ha tomado un gran protagonismo debido a que éstas están siempre disponibles sin la necesidad de instalar ningún programa adicional y con la posibilidad de tener toda la información en un único sitio. El sistema de ayuda a la traducción sobre el que se basa este proyecto: CasMaCat, está desarrollado bajo una plataforma web, lo que permite fácilmente ser usado desde cualquier terminal que tenga un navegador. Esta ventaja trae consigo también una serie de inconvenientes, como por ejemplo el acceso al micrófono para poder capturar la voz del usuario.

En este apartado vamos a ver los protocolos y aplicaciones que se han utilizado para el desarrollo del trabajo. Dado que es una aplicación web, el trabajo consta de dos partes principales: la parte del cliente y la parte del servidor; y cada una de ellas se ha programado utilizando diferentes aplicaciones y lenguajes como vamos a ver a continuación.

Para la elaboración del cliente web, además de usar HTML5[13] y CSS3[14] como para el maquetado y el diseño, se han utilizado otra serie de herramientas para facilitar las comunicaciones y la captura del audio del navegador.

Para transferir datos desde el cliente al servidor mediante una comunicación directa y transferir los paquetes de forma ordenada, se ha hecho uso de los WebSocket. Este estándar permite crear una conexión punto a punto de manera similar a los sockets utilizados en otros lenguajes con la ventaja de que ciertos navegadores ya empiezan a ser compatibles con este estándar y podemos utilizarlos desde la web, además, los WebSocket permiten atravesar cortafuegos y restricciones de seguridad impuestas por los administradores de sistemas de forma transparente. A la hora de capturar el audio se ha utilizado el API de MediaStream, el cual permite acceder al audio del micrófono sin la necesidad de utilizar un *plugin* o una herramienta externa al navegador.

Por otro lado, la parte del servidor ha sido implementada principalmente en Python haciendo uso de otras herramientas externas, como por

ejemplo, la librería de comunicación utilizada para los WebSocket llamada SimpleWebSocketServer[15], además de otras que se detallan en el capítulo 5.

4.1. WebSocket

WebSocket es un API diseñado para realizar una comunicación sobre un único socket TCP de manera fiable y ordenada. Ha sido creado para ser utilizado principalmente en navegadores y servidores web, aunque puede ser usado en cualquier otro tipo de aplicaciones que requiera de una comunicación.

4.1.1. Negociación del protocolo

Para poder establecer la comunicación, el cliente primero debe de realizar una petición de conexión al servidor. Con este primer mensaje se le envía al servidor una clave con la que el servidor generará la respuesta. A continuación se muestra un ejemplo con parte de la cabecera de un mensaje real enviado.

```
GET / HTTP/1.1
Cache-Control:no-cache
Connection:Upgrade
Host:glass:8000
Origin:https://glass:4443
Pragma:no-cache
Sec-WebSocket-Key:ph3MyriKEzJ6eMSVgP3izQ==
Sec-WebSocket-Version:13
Upgrade:websocket
```

Código 4.1: Cabecera de la solicitud de comunicación de WebSocket

Cuando el servidor recibe la petición, este genera una clave de aceptación mediante la concatenación de la clave enviada por el cliente y una cadena preestablecida. El resultado de esta cadena es cifrada mediante un algoritmo SHA-1. La cadena generada se puede ver en la respuesta del servidor a la petición anterior.

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept: anB8DdIpQDoLCQ1TgV4U8uNzm9Y=
Upgrade: WebSocket
```

Código 4.2: Cabecera de la aceptación de comunicación de WebSocket

En la mayoría de los navegadores el API de WebSocket ya viene implementado, por lo que simplemente hay que sobrescribir las funciones que vienen dadas. Para poder utilizar WebSocket en el servidor se puede implementar el protocolo, aunque lo más cómodo es utilizar librerías mantenidas por la comunidad *Open Source* escritas en diversos lenguajes que facilitan un API para su uso.

4.1.2. WebSocket en el navegador

Crear una conexión mediante WebSocket en un navegador es una tarea bastante sencilla. Cabe destacar que en WebSocket la dirección de destino se debe de indicar con un esquema de URL diferente al comúnmente utilizado *http*, indicando **ws:** para conexión normales o **wss:** para el caso de una conexión segura bajo el protocolo SSL. La clase WebSocket está compuesta principalmente por cuatro eventos:

- **onopen:** este método se activa cuando la conexión ha sido creada correctamente. Se puede utilizar para saber cuando podemos empezar a enviar mensajes.
- **onclose:** cuando la conexión con el servidor se pierde o es cerrada por este se realiza una llamada a este método.
- **onmessage:** los mensajes que han sido enviados por el servidor pueden recogerse mediante este método. Los datos recibidos estarán disponibles en la variable **data** del parámetro de la función.
- **onerror:** si la conexión no puede establecerse podremos recoger el error gracias a este método.

A continuación se muestra un ejemplo básico del uso del API de WebSocket en el navegador:

```
1  try {
2    var host = "ws://" + window.location.hostname;
3    var ws = new WebSocket(host);
4    ws.binaryType = "arraybuffer"
5    ws.onopen = function (e) {
6      console.log("Socket opened.");
7      connection.send("Message to server");
8    };
9    ws.onclose = function (e) {
10     console.log("Socket closed.");
11   };
```

```

12 ws.onmessage = function (e) {
13     console.log("Socket data:", e.data);
14 };
15 ws.onerror = function (e) {
16     console.log("Socket error:", e);
17 };
18 } catch (ex) {
19     console.log("Socket exception:", ex);
20 }

```

Código 4.3: Ejemplo básico de WebSocket en el navegador

Una vez se ha activado el evento `onopen` se puede empezar a enviar mensajes al servidor. La especificación de WebSocket permite enviar cadenas de texto simple y objetos binarios de tipo *Blob* o *ArrayBuffer*. Para enviar un mensaje tan solo hay que hacer uso del método `send`.

4.2. MediaStream API

MediaStream API o *getUserMedia* (conocido así por el nombre de la función que lo inicializa) permite sincronizar el vídeo y el audio del dispositivo que lo ejecuta mediante *streams*. Esto permite enviar un flujo de datos obtenido en tiempo real directamente desde el micrófono gracias al objeto `AudioNode`. Un `AudioNode` es una interfaz genérica que representa la fuente y/o el destino del audio y un módulo de procesamiento del sonido.

Para poder capturar el audio del micrófono el API de MediaStream se ayuda de Web Audio API. El objetivo del Web Audio API es permitir utilizar el micrófono del ordenador desde el navegador sin tener que recurrir a *Flash* o *plugins* adicionales. Web Audio API dispone de un objeto principal encargado de gestionar el sonido llamado `AudioContext`, mediante el cual se obtiene el `AudioNode`.

MediaStream API requiere de permisos especiales para poder acceder al micrófono. Estos permisos se piden cada vez que el usuario accede a la página, a excepción de las páginas con conexión segura `https` en las que sólo se piden la primera vez.

```

1 var proc;
2
3 navigator.getUserMedia({ audio: true }, function(stream) {
4     window.AudioContext = window.AudioContext || window.
5         webkitAudioContext;
6     var context = new AudioContext();
7     var mediaStreamSource = context.createMediaStreamSource(stream
8         );
9 }

```

```
7   proc = context.createScriptProcessor(4096, 2, 2);
8
9   mediaStreamSource.connect(proc);
10  proc.connect(context.destination);
11
12  mediaStreamSource.connect(audioContext.destination);
13  }, function(e) {
14    alert("No se pudo obtener el audio");
15  });
```

Código 4.4: Ejemplo inicialización de MediaStream API para habilitar el micrófono en el navegador

En el código 4.4 se muestra la inicialización de *getUserMedia* para obtener el audio desde el navegador. A partir del *AudioNode* se puede obtener el audio y darle el uso que se desee. Para cumplir los objetivos de esta memoria se ha utilizado el *AudioNode* para enviar el audio del navegador, como se puede apreciar en el código 4.5.

```
1   proc.onaudioprocess = function(event) {
2     var audio = event.inputBuffer.getChannelData(0);
3     var buffer = new ArrayBuffer(audio.length * 2);
4     var view = new DataView(buffer);
5     floatTo16BitPCM(view, 0, audio);
6     ws.send(view);
7   }
```

Código 4.5: Ejemplo de captura de audio desde el navegador

Donde la función *floatTo16BitPCM* se encarga de codificar el audio obtenido a una codificación PCM de 16 bits y la variable *ws* es un objeto *WebSocket* que se supone inicializado correctamente.

4.3. iATROS

La herramienta iATROS [17] es un sistema de reconocimiento de habla e imágenes manuscritas desarrollado por el centro de investigación *Pattern Recognition and Human Language Technology* (PRHLT) de la Universitat Politècnica de València. El sistema ha sido implementado en su totalidad en el lenguaje C. iATROS es la mejora de un reconocedor de voz anterior (ATROS) que ha sido adaptado para poder reconocer tanto el habla como texto manuscrito.

Este sistema está formado por tres módulos principales, de los cuales, dos son los encargados de realizar la extracción de características de las señales

acústicas y de las imágenes. Por otra parte se encuentra el modulo de reconocimiento, el cual recoge los vectores de características devueltos por los otros dos módulos y retorna la secuencia de palabras reconocida.

Procesamiento del habla

El sistema de sonido de iATROS está basado en los módulos de sonido ALSA y permite leer tres formatos diferentes de sonido: raw, ficheros AD (definidos por el grupo PRHLT) y ficheros de audio wav sin compresión.

La extracción de característica de iATROS utiliza los *coeficientes cepstrales en la frecuencia de Mel* que en el caso del audio se extraen mediante el sistema explicado en la sección 3.1.1.

Reconocimiento de los coeficientes cepstrales

Como cualquier otro reconocedor de habla, iATROS necesita de un modelo acústico y un modelo de lenguaje para realizar el reconocimiento. El modelo acústico viene dado en iATROS por los Modelos Ocultos de Markov y estos deben ser pasados explícitamente al sistema. Además el sistema permite entrenar un modelo para que devuelva un HMM. Para poder utilizar el HMM iATROS necesita también de un modelo léxico, el cual es un diccionario con todas las palabras del vocabulario que se desea reconocer y en el que se indica para cada palabra la secuencia de fonemas que corresponde con su pronunciación.

El modelo de lenguaje viene dado por la gramática del lenguaje. La gramática a su vez está compuesta por los n -gramas que forman las palabras del diccionario. Para la gramática no se usan valores de n superiores a 3.

El proceso de decodificación es llevado a cabo utilizando el algoritmo de Viterbi. En este proceso se busca la secuencia de palabras más probable en una red que incluye los modelos morfológicos, léxico y sintácticos.

4.4. CasMaCat



Figura 4.1: Logotipo del proyecto CasMaCat

El proyecto CasMaCat ha sido desarrollado por el PRHLT conjuntamente con el grupo *Statistical Machine Translation* (Universidad de Edimburgo) y el *Center for Research and Innovation in Translation and Translation Technology* (Escuela de Negocios de Copenhague). Dentro del marco de este proyecto se ha desarrollado el sistema de traducción asistida por computador de próxima generación.

La interfaz del sistema ha sido desarrollada mediante una plataforma web utilizando HTML5, CSS3 y PHP. Los motores de CAT y MT que utiliza han sido implementados en Python (al igual que el motor ASR realizado en este trabajo) pero también utilizan herramientas escritas en otros lenguajes diferentes.

4.4.1. Interfaz de CasMaCat

La interfaz está formada por una serie de vistas diseñadas para cada una de las diferentes tareas. La vista principal es la interfaz de traducción mediante la cual el usuario puede traducir documentos y realizar la post-edición de los mismos.

A su vez también existe una interfaz que permite al usuario subir los documentos que desea traducir en formato XLIFF. Desde esta vista se puede también crear un proyecto el cual contendrá un conjunto de documentos. Por otro lado hay otra interfaz encargada de gestionar los ficheros subidos o que ya están en el sistema.

Interfaz de post-edición

Desde esta interfaz el usuario del sistema puede realizar las traducciones. El texto subido al sistema es segmentado y traducido por partes. La traducción completa de un documento se realiza traduciendo cada uno de los segmentos por separado. Cada vez que se elige un segmento, este es resaltado y traducido mediante la MT como se muestra en la figura 4.2, además, se muestra igualmente el texto original para que el usuario pueda comprobar la traducción.

La interfaz de post-edición ofrece también atajos de teclado para poder trabajar más rápido y tiene diferentes estados para los segmentos, así cuando un segmento ha sido traducido correctamente puede marcarse como traducido.

Otra de las características que ofrece es un sistema de registro de la actividad el cual resulta muy útil a la hora de tomar decisiones para mejorar el sistema.

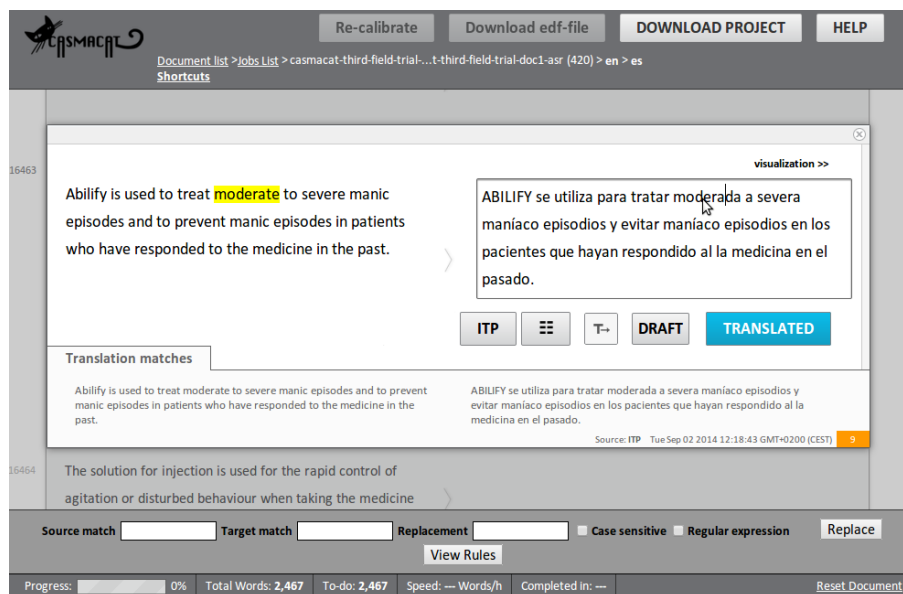


Figura 4.2: Interfaz de post-edición

Sistema de traducción interactivo-predictivo

En este apartado se muestran algunas de las principales características de las herramientas de ITP de las que dispone CasMaCat:

- **Autocompletado inteligente:** el sistema produce la predicción del resto de la sentencia por de acuerdo al texto que el usuario ya ha introducido.
- **Medidas de confianza:** se utilizan medidas de confianza para informar al usuario sobre posibles errores en la traducción, advirtiéndolo con diferentes colores según la gravedad del error.
- **Alineamiento de palabras:** Cuando se pasa el ratón sobre alguna de las palabras, se resaltan mediante colores las correspondencias entre esa palabra en el otro idioma. Se puede ver un ejemplo en la figura 4.2 donde la palabra *moderate* se encuentra resaltada debido a que el usuario ha puesto el ratón sobre la palabra moderada.
- **Rechazo de la predicción:** El usuario puede pedir al sistema que realice una nueva traducción completa de una frase en el caso de que desee rechazar la traducción actual. Esto invalida la traducción actual para la frase que ha sido traducida y devuelve al usuario una nueva alternativa.

4.4.2. Componentes

CasMaCat ha sido implementado de forma modular, esto permite que cada módulo tenga un API específico para cada tarea con las ventajas que esto trae. Existen tres módulos principales: La interfaz gráfica, el servidor CAT y el servidor MT.

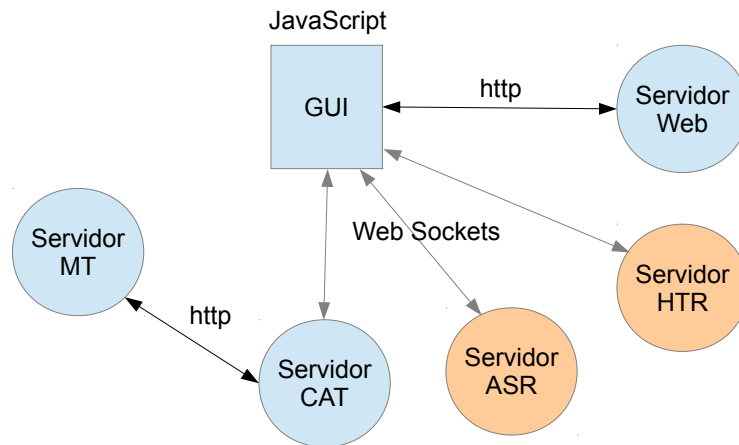


Figura 4.3: Diseño de los diferentes componentes de CasMaCat

Además de estos módulos CasMaCat trae también el módulo de reconocimiento de gestos y escritura, el módulo de seguimiento de ojos y el módulo ASR implementado para este trabajo.

Capítulo 5

Implementación del trabajo

En este apartado de la memoria se va a describir el trabajo realizado y el proceso de desarrollo que se ha seguido para la implementación del TFG. El lector podrá comprender como ha sido el transcurso del trabajo, y que problemas y soluciones se han ido encontrando para poder abordar los objetivos iniciales. El orden en el que vienen explicadas cada una de las partes corresponde con el ciclo de funcionamiento del sistema.

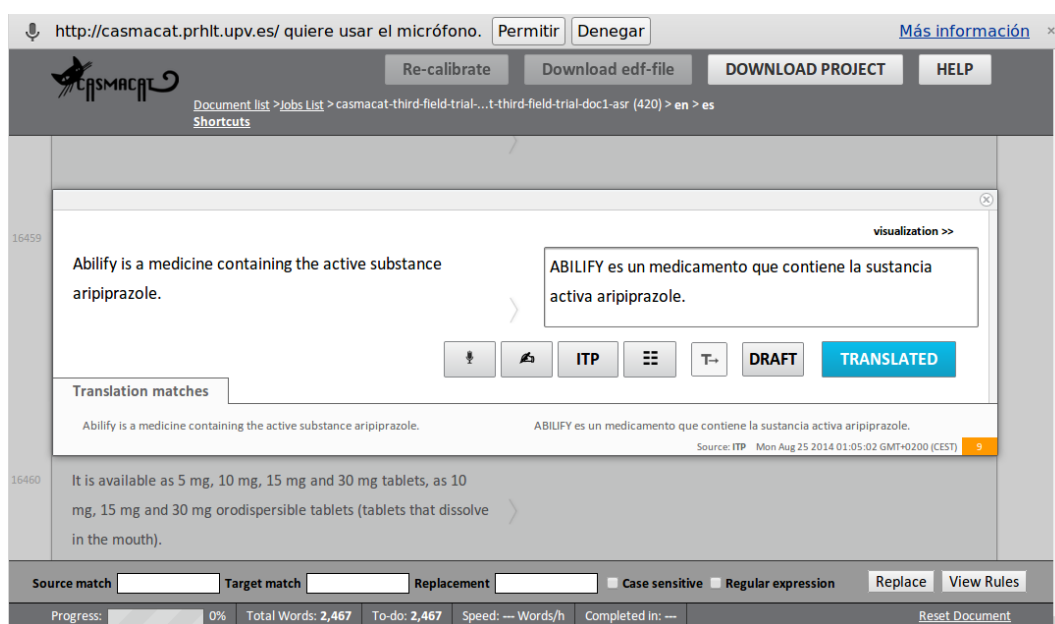


Figura 5.1: Interfaz de CasMaCat con el modulo de reconocimiento de voz

El proceso de corrección mediante la voz comienza seleccionando con el ratón las palabras que se desean sustituir o situando el puntero en el espacio donde se añadirán las palabras dictadas. Una vez se ha elegido, al pulsar

el botón del reconocedor de voz, el sistema empezará a enviar el audio del micrófono al servidor ASR. Cuando se desee terminar la grabación se pulsará de nuevo el botón y el servidor comenzará el proceso de transcripción. Para transcribir el audio, el servidor primero guarda el buffer de datos como un fichero de audio wav, después extrae los coeficientes cepstrales y realiza el reconocimiento del audio. Una vez el servidor ASR ha recuperado el texto, este lo envía de nuevo al navegador para que lo tokenice y lo añada en la interfaz.

La explicación de las distintas partes se va a realizar siguiendo el flujo de ejecución normal de la aplicación, es decir, desde que el usuario que interactúa con el sistema decide corregir una traducción con la voz y realiza una grabación, está se envía al servidor, se transcribe el audio, se envía al navegador y se muestra en la pantalla en resultado final. El flujo de trabajo junto con las tecnologías usadas se puede ver en la figura 5.2.

En cuanto a la interfaz, al integrar el sistema de reconocimiento del habla en *CasMaCat* se ha procurado mantener el *look and feel* original del sistema. Siguiendo la misma línea que el reconocedor de escritura que ya lleva integrado *CasMaCat*, se ha añadido el nuevo botón de captura del audio al lado del resto de botones.

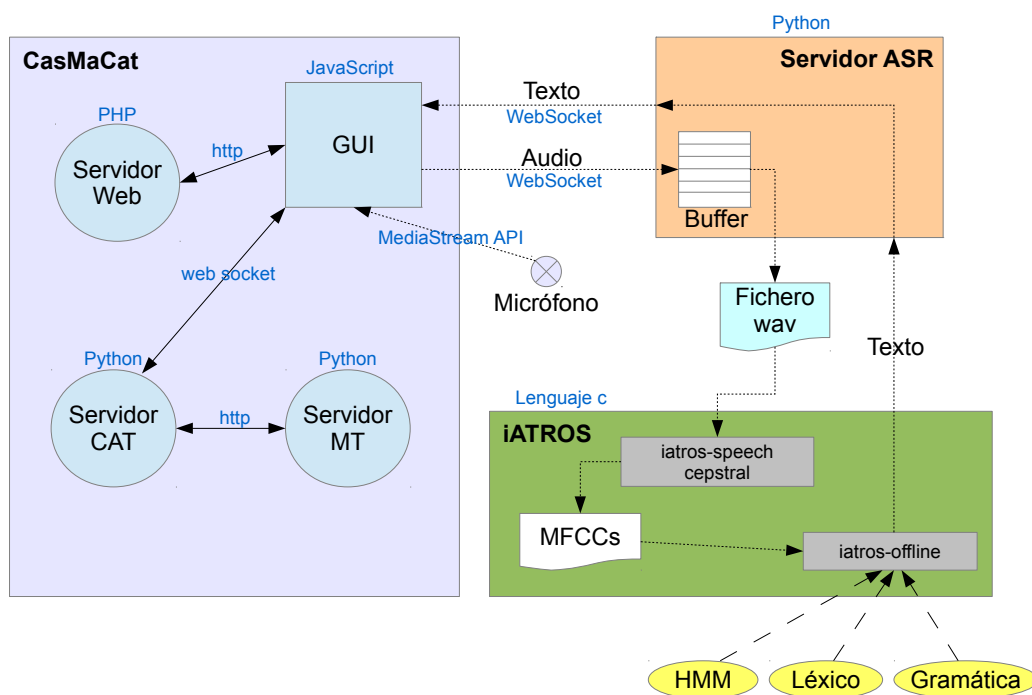


Figura 5.2: Esquema de tecnologías usadas y flujo de la aplicación

5.1. Captura del audio

La captura del audio se realiza en stream, es decir, se envían los datos del audio al servidor en cuanto se obtienen. El navegador va enviando el audio a la vez que lo recibe del micrófono y es el servidor el encargado de juntar todos los paquetes y crear el archivo de sonido que más tarde se transcribirá.

5.1.1. Captura del audio del navegador

La grabación de audio en el navegador ha sido realizada en *JavaScript* utilizando el *MediaStream API*. A través de un *AudioNode* se recupera el stream del sonido de uno de los canales del micrófono. Antes de ser enviados los datos al servidor se transforma el formato de audio de float a una codificación PCM de 16 bits.

5.1.2. Envío del stream de audio

En este apartado es donde entran en acción los *WebSocket*. Gracias a ellos se consigue enviar la información de forma ordenada y con la seguridad de que no vamos a perder ningún paquete ya que está implementado sobre TCP/IP. El API de *WebSocket* facilita mucho el trabajo en la comunicación dado que permite atravesar cortafuegos de forma transparente, y como se puede ver en la sección 4.1 es muy fácil de implementar en el navegador.

Para avisar al servidor de que se le va a empezar a enviar el audio, primero se envía la señal *start* que establece el contexto del reconocimiento, y a continuación se envían los datos del audio conforme el micrófono los captura. Para avisar al servidor de que el usuario ha terminado la grabación y que debe de guardar el audio se utiliza la señal *end*.

5.1.3. Guardado del audio en el servidor

Como se ha comentado anteriormente, el servidor ha sido implementado para utilizar una serie de señales y saber en que estado se encuentra. Cuando el servidor recibe la señal de *start* sabe que los siguientes paquetes que van ha llegar son de audio, y estos los almacena en un buffer. Una vez recibe la señal *end* vuelca el contenido del buffer en un fichero de audio en formato wav.

Antes de poder usar el archivo de audio hay que reducir la frecuencia de muestreo de 44100Hz a 16kHz dado que el modelo acústico utilizado en iATROS ha sido entrenado en esa frecuencia.

5.2. Reconocimiento del audio y retorno del resultado

Ahora veremos como se utiliza iATROS para reconocer el audio y como el texto reconocido es enviado de nuevo a la interfaz. Para empezar el primer paso es generar los coeficientes cepstrales a partir del audio.

5.2.1. Generación de Coeficientes Cepstrales

Una vez se ha guardado el audio en el formato correcto se obtienen los Coeficientes Cepstrales. Para ellos se utiliza el módulo de iATROS encargado de obtener el vector de características del audio.

```
iatros-speech-cepstral -c cepstral.conf -i audio.wav -t wav
```

Código 5.1: Comando para obtener los Coeficientes Cepstrales

Se debe de indicar a demás del audio, el fichero de configuración del módulo de iATROS. En este fichero viene información sobre el formato de la muestra de audio o el umbral de silencio a partir del cual se ignorará el sonido.

El código 5.2 muestra el contenido del fichero *cepstral.conf* donde pueden apreciarse los parámetros necesarios para las distintas fases del preproceado. Entre estos parámetros se encuentran la frecuencia de muestreo del audio, la longitud de la FFT, el tamaño de la ventana de Hamming, entre otros. Hay más información acerca de la fase de preproceso en la sección 3.1.1.

```
1 # Configuration file for iATROS system
2 # Generated by iATROS Configurator
3 # Thu Feb 21 12:32:11 2008
4
5 <filter>
6 TriangularFilters 23
7 </filter>
8
9 <parameters>
10 FrameSize 410
11 SampleFreq 16000
12 SubSampleFreq 100
13 FFTlength 512
14 FrameShift 160
15 WindowLen 0.025625
16 Factor 0.97
17 WindowAlfa 0.54
```

Locutores	164
Palabras pronunciadas	42.000
Longitud (en horas)	4

Cuadro 5.1: Información del corpus *Albayzin*

```

18 CepCoefNumb 13
19 Channels 1
20 Bits 16
21 Frames 32
22 SecondsSilence 0.5
23 SilenceThreshold 0
24 Derivative 2
25 </parameters>
26
27 <buffers>
28 SizeSignal 100000
29 SizeCC 1000
30 </buffers>
31
32 <device>
33 InputDevice default
34 OutputDevice default
35 </device>

```

Código 5.2: Contenido del fichero *cepstral.conf*

5.2.2. Reconocimiento del audio

Para reconocer el audio son necesarios el modelo acústico y el modelo de lenguaje. El modelo acústico que se ha utilizado se encuentra en el fichero *albayzin_iatros_64gs.hmm*. Este modelo acústico ha sido obtenido entrenando el corpus geográfico *Albayzin*, que consta de consultas orales realizadas a una base de datos geográfica y contiene oraciones habladas en español fonéticamente equilibradas. En la tabla 5.1 se puede encontrar más información acerca del corpus.

Junto con el modelo acústico viene el diccionario léxico, el cual consiste en una lista con el vocabulario que el sistema será capaz de reconocer, y en el que cada palabra viene junto con su pronunciación para que el sistema sepa como “suena” cada palabra. Para poder expresar la pronunciación se han seguido las pautas descritas en el libro de *Fonética acústica de la lengua española* [18].

Por otra parte el modelo de lenguaje en iATROS viene dado por la gramática del lenguaje. La gramática es representada mediante n-gramas uti-

Fonema	Pronunciación
rr	@
x	ks
y	i
qu	k
v	b

Cuadro 5.2: Ejemplo de pronunciaciones de algunos fonemas

lizando cada una de las palabras que aparece en el diccionario léxico. Además también se indica la probabilidad de aparición, ya sea de la palabra o de del n-grama

```
iatros-offline -c config/offline.conf
```

Código 5.3: Comando para reconocer los Coeficientes Cepstrales

Indicando a iATROS el modelo acústico, el modelo léxico y la gramática, junto con los coeficientes cepstrales obtenidos en el preproceso, se obtiene el texto reconocido. Como se ve en el comando del código 5.3, todos estos modelos se indican en el fichero de configuración.

5.2.3. Envío del texto desde Python al navegador

El envío de la transcripción del audio se realiza desde Python de manera muy sencilla gracias a la librería de WebSockets utilizada: *SimpleWebSocketServer*. Tan solo haciendo uso del método *send* se consigue enviar al navegador la cadena de texto del audio reconocido.

5.3. Integración con CasMaCat

Para integrar el reconocedor con CasMaCat se ha tenido que modificar la interfaz de éste y añadir nuevos ficheros JavaScript, en los que aparte del código para recuperar el audio, se encuentra el código utilizado para insertar el texto.

5.3.1. Comprobación de los tokens seleccionados

Una vez que el navegador ha recibido el texto, se comprueba los tokens que fueron seleccionados o la posición del ratón para sustituir o añadir el texto respectivamente. Para comprobar que tokens han sido seleccionados

JavaScript tiene un método que permite saber la posición del primer y el último elemento seleccionado, de esta manera se pueden recorrer todos los elementos intermedios y obtener así el texto completo que el usuario ha elegido.

5.3.2. Inserción de la transcripción en el navegador

En el caso de que se deseen sustituir los tokens seleccionados, el sistema borra todos los tokens a excepción del último, el cual sustituye su contenido con el texto de la transcripción recibido. Para el caso de la inserción simplemente se agrega el nuevo texto detrás de la posición del cursor.

CasMaCat ofrece una serie de métodos para realizar la comprobación de los tokens, cada vez que se modifica el texto es necesario realizar una llamada a uno de los métodos para que compruebe el estado de los tokens y en el caso de que alguno contenga más de una palabra, el sistema divide ese token en tantos como palabras contenga.

Integración con ITP

Cada vez que se insertan los nuevos tokens, el sufijo es actualizado teniendo en cuenta la frase reconocida. De este modo ofrece una traducción mejor y no solo “pega” el texto tal cual es recibido del servidor ASR.



5.4. Explotando la información contextual para mejorar el reconocimiento

El sistema descrito anteriormente no es un sistema perfecto, comete errores. Además para un vocabulario de reconocimiento demasiado grande puede resultar lento en devolver el resultado. Para intentar minimizar los errores y mejorar la velocidad, se han implementado dos mejoras que se detallan a continuación.

La primera de ellas consiste en enviar además del audio el prefijo p que ya ha sido corregido y la frase original x , de esta manera iATROS puede buscar una mejor solución. Adaptando la ecuación 3.1 para que tenga en cuenta los valores citados, la nueva ecuación queda de la forma:

$$\hat{w} = \arg \max_w P(w|a, p, x) \quad (5.1)$$

Donde $P(w|a, p, x)$ es la probabilidad de que se obtenga w cuando: se haya pronunciado a , p precede a w y se esté traduciendo la frase x . Ampliando la

	x	Click OK to close the print dialog
1	\hat{s}	Haga clic para cerrar el diálogo de impresión
	p	Haga clic
	a	
	\hat{w}	en ACEPTAR
2	\hat{s}	en ACEPTAR para cerrar el diálogo de impresión
	p	Haga clic en ACEPTAR para cerrar el
	a	
	\hat{w}	cuadro de
3	\hat{s}	cuadro de diálogo de impresión
	p	Haga clic en ACEPTAR para cerrar el cuadro de diálogo de impresión

Cuadro 5.3: Ejemplo de interacción con el sistema haciendo uso del prefijo.
El símbolo | representa el cursor situado por el usuario

fórmula anterior se obtiene la ecuación 5.2 que puede ser resuelta mediante interpolación lineal.

$$\hat{w} = \arg \max_w P(w|p, x) \cdot P(a|w) \quad (5.2)$$

Como se puede apreciar el modelo acústico es el mismo que se ha utilizado hasta ahora y la diferencia están en el modelo de lenguaje que ahora incluye también el prefijo p y la frase original x .

La segunda mejora consiste en aprovechar la información que ha generado CasMaCat para traducir la frase original y que puede resultar muy útil para el reconocimiento del habla. Utilizando el grafo de palabras de traducción que genera CasMaCat para traducir una frase se puede obtener un modelo de lenguaje compatible con iATROS. Dado que este modelo ha sido obtenido a partir de las posibles traducciones de la frase original, el vocabulario está muy limitado, por ello además de este modelo se utilizan dos más: un diccionario estadístico del cual se extraen las posibles traducciones de cada una de las palabras del texto origen por separado, y un modelo de lenguaje general sacado de texto que tienen la misma temática que el texto a traducir.

Los pesos asociados a cada uno de los modelos durante la interpolación de los mismos, será la clave para que el reconocedor de voz sea más preciso. En el siguiente capítulo se realiza una evaluación de la interpolación de los distintos modelos utilizando o no el prefijo corregido.

Capítulo 6

Evaluación del sistema

Para poder evaluar las mejoras que se han incorporado, se ha utilizado como medida la perplejidad de los modelos. La perplejidad de un modelo de lenguaje con respecto a una palabra w_1 representa el número de palabras que podrían seguir a w_1 . Cuanto menor sea la perplejidad indica que un modelo de lenguaje será más predecible y por tanto, el modelo de lenguaje será mejor. Para realizar el cálculo de la perplejidad se ha utilizado la herramienta *ngram*, un programa incluido dentro del *SRI Language Modeling Toolkit (SRILM)*[20].

El inconveniente de utilizar la perplejidad como medida de calidad de los modelos de lenguajes, es que en nuestro caso, los modelos a comparar no contienen las mismas palabras. Al realizar el cálculo de la perplejidad con un mismo fichero de *test* en modelos con palabras distintas, las palabras fuera de vocabulario (que no aparecen en el texto de *test*) son ignoradas y no se tienen en cuenta, por lo que no perjudica para el cálculo y beneficia injustamente a modelos que contengan pocas palabras con respecto a otros que son más completos.

Para solucionar este problema, se ha obtenido el vocabulario total de los tres modelos: el modelo general, el modelo del grafo de traducción y el modelo obtenido del diccionario estadístico. Este vocabulario se ha utilizado para añadir a cada modelo las palabras fuera de vocabulario con una probabilidad baja, de este modo se tienen en cuenta dichas palabras.¹

El *corpus* utilizado para evaluar el sistema ha sido una extracción de 5337 frases del *corpus* xenox, el cual ha sido extraído a partir de una colección de manuales técnicos.

¹SRI International. SRILM-FAQ. <http://www.speech.sri.com/projects/srilm/manpages/srilm-faq.7.html>, Consultado el 17/09/2014

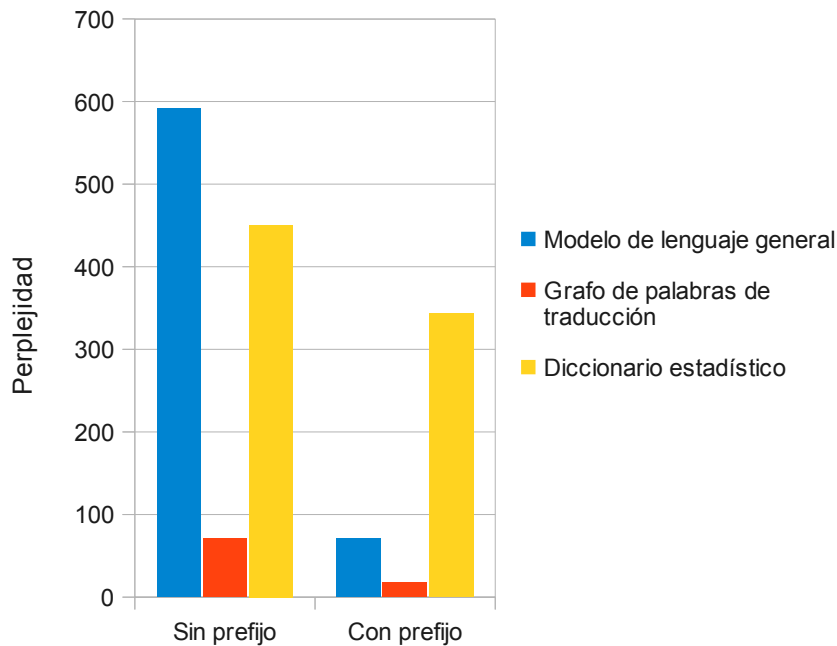


Figura 6.1: Gráfica de la perplejidad de los tres modelos

En la figura 6.1 se puede apreciar como el mejor modelo es el obtenido a partir del grafo de traducción dado que es el que más se ajusta a cada una de las frases de *test*. También se ve claramente como el uso del prefijo mejora considerablemente la transcripción, aunque en el caso que más se nota es el modelo general dado que este modelo ha sido obtenido utilizando todas las frases del *corpus*.

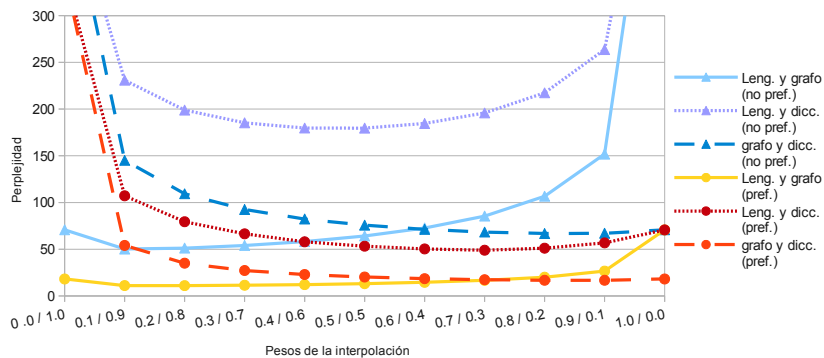


Figura 6.2: Gráfica de la perplejidad de la interpolación de los modelos

Observando el gráfico de la figura 6.2 se puede ver que haciendo uso de

la interpolación de varios modelos se ha conseguido mejores resultados que utilizando un único modelo. En este caso el mejor resultado se ha obtenido haciendo la interpolación del modelo general junto con el modelo del grafo de traducción, aplicando unos pesos de 0.1 y 0.9 respectivamente.

Capítulo 7

Conclusiones

La implementación de este trabajo fin de grado ha resultado una gran experiencia. No solo por el hecho de poder probar la teoría que vemos en clase en un entorno real, si no también por haber podido realizar el TFG en el PRHLT, donde he podido comprobar como es trabajar en un centro de investigación y acercarme más a este mundo.

Durante el desarrollo he podido mejorar mis conocimientos sobre JavaScript y sokets además de reforzar los conocimientos sobre reconocimiento del habla y traducción automática.

En cuanto a los objetivos, se ha conseguido capturar y guardar en el servidor el audio correctamente, pese a la dificultad de que la tecnología utilizada es bastante reciente y no se encontraba mucha información al respecto. También se ha logrado crear un servidor capaz de comunicarse con iATROS y realizar el reconocimiento del audio. Se ha sido capaz de modificar la interfaz de CasMaCat para integrar el sistema de captura y reconocimiento del audio, aunque personalmente el manejo de la interfaz para el reconocimiento de voz resulta un tanto complicada. El último objetivo también se ha cumplido puesto que se ha conseguido enviar el prefijo validado para cada frase.

Se ha demostrado que las mejoras introducidas mejoran el reconocimiento de la voz, aunque con un estudio más exhaustivo se podría comprobar las combinaciones de interpolar los tres modelos para ver si también hay mejora. También se ha intentado realizar una evaluación del sistema utilizando grabaciones reales y recuperando la transcripción utilizando cada uno de los modelos, pero por falta de tiempo no se ha podido incluir en esta memoria.

Por último se podría mejorar en el trabajo la forma en la que el usuario interactúa con el sistema dado que no resulta muy cómodo. Además, hay un cuello de botella en el sistema dado que el audio se envía por stream pero no es utilizado hasta que se almacena en un fichero.

Índice de códigos

4.1.	Cabecera de la solicitud de comunicación de WebSocket	18
4.2.	Cabecera de la aceptación de comunicación de WebSocket . . .	18
4.3.	Ejemplo básico de WebSocket en el navegador	19
4.4.	Ejemplo inicialización de MediaStream API para habilitar el micrófono en el navegador	20
4.5.	Ejemplo de captura de audio desde el navegador	21
5.1.	Comando para obtener los Coeficientes Cepstrales	29
5.2.	Contenido del fichero <i>cepstral.conf</i>	29
5.3.	Comando para reconocer los Coeficientes Cepstrales	31

Índice de figuras

2.1. Tableta digitalizadora	11
3.1. Esquema del preproceso del audio	14
3.2. Ejemplo de un modelo oculto de markov	16
4.1. Logotipo del proyecto CasMaCat	22
4.2. Interfaz de post-edición de CasMaCat	24
4.3. Diseño de los diferentes componentes de CasMaCat	25
5.1. Interfaz de CasMaCat con el modulo de reconocimiento de voz	26
5.2. Esquema de tecnologías usadas y flujo de la aplicación	27
6.1. Gráfica de la perplejidad de los tres modelos	35
6.2. Gráfica de la perplejidad de la interpolación de los modelos	35

Índice de cuadros

2.1. Ejemplo de traducción de un sistema CAT	11
5.1. Información del corpus <i>Albayzin</i>	30
5.2. Ejemplo de pronunciaciones de algunos fonemas	31
5.3. Ejemplo de interacción con el sistema haciendo uso del prefijo. El simbolo representa el cursor situado por el usuario	33

Bibliografía

- [1] Wikipedia. Words per minute. http://en.wikipedia.org/wiki/Words_per_minute, Consultado el 02/09/2014.
- [2] Jim Wardell. Speech recognition for translators: microphone tips. <http://www.knowbrainer.com/index.cfm/blog/speech-recognition-for-translators-microphone-tips/>, Consultado el 02/09/2014.
- [3] Wikipedia. Traducción asistida por computadora. http://es.wikipedia.org/wiki/Traducción_asistida_por_computadora, Consultado el 11/08/2014.
- [4] Wikipedia. Memoria de traducción. http://es.wikipedia.org/wiki/Memoria_de_traducción, Consultado el 14/07/2014.
- [5] Peter F. Brown, Stephen A. Della Pietra, Vicent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational Linguistics - Special issue on using large corpora: II (Volumen 19, issue 2)*, pages 266–311, 1993.
- [6] Antonio L. Lagarda, Vicent Alabau, Carlos D. Martínez-Hinarejos, Alejandro H. Toselli, Verónica Romero, José R. Navarro-Cerdan, and Enrique Vidal. Computer-assisted handwritten text transcription using speech recognition. In *V Jornadas en Tecnología del Habla (VJTH'2008)*, pages 229–232, 2008.
- [7] F. Casacuberta, J.A. Sanchez, and E. Vidal. IV-5: Computer Assisted Translation. Transparencias de la asignatura de Traducción Automática, Abril 2011.
- [8] Marina Calvo, Laura Chicote, and Tamara Domenech. File:graphic tablet. http://commons.wikimedia.org/wiki/File%3AGraphic_tablet.svg, Consultado el 21/08/2014.

-
- [9] Frederik Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.
- [10] Tema 2.3. Representación de objetos: señales acústicas. Transparencias de la asignatura de Percepción, Febrero 2013.
- [11] Joe Tebelskis. *Speech Recognition using Neural Networks*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1995.
- [12] G.T. Tsenov and V.M. Mladenov. Speech recognition using neural networks. In *Neural Network Applications in Electrical Engineering (NEUREL), 2010 10th Symposium on*, pages 181–186, 2010.
- [13] W3C. HTML5. <http://www.w3.org/TR/html5/>, Consultado el 08/07/2014.
- [14] W3C. Cascading Style Sheets (CSS) Snapshot 2010. <http://www.w3.org/TR/CSS/>, Consultado el 08/07/2014.
- [15] Simple WebSocket Server. <http://opiate.github.io/SimpleWebSocketServer/>, Consultado el 09/07/2014.
- [16] Boris Smus. Getting Started with Web Audio API. <http://www.html5rocks.com/en/tutorials/webaudio/intro/>, Consultado el 23/08/2014.
- [17] Míriam Luján-Mares, Vicent Tamarit, Vicent Alabau, Carlos D. Martínez-Hinarejos, Moisés Pastor i Gadea, Alberto Sanchis, and Alejandro H. Toselli. iATROS: A speech and handwriting recognition system. In *V Jornadas en Tecnologies del Habla (VJTH'2008)*, pages 75–78, 2008.
- [18] Antonio Quilis. *Fonética acústica de la lengua española*. Gredos, 1987.
- [19] SRI Internacional. SRILM-FAQ. <http://www.speech.sri.com/projects/srilm/manpages/srilm-faq.7.html>, Consultado el 19/09/2014.
- [20] SRI Internacional. SRILM - The SRI Language Modeling Toolkit. <http://www.speech.sri.com/projects/srilm/>, Consultado el 19/09/2014.