UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño de mecanismos basados en traducción automática para la portabilidad entre idiomas en un sistema de comprensión del habla

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carlos Millán Soler

Tutor: Encarna Segarra Soriano
Fernando García Granada

Curso 2015-2016

Resumen

En este proyecto se hace uso de herramientas de traducción automática para la creación de un sistema de comprensión del habla en inglés y francés a partir de uno originalmente en castellano. Gracias al uso de estas herramientas se evita la necesidad de una traducción y etiquetado manual para la portabilidad a un nuevo idioma. Se ha seguido la aproximación *train-on-target* para volver a estimar un nuevo modelo de comprensión en estos idiomas. Para ello ha sido necesaria la obtención de un conjunto de entrenamiento en ambos idiomas al que se le ha aplicado un proceso de normalización y otro posterior de segmentación y etiquetado.

Palabras clave: Comprensión del habla, multilingüismo, normalización, segmentación, estrategia *train-on-target*

Resum

En aquest projecte es fa ús d'eines de traducció automàtica per a la creació d'un sistema de comprensió de la parla en anglès i francès a partir d'un obtingut originalment en castellà. Gràcies a l'ús d'aquestes eines s'evita la necessitat d'una traducció i etiquetat manual per a la portabilitat a un nou idioma. S'ha seguit l'aproximació *train-on-target* per a tornar a estimar un nou model de comprensió en aquests idiomes. Per a això ha sigut necessària l'obtenció d'un conjunt d'entrenament en ambdós idiomes al que se li ha aplicat un procés de normalització i altre posterior de segmentació i etiquetatge.

Paraules clau: Comprensió de la parla, multilingüisme, normalització, segmentació, estratègia *train-on-target*

Abstract

In this project machine translation tools are used to create a spoken language understanding system in English and French from one originally in Spanish. By using these tools the need of manual translation and labeling is avoided for its portability to a new language. The train-on-target approach has been followed to re-estimate a new understanding model in these languages. In order to accomplish this, a training set in both languages has been required, and a process of normalization and another one of segmentation and labeling have been applied to these sets.

Key words: Spoken language understanding, multilingualism, normalization, segmentation, train-on-target strategy

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1 Comprensión del habla

La comprensión del habla se enmarca dentro de lo que en la actualidad se denomina como "Ingeniería del lenguaje". Dentro de este ámbito hay un gran volumen de aplicaciones prácticas que son útiles en muchos ámbitos de la vida. Se pueden encontrar desde sistemas que pueden interpretar el idioma en el que está escrito un texto, pasando por herramientas de reconocimiento de la voz hasta aplicaciones de diálogo y etiquetado semántico del habla.

Una de las características más importantes de estos sistemas es la posibilidad de ser dotados de multilingüismo, que consiste en la capacidad de poder ser usados en otros idiomas diferentes al original. Esta característica no fue posible hasta los años noventa, en los que se realizaron grandes avances de aplicación en traducción automática. Esta ha sido la motivación principal en le realización de este proyecto.

Uno de las principales aplicaciones de los sistema de comprensión del habla es la de proveer un sistema de diálogo para un dominio limitado. Estos sistemas se basan en la interacción de los usuarios con un sistema capaz de etiquetar semánticamente la información proporcionada por el usuario mediante lenguaje natural. A partir de esta información se extraen los datos que el usuario solicita al sistema y se genera una respuesta acorde a la información requerida. Estos sistemas suelen estar formados por un reconocedor del habla que transcribe la voz del usuario, un módulo semántico que se encarga del etiquetado semántico y un módulo que genera la consulta que necesita el sistema de información para proporcionar la respuesta al usuario.

En la figura 1.1 se puede visualizar el esquema general de un sistema de comprensión diálogo hablado de consulta a un sistema de información. En él se pueden ver todos los módulos que se han comentado y dónde se encuentra el componente de comprensión del habla. Se puede observar que el analizador semántico depende de un modelo semántico, cuyo proceso de obtención se explicará a lo largo del presente proyecto.

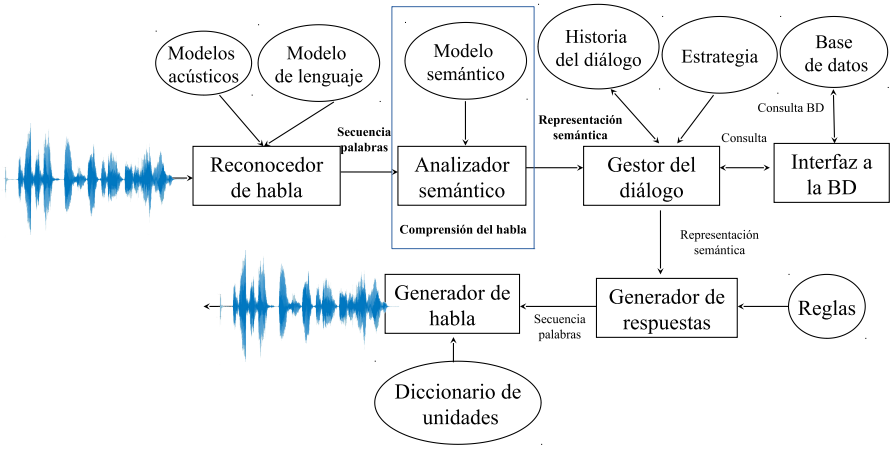

Figura 1.1: Esquema general de un sistema de diálogo de consulta a un sistema de información

El proceso de comprensión se puede resumir en seis puntos que se enuncian a continuación.

1. Reconocimiento y transcripción de la voz del usuario. Esta acción se realiza mediante un módulo de reconocimiento del habla entrenado previamente en el idioma destino que sea capaz de reconocer frases dentro del contexto de aplicación del sistema. Este módulo hace uso de unos modelos acústicos ya entrenados y de un modelo de lenguaje para la transcripción.

2. Etiquetado semántico de las palabras de entrada al sistema de comprensión del habla. Estas etiquetas semánticas se corresponden con los conceptos que entiende el sistema de consulta con el que interactúa el usuario. Tras el proceso de reconocimiento del habla, el analizador semántico se encarga de generar la representación semántica de la entrada. Además, este último módulo nombrado depende del modelo semántico, cuya obtención se explicará en los siguientes capítulos.

3. Creación de la consulta. Es la acción que se realiza una vez se ha obtenido la representación semántica de la entrada al sistema. Este módulo (gestor de diálogo) recibe información de la historia de diálogo y de una estrategia definida previamente. Se encarga de generar la salida en un formato comprensible para la interfaz de la base de datos.

4. Consulta a la base de datos. La interfaz de la base de datos realiza la consulta y extrae la información solicitada por el usuario.

5. Recuperación de la información solicitada. A partir de la información obtenida de la base de datos y de su representación semántica se genera la respuesta en forma de lenguaje natural siguiendo unas reglas predefinidas para su correcta realización.

6. Generación del habla. Si la respuesta al usuario se hace de forma hablada, se transforma el texto proporcionado por el generador de respuestas a voz. Una vez generada la salida en forma de voz, ésta se transmite al usuario.

Es un reto actualmente conseguir reducir el porcentaje de error de los sistemas de comprensión del habla. Esto es un problema más fácilmente abordable cuando se trata de un sistema que trabaja en un ámbito semántico restringido. En estos casos la tasa de error se ha conseguido reducir drásticamente durante los últimos años.

1.2 Estado del arte

1.2.1 Inicios

Como se ha comentado, en los años noventa surgieron los primeros avances relevantes en el campo de la comprensión del habla.

Una de las principales propuestas de sistema de comprensión del lenguaje natural de esta época fue realizada por el MIT (*Massachusetts Institute of Technology*) con el analizador lingüístico TINA [1]. Se trata de un conjunto probabilístico de reglas basadas en gramáticas incontextuales que en tiempo de ejecución actúa como una red donde cada nodo representa una categoría sintáctica o semántica. Las probabilidades estimadas a partir de los datos de entrenamiento permiten acotar la búsqueda durante el proceso de reconocimiento y etiquetar cada palabra de la frase con la categoría del nodo que le corresponde.

Actualmente existen otras tecnologías utilizadas en el ámbito de la "Ingeniería del lenguaje" que han mejorado mucho en los últimos años llegando a arrojar resultados muy relevantes en este campo.

1.2.2 Máquinas de vectores de soporte

Se trata de un método aparecido en 1979 pero muy usado en la actualidad. Se basa en el uso de algoritmos de aprendizaje supervisado y es altamente utilizado en problemas de clasificación y regresión.

Los SVM (*Support Vector Machines*) construyen un modelo de clasificación a partir de los datos de entrenamiento que se les proporciona. Este modelo representa la posición de las muestras en el espacio separando cada clase lo más ampliamente posible. Para ello, los SVM construyen un hiperplano o conjunto de hiperplanos separadores que maximicen la distancia entre las clases.

Un ejemplo de la correcta clasificación de dos clases en un espacio de dos dimensiones se puede ver en la figura 1.2.

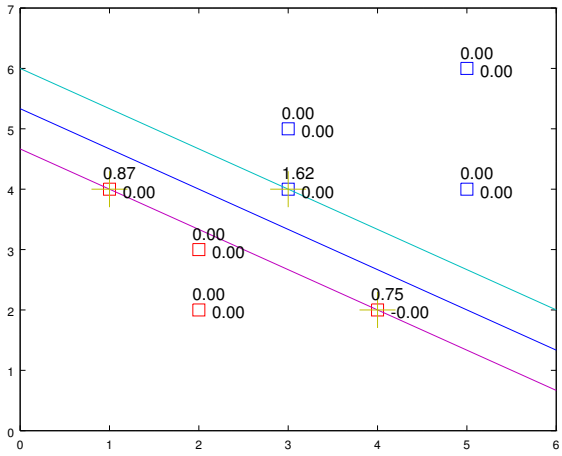

Figura 1.2: Ejemplo de clasificación usando SVM

Para el ámbito de la comprensión del habla existen trabajos [2] que hacen uso de esta técnica para la clasificación de las palabras de las frases de entrada en distintas categorías gramaticales.

En el caso citado por ejemplo, se hace uso de esta técnica para entrenar modelos a partir de tuplas de categorías y valores. La salida del algoritmo es un árbol semántico con conceptos en los nodos terminales asociados a los valores de la base de datos.

1.2.3 Redes neuronales

Es una de las técnicas que más está siendo utilizada en los últimos años. En 1943 surgieron los primeros modelos de ANN (*Artificial Neural Networks*), pero no fue hasta 1960 cuando gracias al desarrollo del ADALINE se aplicaron por primera vez de forma industrial.

Se basan en un paradigma de aprendizaje y procesamiento de la información automático y se inspiran en el funcionamiento de las redes neuronales del sistema nervioso. Se componen de una capa de entrada y una de salida además de una o más capas ocultas. Cada una de las capas está formada por una o más neuronas que tienen su correspondiente función de activación. El resultado de cada neurona es enviado a la neurona siguiente para su correspondiente cálculo. En la capa de salida se produce la salida final de la ANN.

Se puede ver un ejemplo de ANN en la figura 1.3 con i neuronas en la capa de entrada, k en la de salida y una capa oculta con j neuronas.

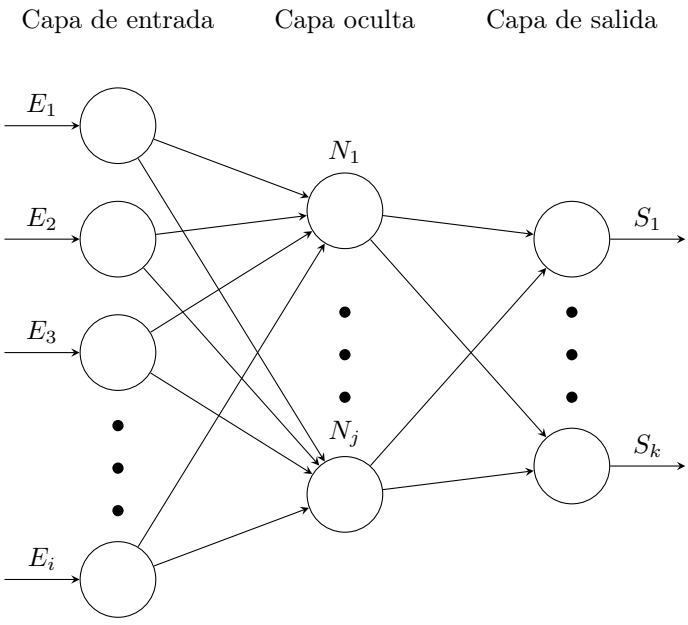

Figura 1.3: Ejemplo de red neuronal

Las ANN se utilizan también para tratar problemas de comprensión del habla [3] en los que como ya se ha descrito hace falta un etiquetado semántico de las frases.

En el ejemplo citado se hace uso de las ANN para extraer los conceptos semánticos asociados a las palabras de las frases de entrada y rellenar los campos en una representación basada en *frames* que son utilizados para la interacción con el usuario.

1.2.4 CRFs

Los CRFs (*Conditional Random Fields*) son modelos discriminativos "log-lineal" en los que se realiza una normalización a nivel de toda la frase.

Un CRF se define como un grafo de dependencias y una serie de características con un peso asociado que es calculado durante el proceso de aprendizaje. Los CRFs reúnen características de los sistemas generativos y discriminativos. Con ellos se puede tener en cuenta tanto las características de la entrada que se han aprendido de forma discriminativa como las decisiones previas a la hora de escoger la mejor etiqueta semántica en cada momento.

Es una de las técnicas de referencia actuales para la creación de sistemas de comprensión del habla [4] por su precisión a la hora de asignar etiquetas semánticas a cada palabra de las frases de entrada al sistema.

En el artículo que se acaba de citar se explican ampliamente los CRFs y se hace uso de ellos para realizar una comparación con los MEMMs (*Maximum entropy Markov models*). Los modelos no generativos de estados finitos como los MEMMs o los HMMs (*Hidden Markov Models*) poseen una desventaja respecto a los CRFs, que se trata del *label bias problem*. Esta debilidad consiste en el hecho de que las salidas de un estado dado en estos modelos solo compite contra otro, en lugar de hacerlo contra todas las transiciones del modelo como ocurre en los CRFs. En el artículo se explican también como funciona el algoritmo que implementa la estimación de parámetros en los CRFs.

Los CRFs serán explicados detalladamente en la sección 4.3 ya que han sido la técnica utilizada para entrenar los modelos semánticos multilingües obtenidos en este proyecto.

1.2.5 Máquinas de estados finitos

Las FSMs (*Finite State Machines*) son un modelo de computación que genera una salida a partir de una entrada dada y un conjunto de operaciones que se realizan de forma automática. En 1943 surge el modelo neuronal de McCulloch-Pitts como primera aproximación formal de las FSMs.

Una máquina de estados finitos se puede definir formalmente por una tupla con cinco componentes (Q, Σ, $q_0 \in Q$, $\delta\colon Q \otimes \Sigma \to Q$, $F \subseteq Q$) formada respectivamente por un conjunto finito de estados, un alfabeto finito, un estado inicial perteneciente al conjunto de estados, una función de transición y un conjunto de estados finales o de aceptación. El estado en el que se encuentra la máquina en cada momento (solo puede estar en uno de ellos a la vez) se denomina "estado actual". Al analizar la cadena entrada, la máquina pasa de un estado a otro y finaliza cuando se computa el último símbolo de la cadena. Si el estado en el que acaba pertenece al conjunto de estados finales, entonces se acepta la cadena.

En la figura 1.4 se puede observar un ejemplo FSM con las transiciones definidas y un total de tres estados, siendo uno de ellos inicial y final.

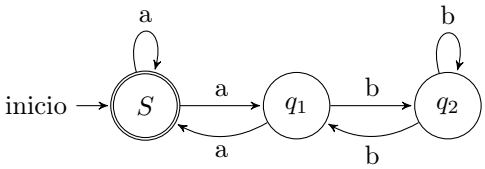

Figura 1.4: Ejemplo de autómata de estados finitos

Actualmente son usados también en comprensión del habla [5, 6, 7] principalmente para la decodificación semántica. En el artículo citado se desarrollan dos grafos. El primero de ellos consiste en un grafo de traducciones generado a partir de las traducciones de las frases de entrada al sistema. Este grafo está construido a partir de un lenguaje finito que representa la generalización de todas las traducciones. El segundo grafo trata la entrada para proporcionar la mejor ruta en el grafo de traducciones de acuerdo a un modelo semántico.

1.3 Objetivos

El presente proyecto trata de buscar un método de creación de un sistema de comprensión del habla multilingüe en los idiomas inglés y francés a partir de un sistema en castellano. Este sistema se ha aplicado a la tarea del *corpus* `DIHANA` de consulta de horarios, precios y servicios de trenes de largo recorrido en España, que se explica detalladamente en el Capítulo 2.

El propósito de un sistema de comprensión del habla es el de proporcionar una interpretación semántica a la frase de entrada en términos de unidades semánticas. Lo que se busca es identificar la información relevante que es necesaria para proporcionar una respuesta al usuario. Con la incorporación del multilingüismo se consigue un sistema de comprensión válido para una entrada tanto en inglés como en francés. De esta forma, un sistema que originalmente está diseñado pa-

ra tratar frases en castellano se adapta para hacerlo también con estos dos idiomas.

Para lograr este objetivo se puede hacer uso de dos aproximaciones diferentes.

- El primer caso se trata de la aproximación *test-on-source*, donde se espera una entrada al sistema en un idioma distinto a aquél para el que ha sido diseñado. Esta estrategia centra su esfuerzo en la traducción y el tratamiento de la entrada para poder ser analizada por el sistema de comprensión.

- En segundo lugar se encuentra la aproximación *train-on-target*, que es la que se ha decidido emplear en este trabajo para la implementación del multilingüismo. Se trata de una aproximación en la que el trabajo recae sobre la obtención de un nuevo sistema de comprensión en los idiomas a los que se quiere portar a partir del sistema original. Es decir, dado un sistema de comprensión del habla funcional para un idioma se obtiene otro para un idioma objetivo. Para conseguirlo es necesario traducir el *corpus* de entrenamiento del idioma original a los nuevos idiomas y entrenar los correspondientes modelos semánticos a partir de estas traducciones. Cabe destacar que el proceso de obtención de los nuevos modelos semánticos no consiste solo en la traducción del *corpus* de entrenamiento, sino también en la segmentación y el correcto etiquetado semántico de las frases de entrenamiento traducidas en los nuevos idiomas.

Por ello, este trabajo se divide en varios capítulos en los que se explican los diferentes pasos seguidos para la obtención de un sistema de comprensión del habla en inglés y francés a partir de un *corpus* disponible en castellano.

- Descripción del *corpus* (Capítulo 2): se describe el *corpus* utilizado tanto para el entrenamiento de los nuevos modelos semánticos como para las pruebas realizadas.

- Obtención del *corpus* multilingüe (Capítulo 3): en este capítulo se explica la forma de obtención de las frases de entrenamiento en inglés y francés así como el tratamiento realizado en todas ellas con el fin de normalizarlas y los traductores *online* utilizados.

- Sistema de comprensión del habla (Capítulo 4): se desarrolla el proceso de segmentación y de obtención de los segmentos de las frases de entrenamiento traducidas.

- Experimentación (Capítulo 5): se discuten los resultados obtenidos con la aproximación utilizada para la obtención del sistema de comprensión multilingüe y se compara con otras aproximaciones.

- Conclusiones y trabajo futuro (Capítulo 6): en este último capítulo se resume el trabajo realizado y se abre la posibilidad de lineas futuras de investigación.

Para conseguir las traducciones de las frases de entrenamiento se ha hecho uso de diferentes traductores *online* de propósito general (`Apertium`, `Bing`, `Google`, `Lucy` y `Systranet`), de los cuales se puede ver una explicación en la subsección 3.3.3.

Mediante estos traductores se han conseguido las traducciones tanto en inglés como en francés de un subconjunto de frases del *corpus* `DIHANA` que han sido normalizadas tal y como se explica en el Capítulo 3. Una vez conseguido esto, se ha realizado un proceso de segmentación cuya explicación se puede leer en la sección 4.4. Mediante este proceso se consigue alinear los segmentos traducidos con la frase y se etiqueta cada segmento con la etiqueta semántica que le corresponde.

Cuando se tienen los segmentos etiquetados correctamente, se procede a entrenar los modelos semánticos haciendo uso de CRFs. Se ha entrenado un modelo por cada traductor utilizado y otro en el que intervienen todos ellos. También se han realizado combinaciones con las traducciones de diferentes traductores. La idea de esto es la de poder realizar una comparación de los resultados obtenidos para cada modelo semántico diferente, tal y como se comenta en el Capítulo 6.

Cuando se tienen los modelos semánticos entrenados, el sistema de comprensión ya está preparado para recibir la consulta del usuario. Esta consulta en forma de lenguaje natural se interpreta por el modelo semántico y es etiquetada a nivel de concepto. A partir de estas etiquetas se genera una consulta formal en el lenguaje que entiende el sistema de información. Por último se construye la respuesta y ésta es transmitida al usuario.

Capítulo 2

Descripción del *corpus*

El *corpus* utilizado para el sistema de comprensión del habla del presente proyecto es el `DIHANA` [8]. Se trata de un *corpus* en castellano de diálogos transcrito y etiquetado manualmente a nivel de concepto. Está adquirido en tres laboratorios de investigación en diferentes universidades de España (Universidad del País Vasco, Universidad de Zaragoza y Universidad Politécnica de Valencia) siguiendo la técnica del Mago de Oz. Esta técnica se basa en una interacción persona-computador donde los usuarios interactúan con un sistema aparentemente independiente pero controlado por un experto, lo que permite que se presenten la mayoría de las características del habla espontánea.

`DIHANA` está formado por un total de 900 diálogos entre 225 usuarios (153 varones y 72 mujeres) formando un total de 6.278 turnos, de los cuales 4.887 son de entrenamiento, y posee un vocabulario de 811 palabras diferentes.

La tarea de este *corpus* consiste en un sistema de información sobre los horarios y precios de viajes de larga distancia en diferentes tipos de trenes por ciudades españolas.

2.1 Representación semántica del *corpus*

En el *corpus* se han definido un total de 30 etiquetas semánticas que se pueden clasificar en tres categorías diferentes:

1. Etiquetas independientes de la tarea: no están relacionadas directamente con el vocabulario específico de la tarea y no son tenidas en cuenta para realizar

la consulta al sistema de información. Por ejemplo: "coletilla", "consulta", "cortesia"...

2. Etiquetas dependientes de la tarea:

   a) Etiquetas que son atributos: la etiqueta está asociada a un segmento que contiene un valor relevante para la comprensión. Por ejemplo: "ciudad_destino", "ciudad_origen", "clase_billete"...

   b) Entidades del sistema de información: son aquellas etiquetas que guardan relación con la tarea del *corpus* y los valores consultados por el usuario. Estas etiquetas además se colocan entre las marcas < y > para diferenciarlas de las del punto *a*. Por ejemplo: "<afirmacion>", "<hora_llegada>", "<hora_salida>"...

La representación semántica del turno de usuario se basa en una segmentación de la frase donde a cada segmento se le asocia una etiqueta semántica. Se puede observar un ejemplo de la representación semántica del *corpus* en la figura 2.1. En este ejemplo se representa la frase "*Hola buenos días quería horarios de trenes a Sevilla desde Almería*" con los conceptos semánticos asociados a cada segmento de la frase.

Segmento	Etiqueta semántica
hola buenos días	cortesia
quería	consulta
horarios de trenes	<hora>
a Sevilla	ciudad_destino
desde Almería	ciudad_origen

Figura 2.1: Ejemplos de representación semántica del *corpus*

Como se ha explicado antes y tal y como se ha podido ver en la figura anterior, cada segmento de la frase tiene asociada una etiqueta semántica. Podemos representar por tanto una frase como una secuencia de segmentos con una etiqueta semántica asociada y un segmento como una secuencia de palabras.

En la figura 2.2 se puede ver una tabla donde se resumen las principales características del *corpus* etiquetado semánticamente.

Número total de turnos de usuario:	6.229
Número total de palabras:	47.222
Tamaño del vocabulario:	811
Número medio de palabras por turno de usuario:	7,6
Número total de segmentos semánticos:	18.588
Número medio de palabras por segmento semántico:	2,5
Número medio de segmentos por turno de usuario:	3,0
Número medio de muestras por unidad semántica:	599,6

Figura 2.2: Características del *corpus* etiquetado semánticamente

2.2 Sistema de información

Cabe recordar que el objetivo principal del sistema de comprensión es el de extraer la información que proporciona el usuario y transformarla en un formato que entienda el gestor de diálogo. En última instancia lo que el usuario necesita es recuperar información sobre un dominio determinado.

Por ello, cuando el sistema de comprensión del habla recibe la frase de entrada del usuario, asigna una secuencia de etiquetas semánticas a cada palabra de la frase. Cuando la frase está etiquetada se realiza la conversión al formato que entiende el gestor de diálogo. Esta evolución es realizada a partir de un conjunto de reglas que transforman esta representación intermedia en un *frame*, que consiste en una secuencia de conceptos y atributos con sus valores asociados. En este proceso se eliminan las etiquetas semánticas irrelevantes para la consulta y sus valores asociados y se reordenan las etiquetas siguiendo un orden preestablecido. En la figura 2.3 se puede ver un ejemplo de frase etiquetada semánticamente en el *corpus* `DIHANA` y otro de un ejemplo de *frame* obtenido a partir de esta frase. La frase de ejemplo es "*Sí quería saber el precio de un viaje de ida y vuelta de Valencia a Murcia*".

```
sí: <afirmacion>
quería saber: consulta                    (AFIRMACION)
el precio de: <precio>                    TIPO-VIAJE: ida_y_vuelta
un viaje de ida y vuelta: tipo_viaje      (PRECIO)
de Valencia: ciudad_origen                CIUDAD-ORIGEN: Valencia
a Murcia: ciudad_destino                  CIUDAD-DESTINO: Murcia
```

Figura 2.3: Ejemplo de frase etiquetada (izquierda) y su *frame* derivado (derecha)

Como se puede observar, toda la información que solicita el usuario cuando interacciona con el sistema está recogida en el *frame*. Se puede ver que el usuario

empieza con una frase afirmativa, lo cual se denota por el campo "`(AFIRMACION)`" seguido del campo "`(PRECIO)`", por lo que se deduce que el usuario quiere obtener el precio del viaje. Por otra parte, el campo "`TIPO-VIAJE: ida_y_vuelta`" indica al sistema que el usuario está solicitando un viaje de ida y vuelta. Por último, los campos "`CIUDAD-ORIGEN: Valencia`" y "`CIUDAD-DESTINO: Murcia`" contienen la ciudad desde la que el usuario quiere salir y también a la que quiere llegar.

Otro ejemplo se puede ver en la figura 2.4 con la frase "*sí quería saber el tipo de tren de las siete y tres de la mañana*".

```
sí: <afirmacion>
quería saber: consulta                          TIPO-VIAJE:nil
el tipo de tren: <tipo_tren>                    (TIPO-TREN)
de las siete y tres de la mañana: hora          HORA:+-30(07.03.AM)
```

Figura 2.4: Ejemplo de frase etiquetada y su *frame* derivado

En este último ejemplo se observa por el campo "TIPO-VIAJE:nil" que el usuario no ha especificado el tipo de viaje pero sí que solicita un tipo de tren por el campo "(TIPO-TREN)". Además, en el campo "HORA:+-30(07.03.AM)" se concreta la hora a la que el usuario quiere realizar el viaje.

Con el *frame* ya generado, el gestor de diálogo es capaz de recuperar de la base de datos la información necesaria que es mostrada finalmente al usuario.

2.3 *Corpus* multilingüe

El *corpus* `DIHANA` fue dividido en un conjunto de entrenamiento de 4.887 turnos y otro de prueba con 1.000 turnos. El conjunto de entrenamiento posee un total de 14.517 segmentos etiquetados, 36.930 palabras y un vocabulario de 716 palabras. El conjunto de prueba fue traducido manualmente al francés e inglés y pronunciado por usuarios nativos. El conjunto de pruebas en francés está compuesto por 1.000 turnos, 500 de los cuales fueron pronunciados por usuarios nativos. El conjunto de pruebas en inglés consiste en 1.000 turnos y todos ellos fueron pronunciados por usuarios nativos.

En la tabla 2.1 se puede ver un resumen de las principales características de los conjuntos de prueba tanto para inglés como para francés y una comparación con las frases en castellano.

En este trabajo se ha obtenido las traducciones de todas las frases del conjunto de entrenamiento del *corpus* realizando un preproceso para preparar los ficheros

de traducción y un postproceso para normalizar las traducciones tal y como se explica en el Capítulo 3. Además, ha sido necesario posteriormente un proceso de segmentación y alineamiento de los segmentos traducidos con cada frase traducida. Esto permite tener un conjunto de entrenamiento etiquetado semánticamente en inglés y francés a partir del *corpus* original. Este problema y su solución se abordan en el Capítulo 4.

Inglés			
	`Frases`	`Palabras`	`Vocabulario`
texto	1000	7700	448
Francés			
	`Frases`	`Palabras`	`Vocabulario`
texto	1000	7669	522
texto	500	3788	358
Castellano			
	`Frases`	`Palabras`	`Vocabulario`
texto	1000	7637	464

Tabla 2.1: Características de los conjuntos de prueba

Capítulo 3

Obtención del *corpus* multilingüe

3.1 Motivación

La obtención del *corpus* multilingüe ha sido una de las partes más importantes y delicadas del presente proyecto. Partimos de la base de que los traductores *online* utilizados para las traducciones de distintas frases del *corpus* `DIHANA`, descrito anteriormente en el Capítulo 2, están construidos de forma diferente y por tanto realizan un tratamiento léxico que difiere significativamente del de sus respectivos competidores. Es por esta razón que resulta necesario realizar un preproceso de las frases que se traducirán en cada uno de los traductores y un postproceso en las propias traducciones con el objetivo de que todas estén normalizadas de la misma manera, ya que como se explicará en la sección 3.4, cada traductor *online* genera unas traducciones con un tratamiento léxico diferente, como puede ser el uso de números en vez de letras o la utilización de caracteres especiales para aclarar algún aspecto sobre la traducción, tal y como se verá en los siguientes apartados.

3.2 Herramientas utilizadas

Para la implementación de los *scripts* utilizados en el preproceso y el postproceso se ha hecho uso de la herramienta `sed` de los sistemas UNIX, además de los lenguajes de programación `AWK` y `Python`, que se describen brevemente a continuación.

3.2.1 sed

stream editor es un potente editor de flujo [9] propio de los sistemas UNIX diseñado por Lee E. McMahon entre los años 1973 y 1974 que permite realizar modificaciones para una entrada de texto dada. Resulta además una herramienta especialmente potente cuando se requiere del uso de expresiones regulares y patrones de búsqueda. Para el presente proyecto ha sido una de las herramientas más útiles en el postproceso por resultar realmente eficaz y rápida en la normalización de las traducciones, como se explicará en la sección 3.4.

3.2.2 AWK

Aho Weinberger Kernighan es un lenguaje de programación interpretado [10] que recibe el nombre de las iniciales de los apellidos de sus creadores y que apareció por primera vez el año 1977. Es una de las primeras herramientas que apareció en los sistemas UNIX y que posteriormente se convirtió en una de las herramientas imprescindibles en los citados sistemas. Al igual que la herramienta `sed` concentra su mayor utilidad en el tratamiento de texto y es a su vez un potente motor de búsqueda y tratamiento de patrones, por lo que su uso tanto en el preproceso, para generar el fichero de traducción, como en el postproceso para la normalización de las traducciones se ha convertido en imprescindible para el proyecto.

3.2.3 Python

El nombre de este lenguaje de programación interpretado [11] es un homenaje de su creador, Van Rossum, a los humoristas británicos Monty Python, siendo el año 1991 el de la primera aparición del código del lenguaje. Destaca por su flexibilidad y baja curva de aprendizaje, además de ser uno de los más populares y extendidos lenguajes de programación de la actualidad tal y como explica se puede observar en la revista mensual Tiobe [12]. Para este proyecto se ha hecho uso de la versión 2.7 para la parte del postproceso, concretamente en el alineamiento de los segmentos traducidos con las frases traducidas, cuya explicación se puede leer en la sección 4.4.

3.3 Preproceso del fichero a traducir

En primer lugar debemos especificar el contenido que se debe traducir. Dado que necesitamos traducciones tanto de las frases enteras como de sus respectivos segmentos, con las que posteriormente se realizará un postproceso que se explicará más adelante en el sección 3.4, se ha optado por generar un fichero formado por secuencias de palabras en las que se incluye a dichas frases y los segmentos que le corresponden. De esta forma, se ha usado un separador que marca el inicio de una frase y que los traductores *online* ignorarán, seguido de un salto de linea y la frase completa, para a continuación colocar cada uno de los segmentos separados por el mismo salto de linea. En el ejemplo de la figura 3.1 se puede observar un ejemplo de dos de las frases del fichero con sus respectivos segmentos que serán traducidas por los traductores *online*.

```
|||
el próximo sábado día trece antes de las once
el próximo sábado día trece
antes de las once
|||
sí quiero los horarios antes de las once
sí
quiero los horarios
antes de las once
|||
```

Figura 3.1: Subconjunto de grupos a traducir

3.3.1 Palabras invariantes

Cabe destacar que uno de los requisitos fundamentales de las traducciones es que deben preservar ciertas palabras o conjunto de palabras ya normalizadas en el *corpus* DIHANA y que en los ficheros de test encontraremos escritas exactamente de la misma manera, como es el caso de las ciudades, los tipos de trenes o algunos nombres de fiestas, como los que se puede observar en negrita en la figura 3.2, y del que se puede ver el listado completo en el anexo A.

```
U0410:quiero conocer horario y precio a Valencia
U0642:sí repítame los horarios del talgo que me viene mejor
U0641:a qué hora sale el intercity
```

Figura 3.2: Ejemplo de palabras invariantes

Uno de los problemas que encontramos en los traductores y que está relacionado con estas palabras es el hecho de que no hay un método consistente en todos los traductores utilizados para mantener dichas palabras sin traducir, por lo que ha sido necesario buscar una solución alternativa que sirviese para cualquier traductor *online* y que preservase las palabras originales. En principio se intentó mantener las palabras originales convirtiéndolas a mayúsculas con la intención de que los traductores las dejasen intactas para posteriormente devolverlas a su estado original, pero algunos de los traductores utilizados para el trabajo traducían de igual manera estas palabras.

Por este motivo, finalmente se ha optado por guardar todas las palabras o conjunto de palabras que no deben ser alteradas por los traductores en un fichero que hace de diccionario, que se puede visualizar en el siguiente enlace:
`www.github.com/carmilso/TFG/blob/master/dictionary`, el cual actúa como indexador de las palabras a mantener, donde cada palabra ocupa una linea diferente. De esta forma se sustituyen las apariciones de dichas palabras por un par de paréntesis que contienen su correspondiente índice en el diccionario, que como se ha explicado anteriormente hace referencia a la linea que ocupa en el mismo. En la figura 3.3 se puede observar un ejemplo del resultado de la sustitución realizada. Además, las palabras a preservar se han indexado a partir de la linea cincuenta dado que algunos traductores están especialmente entrenados para detectar y traducir números con una alta frecuencia de aparición, como es el caso del traductor *online* de `Google` que se describe en la subsección 3.3.3, a pesar de estar entre caracteres especiales como son los paréntesis, por lo que no se respetaba el índice y esto provocaba problemas posteriores a la hora de sustituir la correspondiente palabra del diccionario.

```
|||
quiero conocer horario y precio a ((132))
quiero conocer
horario
y precio
a ((132))
|||
sí repítame los horarios del ((125)) que me viene mejor
sí
repítame
los horarios
del ((125))
que me viene mejor
|||
```

Figura 3.3: Ejemplo de sustituciones

El *script* utilizado para generar el fichero que será traducido en el que se han resuelto todas las cuestiones tratadas hasta el momento se puede visualizar en el siguiente enlace:

`www.github.com/carmilso/TFG/blob/master/scripts/prepareSentences.sh`

3.3.2 Conversión a HTML

Además, ha sido necesario encontrar un mecanismo que permitiese traducir todas las frases debido a las restricciones que imponen algunos de los traductores *online* utilizados al número de palabras que se pueden traducir. Finalmente, y dado que esta restricción no existe en la traducción de páginas web, se ha implementado un *script* para convertir el fichero con las frases y los segmentos, con las palabras o conjunto de palabras a mantener ya sustituidas por su correspondiente índice en el diccionario, a formato `HTML`. El *script* se puede visualizar en el siguiente enlace:

`www.github.com/carmilso/TFG/blob/master/scripts/toHtmlFormat.sh`

Se ha hecho uso del `Servicio de Publicación Estándar` para la publicación del fichero ya en formato `HTML`, pudiendo hacer uso así de la *web* personal que ofrece la universidad para conseguir las traducciones de las frases y sus segmentos.

En el caso del traductor `Systranet`, que se describe en la subsección 3.3.3, ha sido necesario dividir el fichero principal en tres diferentes dado que el límite actual de la página *web* para poder ser traducida es de 512.000 *bytes*.

La publicación donde se encuentran un subconjunto de las frases con sus segmentos es accesible desde:

`http://personales.alumno.upv.es/carmilso/index.html`

Un subconjunto de las tres publicaciones preparadas para el traductor `Systranet` son accesibles desde:

`http://personales.alumno.upv.es/carmilso/index0.html`
`http://personales.alumno.upv.es/carmilso/index1.html`
`http://personales.alumno.upv.es/carmilso/index2.html`

3.3.3 Obtención de las traducciones

Para conseguir las traducciones del fichero preprocesado se ha usado los siguientes traductores *online*:

- **Apertium**: es una plataforma de traducción automática de código abierto financiada por el `Gobierno de Español` y la `Generalitat de Catalunya` y desarrollado en la `Universitat d'Alacant` que en un principio se diseñó para la traducción de lenguas relacionadas y a la que en los últimos años se le ha dotado de una mayor potencia.

- **Bing**: es un traductor *online* propiedad de `Microsoft` que soporta la traducción entre más de cincuenta idiomas. Junto al traductor de `Google` es una de las referencias mundiales de la actualidad en traducción automática.

- **Google**: ha logrado convertirse en uno de los traductores *online* de referencia que soporta noventa idiomas. Es además el principal competidor del traductor de `Bing`.

- **Lucy**: es un proveedor de traducciones automáticas con más de treinta años de experiencia en el campo de la traducción automática que ofrece además de un servicio de traducción *online*, un servicio de consultoría personalizado para empresas.

- **Systranet**: es un motor de traducción automática *online* con más de cuarenta idiomas disponibles que dispone también de un sistema de pago con características adicionales de traducción.

Una vez obtenidas las traducciones en inglés y francés de todas las frases con sus segmentos, éstas se han almacenado siendo discriminadas por traductor e idioma, resultando el nombre de fichero `<nombre_del_traductor>_<idioma>.log` para cada una de ellas. Todas las traducciones han sido guardadas en una carpeta con nombre `logs`, para poder realizar sobre ellas el postproceso que se explicará a continuación en la sección 3.4.

3.4 Postproceso de las traducciones

Como ya se ha comentado anteriormente, uno de los principales problemas de los traductores *online* es que no generan resultados normalizados y por tanto cada uno de ellos transcribe de forma diferente palabras con un significado similar. Por ello ha sido necesario implementar una serie de métodos aplicables a todas las traducciones con los que poder hacer un uso equivalente de todas ellas.

En las siguientes subsecciones se describe punto por punto cada uno de los casos que han sido detectados y tratados.

3.4.1 Restauración de las palabras invariantes

En primer lugar lo que se ha hecho ha sido restaurar todas aquellas palabras o conjunto de palabras que debían permanecer invariantes, por lo que se ha realizado el proceso inverso al explicado previamente en la subsección 3.3.1. Se puede ver un ejemplo del resultado de la restauración de las palabras clave en la figura 3.4, que corresponde a un subconjunto de las traducciones para inglés obtenidas del traductor *online* de `Google`. Como se puede observar, el hecho de traducir por separado las frases completas de los segmentos provoca que la traducción de los segmentos en ocasiones no sea coherente con la frase, lo cual se justifica por la falta de contexto.

```
|||
i want to meet schedule and price Valencia
i want to meet
schedule
and price
to Valencia
|||
yes repeats me schedules talgo that comes better
if
repeats me
the schedules
of talgo
that comes better
|||
what time does the intercity
what time does
the intercity
|||
```

Figura 3.4: Ejemplo de palabras clave restauradas

3.4.2 Signos de puntuación

Para normalizar las traducciones se han eliminados algunos signos de puntuación, que se han obtenido como se puede ver en el código 3.4.2. A continuación se muestran los que no se han eliminado y una explicación del por qué.

- Guión (-): Los guiones son utilizados en las horas tanto en inglés como en francés, además de para algunas palabras propias del francés, como podemos ver en el ejemplo de la figura 3.5 para el caso del inglés y en la figura 3.6 para el caso de francés.

- Barra baja (_): Las barras bajas se mantienen únicamente para no corromper las palabras invariantes, explicadas en la subsección 3.3.1, ya que las palabras compuestas hacen uso de este signo de puntuación para mantenerse unidas como si de una única palabra se tratara.
- Comilla simple ('): Se han mantenido las comillas simples dado que tanto en inglés como en francés son necesarias para muchas palabras, como se puede ver en el ejemplo de la figura 3.7 para el caso del inglés y en la figura 3.8 para el caso de francés.
- Barra de separación (|): Como ya se ha explicado en la sección 3.3, este signo replicado tres veces se corresponde con el separador entre cada grupo de frases y segmentos, por lo que es imprescindible mantenerlo para postprocesos posteriores.
- Dos puntos (:): Este signo de puntuación se ha mantenido ya que como se explicará posteriormente en la subsección 3.4.4, es necesario para convertir las horas en formato numérico a palabras. Sin embargo, hay un caso en el que este signo de puntuación se debe eliminar y es aquél en el que no se encuentra acompañado de un número. Para solucionar este caso se han buscado y eliminado aquellos ":" cuya anterior o posterior palabra no fuese un número, independientemente de los espacios que hubiese en medio.

```
the twenty-four
the monday day twenty-two
for twenty-five june
```

Figura 3.5: Ejemplos de palabras con guión en inglés

```
le vingt-quatre mai
aux cinq et cinquante-cinq
pour passer le week-end
```

Figura 3.6: Ejemplos de palabras con guión en francés

```
i'd like to go ave
i don't want to know price
before ten o'clock
```

Figura 3.7: Ejemplos de palabras con comilla simple en inglés

```
le prix d'unvoyage
il me plairait qu'il fût un ave
dans l'après-midi
```

Figura 3.8: Ejemplos de palabras con comilla simple en francés

Con ello además de los signos de puntuación ya comentados y otros irrelevantes que han sido eliminados directamente como los interrogantes, las exclamaciones, los puntos o las comas, obtenemos también aquellos que son introducidos por algunos traductores cuando no comprenden la semántica de una expresión. Estos signos son dos:

- Asterisco (*): Los asteriscos son utilizados por el traductor *online* `Apertium`, explicado anteriormente en la subsección 3.3.3, para marcar las palabras que no han sido analizadas. Podemos ver un ejemplo de las mismas en la figura 3.9.
- Almohadilla (#): Las almohadillas se utilizan también por parte de `Apertium` para indicar que ha habido problemas para procesar la palabra durante el análisis morfológico. En la figura 3.10 se puede ver un ejemplo.

```
Du train *lanzadera
Ne *muchísimas grâces voilà tout
Le mardi *mismamente
```

Figura 3.9: Ejemplos de traducciones con asterisco

```
Il a #avoir une erreur
Le sept mai deux #mil quatre
Aux deux de #soir
```

Figura 3.10: Ejemplos de traducciones con almohadilla

Se han utilizado las siguientes instrucciones para extraer los signos de puntuación.

```
1  $ cat logs/*.log | grep -o '[[:punct:]]' | sort -u
```

Código 3.1: Instrucciones para obtener los signos de puntuación

3.4.3 Espacios y saltos de linea

Cada linea escrita en las traducciones, ya sea una frase, un segmento de la frase o el separador, es separada de su linea predecesora por una linea en blanco. Esto ocurre para todos los traductores *online* utilizados en el presente proyecto, así que estas lineas se han eliminado de las traducciones de todos los traductores.

En el caso del traductor *online* de `Google` además se añade un espacio en blanco solo al inicio de la primera linea de la traducción, quedando así el separador que hay en la primera linea y que marca el inicio de la primera frase con sus segmentos desplazada a la derecha por un espacio, por lo que este caso también ha sido tratado.

En la subsección 3.4.2 se ha explicado que los dos puntos se tenían que mantener a excepción de aquellos casos en los que no estuviesen acompañados por un número, y que la comilla simple se tenía que mantener en todos los casos por ser un signo propio del inglés y el francés. En el caso del traductor `Systranet` para la traducción en francés se han detectado casos en los que se separaba por un espacio la comilla simple de la palabra que la acompañaba, como se puede ver en la figura 3.11, por lo que se ha tratado este caso eliminando dicho espacio de separación.

pourrait dire je si certains **d' eux** est
en effet je m'aimerais savoir le prix du billet **d' allée**
d' ici même

Figura 3.11: Ejemplo de comilla simple con espacio

3.4.4 Tratamiento numérico

El tratamiento de los números ha sido el más complejo de todo el postproceso de las traducciones debido a la alta variabilidad en la representación de los mismos

Es importante señalar que ha sido necesario convertir todos aquellos que estaban representados de forma numérica a palabras ya que esta es la forma en la que se encuentran representados en los ficheros de test, por lo que ha habido que analizar cada caso por separado y realizar un tratamiento que se adaptase a todas las situaciones, que se describen en las siguientes secciones.

Cabe destacar que un mismo traductor no tiene el mismo comportamiento esperado para todos los casos numéricos, sino que dependiendo del contexto en el que los números se encuentren los trata de una manera u otra. El único caso que se mantiene constante es el de aquellos que están entre dos paréntesis y que representan a las palabras invariantes cuyo tratamiento ha sido explicado en la subsección 3.4.1.

Tratamiento de las horas

En primer lugar ha sido necesario encontrar una forma de normalizar todos los casos de la misma manera, para posteriormente poder tratar las horas y convertirlas a palabras mediante un *script* que se explicará más adelante. Para procesar las horas se han realizado tres tratamientos diferentes, uno por cada caso que se ha localizado en las traducciones de los diferentes traductores.

1. El traductor de `Google` en las traducciones de francés utiliza en ocasiones una "h" como separación entre la hora y los minutos. Este caso se ha tratado convirtiendo las "h" en ":", ya que únicamente se han contado quince casos del uso de la "h" como separador entre la hora y los minutos frente a trescientos ochenta y tres casos del uso de ":", facilitando de esta manera la posterior conversión de números a letras. Se puede observar un ejemplo de este caso en la figura 3.12.

2. Se han detectado casos como los que se ven a modo de ejemplo en la figura 3.13. Estas situaciones han aparecido únicamente en las traducciones de francés con el traductor de `Google`, igual que en el caso anterior. En estos casos, y aprovechando que el idioma en el que se producen es el francés se ha

optado por tratarlos sustituyendo el separador de las horas, ya sea "-" o "-à-", por una única "à", ya que lo que se pretende expresar con estos separadores es un periodo donde se marca la hora de inicio y la de fin.

3. En estos momentos de la ejecución del *script* se generan potencialmente nuevos casos, como se observa en la figura 3.14, en los que aparece una hora precedida y seguida de un espacio. Esto se debe a que en su estado anterior esta hora estaba unida a otra y alguna de las dos no tenía minutos, por lo que al separarlas, esta hora queda aislada. Posteriormente se explicará que esto es un problema únicamente para las doce horas por no poder interpretarlas correctamente cuando están expresadas como un par de ceros, por lo que se convierte todo par de ceros precedidos y seguidos por uno o más espacios en un doce.

```
à 00h25
de quitter à 7h30 et 6:30 le calendrier
```

Figura 3.12: Ejemplos de horas con "h" como separador

```
2-30 ou alors
le 12 Avril midi 00-à-02h00
```

Figura 3.13: Ejemplos de traducciones con almohadilla

```
feriez horaires Logroño le 12 avril midi 00 à 02:00
```

Figura 3.14: Ejemplo de hora aislada

Conversión a palabras

A diferencia de los anteriores casos, que se han tratado mediante la implementación de *scripts* en `bash`, para convertir los números detectados en las traducciones a palabras se ha hecho uso del lenguaje de programación `Python` ya explicado en la subsección 3.2.3 por la cantidad de librerías disponibles y la facilidad de uso que proporciona.

Se ha utilizado la librería `num2words` para convertir los números a palabras que está basada en una librería anterior con nombre `pynum2word` creada por Taro Ogawa en el año 2003. Actualmente es mantenida como software libre por Virgil Dupras y soporta trece idiomas diferentes. La función para convertir los números a palabras debe ser llamada pasándole como mínimo un argumento con el número a convertir pero permite además especificar un segundo argumento en el que se indica si el resultado debe ser un número ordinal o cardinal y un tercer argumento en el que se especifica el idioma objetivo al que se quiere convertir el número. Por defecto los números se devuelven en formato cardinal y en idioma inglés.

Ha habido que analizar todos los casos posibles de inglés y francés y construir un conversor a medida para cada idioma y poder transformar correctamente todos los casos numéricos en palabras, para lo que se ha hecho uso de dos *scripts* que llevan a cabo esta tarea. El *script* que trata la conversión de los casos numéricos en las traducciones en francés ha sido cedido por los tutores del proyecto debido a la dificultad en la implementación del mismo por el desconocimiento sobre este idioma. Sin embargo, el *script* para las traducciones en inglés sí ha sido implementado desde cero para el presente proyecto, que es el que se procede a explicar a continuación.

Para analizar todos los casos numéricos, y teniendo en cuenta que ya se han tratado los diferentes casos numéricos existentes para tener las traducciones normalizadas, se han utilizado tres expresiones regulares que pretenden abarcarlos todos.

```
1 hourspattern = compile('\d+:\d+(:\d+)?')
2 numeralspattern = compile('\d+(th|st|nd|rd)(-[a-zA-Z]+)?')
3 worddatepattern = compile('[a-zA-Z]+\d+')
```

Código 3.2: Expresiones regulares para detectar números

Podemos observar que de esta manera tenemos control sobre todos los casos explicados anteriormente. En primer lugar encontramos la variable `hourspattern`, con la que tenemos acceso a las horas unidas a los minutos mediante dos puntos, que han sido explicadas en el punto 1 del tratamiento de las horas. Para seguir, con la variable `numeralspattern` podemos acceder a aquellos números que por representar una fecha tienen un formato ordinal y como consecuencia van acompañados por uno de los siguientes sufijos: "th", "st", "nd" o "rd". Además se ha detectado un caso en la traducción de `Bing` para el inglés, `19th-Friday`, en la que además se añade un guión con el día de la semana, por lo que se ha añadido esa posibilidad a la expresión regular que estamos describiendo. Por último, encontramos la variable `worddatepattern`, con la que analizamos los cuatro casos aislados que aparecen en la traducción de `Bing` para el inglés y que se pueden ver en la figura 3.15 en los que aparece una palabra pegada a un número.

```
departure17
on17
Madrid29
no26
```

Figura 3.15: Casos aislados del traductor `Bing`

Una vez concretadas las variables que se utilizarán para manipular los diferentes casos numéricos se define una lista con los meses, que servirán para comprobar si el número analizado corresponde al día de un mes o es independiente. Se ha

definido también una función que es la que se encarga de transformar el número que recibe como argumento a palabras, a la que además se le especifica también si el número es ordinal o cardinal, siendo cardinal la opción por defecto.

```
def toWord(num, ord=False):
  return strip(num2words(int(num), ordinal=ord))
```

Código 3.3: Función para convertir números a palabras

Para poder trabajar todos los casos se ha analizado cada traducción linea a linea con todos los tratamientos anteriormente comentados ya realizados. La estructura del *script* donde se ve claramente la estructura con la que se localizan los diferentes casos numéricos es la que se muestra a continuación.

```
if word.isdigit():
  (...)
elif hourspattern.match(word):
  (...)
elif numeralspattern.match(word):
  (...)
elif worddatepattern.match(word):
  (...)
else:
  (...)
```

Código 3.4: Estructura básica del conversor

Para seguir, se analiza cada palabra de la linea y se intenta localizar cuatro casos diferentes.

1. La palabra es un dígito: el hecho de que se cumpla esta condición implica que la palabra que está siendo analizada es un número aislado. En este caso puede pasar que la palabra que le acompaña directamente bien sea por la izquierda o por la derecha sea un mes, pero también podría ser que esté seguida por un "*of*", lo cual denotaría que hace referencia al día de un mes, o precedida por un "*the*", en cuyo caso se trataría también de un número que indica o bien el día de un mes o bien un número de ordenación. En cualquiera de estos casos hay que transformar el número en palabras que representen a ese número en formato ordinal.

2. La palabra es una hora: en caso de que la palabra haga *matching* con la expresión regular que representa las horas, cuyo formato se ha explicado en el punto 1 del tratamiento de las horas, se ha de tratar por separado las horas y los minutos. Si la hora son uno o dos ceros entonces hay que convertir ese cero o ceros en un doce, ya que la función `toWord` no interpreta que se está haciendo referencia a una hora y por tanto escribiría la palabra "zero". En el caso de los minutos se ha llegado a un convenio para decidir cómo deben ser

interpretados, ya que se han tratado un total de cinco casos diferentes para el inglés.

- 00: se escribe en primer lugar la hora en palabras seguida por un "o'clock".
- 15: en este caso se escribe "quarter past" seguido de la hora en palabras.
- 30: para cuando pasan treinta minutos de la hora se ha decidido escribir "half past" con la hora en palabras a continuación.
- 45: para este caso se ha escrito "quarter to" y a continuación la hora en palabras.
- < 30: si los minutos son inferiores a treinta y además no se han cumplido las condiciones anteriores, entonces se escriben los minutos en palabras seguido de un "past" y las horas en palabras.
- $<= 60$: en el caso de que tampoco se haya cumplido la condición de que los minutos sean menores de treinta y tampoco ninguna de las condiciones anteriores, entonces se escribe en primer lugar la hora y a continuación los minutos en palabras.

3. La palabra es un número ordinal: si la palabra que se está analizando es un número que además tiene como sufijo uno de los términos citados anteriormente que denotan una ordenación, entonces se transforma el número en palabras a formato ordinal y se elimina el sufijo. En el caso de que además el número estuviese unido a un guión como es el caso anteriormente citado, se elimina el guión y se deja un espacio entre el número ya transformado en palabras y la palabra que previamente estaba a la parte derecha del guión.

4. La palabra está compuesta por una palabra y un número: este es el caso que se puede ver en la figura 3.15. Para tratarlo se escribe la palabra y a continuación se comprueba si la palabra anterior o posterior está en la lista de los meses. En caso afirmativo se escribe el número en palabras con formato ordinal, mientras que en caso contrario se escribe en formato cardinal.

En el caso de que ninguna de estas condiciones se cumpla, simplemente se añade la palabra a la frase final de salida por no haberse detectado ningún caso numérico.

El *script* donde se han implementado todos los tratamientos que se han comentado en el postproceso de las traducciones se puede visualizar en el siguiente enlace:

`www.github.com/carmilso/TFG/blob/master/scripts/treatTranslations.sh`

3.4.5 Resultados

Después de haber realizado el preproceso del fichero que posteriormente se ha traducido y el postproceso y normalización de cada una de las traducciones, se han guardado en la carpeta `logs` las traducciones normalizadas cuyo nombre es `<nombre_del_traductor>_<idioma>_treated.log`. En la figura 3.16 se puede observar un ejemplo de las traducciones antes y después del proceso de normalización.

```
|||                                          |||
would leave on March 24                      would leave on march twenty-fourth
I would want                                 i would want
get out                                      get out
the March 24                                 the march twenty-fourth
|||                                          |||
I wanted to go to ((129)) on Saturday        i wanted to go to Toledo on saturday
I wanted to go                               i wanted to go
to ((129))                                   to Toledo
next Saturday                                next saturday
|||                                          |||
yes that leaves at 0:25                      yes that leaves at twenty-five past twelve
if                                           if
the leaves                                   the leaves
at 0:25                                      at twenty-five past twelve
|||                                          |||
```

Figura 3.16: Ejemplos de normalizaciones realizadas

En los siguientes capítulos se explicará la utilización que se ha hecho de las traducciones normalizadas y la experimentación realizada.

Capítulo 4

Sistema de comprensión del habla multilingüe

4.1 Planteamiento del problema

En este punto del proyecto ya se han obtenido y procesado todas las traducciones para ser normalizadas como se ha explicado en el Capítulo 3.

En el Capítulo 1 se ha realizado una introducción a la comprensión del habla. Como se ha comentado, es una técnica enmarcada dentro del conjunto denominado como "Ingenierías del lenguaje". Igual que para el contexto del presente proyecto, una de las principales aplicaciones de los sistemas de comprensión del habla es la de proporcionar de interfaz de un sistema de información con el que el usuario interactúa para obtener algún tipo de información.

Como ya se ha explicado, partimos del *corpus* `DIHANA`. Es un *corpus* etiquetado a nivel de concepto del cual se puede ver una explicación detallada en el Capítulo 2. Se dispone de una serie de frases segmentadas y etiquetadas semánticamente. El objetivo que se ha planteado llevar a cabo en este proyecto es el de conseguir un sistema de comprensión a partir del *corpus* `DIHANA` para los idiomas inglés y francés.

Si el proceso de generar un *corpus* paralelo multilingüe se hiciese a mano sería una tarea que consumiría una gran cantidad de tiempo y recursos. Es por ello que se hace uso de herramientas de traducción automática con las que se traduce un subconjunto del *corpus* original y se entrena un sistema de comprensión en inglés

y francés. Este *corpus* debe estar disponible en los idiomas en los que se quiere obtener el sistema de comprensión.

4.2 Estrategias adoptadas

Como ya se ha explicado anteriormente, el objetivo de este proyecto es conseguir un *corpus* multilingüe a partir del *corpus* `DIHANA`. Para ello se ha hecho uso de cinco traductores *online* de propósito general (`Apertium`, `Bing`, `Google`, `Lucy` y `Systranet`).

Existen dos aproximaciones a la hora de construir un sistema de comprensión del habla multilingüe. La utilizada en el presente proyecto recibe el nombre de *train-on-target* y ha sido utilizada en otro proyecto [13] para el *corpus* en francés `MEDIA`. Sin embargo también existe una aproximación diferente con el mismo objetivo denominada *test-on-source* cuya explicación se puede ver en la valoración de los resultados en la sección 5.3.

En la aproximación seguida para este proyecto, la estrategia que se plantea es la de traducir las frases del *corpus* y cada uno de los segmentos de las frases a los idiomas en los que se quiere obtener el *corpus*. Después de tener las traducciones obtenidas de cada traductor utilizado, éstas han de ser procesadas para su correcta normalización. Se debe llevar a cabo además un proceso de segmentación en el que se alineen las traducciones de los segmentos con las traducciones de las frases. Esto nos permite obtener la correspondencia de las etiquetas semánticas en la frase traducida.

Una vez obtenido un *corpus* en cada uno de los idiomas objetivo para cada traductor, se realiza el entrenamiento de las frases con sus etiquetas semánticas. Después de este último proceso se obtienen los modelos semánticos para el *corpus* en ambos idiomas.

4.3 CRFs

Los CRFs [4] (*Conditional Random Fields*), también denominados Campos de Markov aleatorios, son modelos generativos que tratan de clasificar cada una de las palabras dentro de un conjunto de etiquetas. Son actualmente muy utilizados para la comprensión del habla debido a su gran eficacia en la segmentación y el etiquetado de datos.

Este ha sido el modelo estadístico escogido para abarcar el problema del entrenamiento de los modelos semánticos en este proyecto.

Básicamente los CRFs tratan de encontrar la mejor etiqueta semántica en cada momento teniendo en cuenta las decisiones tomadas previamente y las características aprendidas a partir de los datos de entrenamiento. Para ello, se aprenden sus parámetros por máxima verosimilitud a partir de N muestras de entrenamiento previamente etiquetadas con sus correspondientes etiquetas semánticas.

Mientras que en los HMMs (*Hidden Markov Models*) cada estado S_i depende de la observación O_i, en los CRFs cada estado puede depender de más de una observación simultáneamente.

La probabilidad condicional de una serie de etiquetas semánticas $c_1,\ c_2\ \ldots\ c_N$ dada una serie de palabras de entrada $w_1,\ w_2\ \ldots\ w_N$ se puede expresar tal y como se muestra en la Ecuación 4.1.

$$p(c_1^N|w_1^N) = \frac{1}{Z}\left(\prod_{m=1}^{M} \theta_m \cdot h_m(c_{n-1}, c_n, w_{n-k}^{n+k})\right) \tag{4.1}$$

En la ecuación que se acaba de mostrar, θ_m representa el vector de parámetros que se aprende a partir de los datos de entrenamiento etiquetados mientras que $h_m(c_{n-1}, c_n, w_{n-k}^{n+k})$ son las dependencias entre la entrada (palabras o características aprendidas durante el proceso de entrenamiento) y los conceptos de la salida. Z es el factor de normalización aplicado cuya definición se puede ver en la Ecuación 4.2.

$$Z = \sum_{\hat{c}_1^N} \prod_{n=1}^{N} (\hat{c}_{n-1}, \hat{c}_n, w_{n-k}^{n+k}) \tag{4.2}$$

En este caso, $\hat{c}_{n-1}$ es el concepto que se obtiene como salida para la palabra w_{n-1} (la palabra previa) y $\hat{c}_n$ es el concepto obtenido para la palabra w_n (palabra actual).

4.3.1 Implementación usada

En este proyecto se ha usado un *toolkit* implementado en el lenguaje de programación `C++` con nombre `CRF++` [14].

Se trata de un programa de software libre liberado bajo licencia GPL (*General Public License*) que permite el entrenamiento de modelos semánticos usando CRFs. Posee las características que se enuncian a continuación.

- Tiene la capacidad de volver a definir conjuntos de características que son aprendidas durante el aprendizaje.
- Realiza un entrenamiento notablemente rápido haciendo uso del algoritmo *quasi-Newton* LBFGS (*Broyden-Fletcher-Goldfarb-Shanno*), preparado para el procesamiento de largas escalas numéricas.
- Hace un uso moderado de la memoria durante el proceso de entrenamiento y durante las pruebas.
- Es capaz de mostrar una salida con los n mejores resultados obtenidos durante la realización de las pruebas.
- Puede entrenar los parámetros a partir del algoritmo MIRA (*Margin-infused relaxed algorithm*).
- Puede mostrar en la salida las probabilidades marginales para todos los candidatos.

4.4 Segmentación del *corpus* de entrenamiento

4.4.1 Motivación

Tal y como se ha explicado en la sección 4.2, para llevar a cabo la aproximación *train-on-target* para el problema de la comprensión del habla se necesita la traducción de un *corpus* así como sus frases segmentadas y sus correspondientes etiquetas semánticas.

Anteriormente se ha mencionado que la estrategia seguida para este proyecto ha sido la de separar los segmentos de las frases a la hora de traducir. El objetivo de este procedimiento es el de poder recuperar posteriormente los segmentos de la frase que se corresponden con cada concepto. Cada frase está formada por un conjunto de palabras $w_1 w_2 \ \ldots \ w_N$ y cada conjunto de una o más palabras pertenecen a un segmento etiquetado de la frase $s_1 : w_1 w_2 \ \ldots \ w_i$, $s_2 : w_{i+1} w_{i+2} \ \ldots \ w_m$, $\ldots$, $s_n : w_k w_{k+1} \ \ldots \ w_n$.

4.4.2 Extracción de las etiquetas semánticas

El objetivo principal de la obtención de las etiquetas semánticas es el de conseguir una lista con todas ellas para poder recuperarlas después del proceso de segmentación. De esta forma, cuando se haya conseguido alinear los segmentos con las frases se hará uso de la lista de segmentos para poder asignar cada etiqueta semántica al segmento que le corresponde.

Dado que los ficheros de traducción se han preparado siguiendo el mismo orden en el que aparecen las frases en el *corpus* `DIHANA`, se ha extraído cada una de las etiquetas semánticas asociadas a los segmentos detectados en el *corpus* en dicho orden.

La forma de conseguir las etiquetas semánticas ha sido la que se muestra en las siguientes instrucciones.

```
$ cat dihana_corpus.log | \
  grep "[A-Z]\?[a-z]\+.*:\(.*\)" | \
  sed "s/.*:\(.*\)/\1/g"
```

Código 4.1: Instrucciones para extraer las etiquetas semánticas

De esta forma, en primer lugar se separan los segmentos (que son el conjunto de palabras con una etiqueta semántica) de las frases que los contienen, y a continuación se extraen las etiquetas semánticas asociadas a dichos segmentos. Las etiquetas que se obtienen de esta forma se imprimen por la salida estándar y son desviadas a un fichero que actúa de contenedor de las mismas. Se obtienen un total de 14.575 etiquetas semánticas.

A la hora de traducir las frases y los segmentos siguiendo la organización que se ha explicado en la sección 3.3, se obtienen las traducciones de las frases seguidas de las de sus segmentos (cuyas etiquetas semánticas han sido ya guardadas). El principal problema de seguir esta estrategia es que la concatenación de los segmentos traducidos es potencialmente diferente a la traducción de la frase. Esto es debido a la falta de contexto de los segmentos sobre todo en los casos en que se componen por pocas palabras.

4.4.3 Levenshtein

El objetivo de la segmentación es el de localizar los segmentos en la frase a fin de poder realizar el correcto etiquetado de la misma. Para ello se ha implementado una modificación del algoritmo de `Levenshtein`.

La distancia de `Levenshtein` es una métrica capaz de medir la distancia de edición mínima entre dos secuencias de símbolos. Se puede explicar como la cantidad

de operaciones de inserción, sustitución y borrado que hay que realizar sobre una cadena a para transformarla en otra cadena b.

En la Ecuación 4.3 se puede ver una descripción matemática de la misma y en la tabla 4.1 se puede observar un ejemplo de edición.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{si } min(i,j) = 0\ , \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+(a_i \neq b_j) \end{cases} & \text{en otro caso.} \end{cases} \tag{4.3}$$

		d	o		n	o	t		f	r	o	m		C	i	u	d	a	d	_	R	e	a	l
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
n	1	1	2	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
o	2	2	1	2	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
t	3	3	2	2	3	4	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	4	4	3	2	3	4	4	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
f	5	5	4	3	3	4	5	4	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
r	6	6	5	4	4	4	5	5	4	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
o	7	7	6	5	5	4	5	6	5	4	3	4	5	6	7	8	9	10	11	12	13	14	15	16
m	8	8	7	6	6	5	5	6	6	5	4	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	9	8	7	7	6	6	5	6	6	5	4	3	4	5	6	7	8	9	10	11	12	13	14
C	10	10	9	8	8	7	7	6	6	7	6	5	4	3	4	5	6	7	8	9	10	11	12	13
i	11	11	10	9	9	8	8	7	7	7	7	6	5	4	3	4	5	6	7	8	9	10	11	12
u	12	12	11	10	10	9	9	8	8	8	8	7	6	5	4	3	4	5	6	7	8	9	10	11
d	13	12	12	11	11	10	10	9	9	9	9	8	7	6	5	4	3	4	5	6	7	8	9	10
a	14	13	13	12	12	11	11	10	10	10	10	9	8	7	6	5	4	3	4	5	6	7	8	9
d	15	14	14	13	13	12	12	11	11	11	11	10	9	8	7	6	5	4	3	4	5	6	7	8
_	16	15	15	14	14	13	13	12	12	12	12	11	10	9	8	7	6	5	4	3	4	5	6	7
R	17	16	16	15	15	14	14	13	13	13	13	12	11	10	9	8	7	6	5	4	3	4	5	6
e	18	17	17	16	16	15	15	14	14	14	14	13	12	11	10	9	8	7	6	5	4	3	4	5
a	19	18	18	17	17	16	16	15	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	4
l	20	19	19	18	18	17	17	16	16	16	16	15	14	13	12	11	10	9	8	7	6	5	4	3

Tabla 4.1: Ejemplo de frase resultado

La distancia de `Levenshtein` se suele utilizar para calcular la distancia entre dos cadenas a nivel de carácter. Es el ejemplo que podemos ver en la figura que se acaba de mostrar, donde se calcula la distancia de edición mínima para transformar la frase "*do not from Ciudad_Real*" en la frase "*not from Ciudad_Real*", obteniendo como resultado final 3.

En este caso las sustituciones se representan diagonalmente, mientras que las inserciones lo hacen moviendo una casilla hacia abajo y los borrados se realizan moviendo una casilla a la derecha. Cada vez que se realiza una inserción o un borrado la distancia de edición aumenta en uno, mientras que si es una sustitución la distancia aumenta en uno si los caracteres son diferentes y en cero en caso contrario.

4.4.4 Implementación realizada

El objetivo es poder localizar los segmentos traducidos en la frase traducida. Para lograrlo, el primer paso es formar una frase a partir de la concatenación de los segmentos. Una vez conseguido, se calcula la distancia de edición mínima entre la concatenación de los segmentos y la frase completa. A medida que se han realizado los cálculos se ha guardado también el camino de edición mínima que se ha ido obteniendo.

En la Ecuación 4.4 se representa la concatenación de los segmentos como a, mientras que la frase completa está representada por b. En este caso, la comparación entre ambas frases no se realiza a nivel de carácter sino a nivel de palabra. De esta forma, w_i representa la palabra que ocupa la posición i en la concatenación de los segmentos y w_j representa la palabra en la posición j de la frase completa.

$$lev_{a,b}(w_i, w_j) = \begin{cases} max(w_i, w_j) & \text{si } min(i,j) = 0 \ , \\ min \begin{cases} lev_{a,b}(w_{i-1}, w_j) + 1 \\ lev_{a,b}(w_i, w_{j-1}) + 1 \\ lev_{a,b}(w_{i-1}, w_{j-1}) + (w_i \neq w_j) \end{cases} & \text{en otro caso.} \end{cases} \tag{4.4}$$

Se ha inicializado también una variable cuyo valor es la frase donde cada palabra que marca el inicio de un segmento empieza con una barra vertical (|). De esta forma se pueden recuperar en el momento que sea necesario los límites entre segmentos.

Con la matriz de costes y del camino de edición mínima ya conseguidas, el siguiente paso ha sido el de deshacer el camino de edición mínima para obtener los segmentos en la frase completa. Antes de realizar este proceso se inicializa una lista vacía donde se irán añadiendo las palabras resultado de la segmentación. Para conseguir esto se han de tener en cuenta tres casos diferentes.

1. Sustitución: si la operación realizada en el punto que se está tratando es una sustitución, se añade directamente la palabra correspondiente de la frase

completa. Esto significa que la palabra del segmento que se está comparando se ha sustituido por la de la frase.

2. Inserción: en este caso, se añade la palabra de la frase completa igual que en el caso anterior.

3. Borrado: por último, si se ha realizado un borrado en la frase completa no se añade nada a la lista de palabras.

Cada vez que se pasa por un punto del recorrido de distancia de edición mínima se comprueba si la palabra de la concatenación de los segmentos que se está tratando es la que inicia un segmento. Esto se sabe comprobando si esta palabra empieza con la barra separadora en la variable que marca los límites entre segmentos y que se ha explicado previamente. En la tabla 4.2 se puede ver un ejemplo de la matriz resultante del proceso explicado además de la segmentación que se ha realizado.

		Segmentos							
			i	wanna	go	to	Segovia	from	Valencia
		0	1	2	3	4	5	6	7
Frase completa	i	1	0	1	2	3	4	5	6
	want	2	1	1	2	3	4	5	6
	to	3	2	2	2	2	3	4	4
	go	4	3	3	2	3	3	4	5
	to	5	4	4	3	2	3	4	5
	Segovia	6	5	5	4	3	2	3	4
	from	7	6	6	5	4	3	2	3
	Valencia	8	7	7	6	5	4	3	2

Tabla 4.2: Ejemplo de resultado de la segmentación

En este ejemplo se puede observar la entrada al algoritmo de la frase "*i want to go to Segovia from Valencia*" y de los segmentos *"i wanna go", "to Segovia", "from Valencia".*

Como se puede ver, los pasos que ha seguido el algoritmo para obtener la matriz resultado del cálculo de la distancia de edición mínima ha sido el que se muestra a continuación.

1. La "*i*" se ha sustituido por la "*i*" con coste cero ($\searrow$).
2. La palabra "*want*" se ha insertado con coste uno ($\downarrow$).
3. Se sustituye "*to*" por "*wanna*" con coste uno ($\searrow$).
4. Se sustituye "*go*" por "*go*" con coste cero ($\searrow$).
5. Se sustituye "*to*" por "*to*" con coste cero ($\searrow$).
6. Se sustituye "*Segovia*" por "*Segovia*" con coste cero ($\searrow$).
7. Se sustituye "*from*" por "*from*" con coste cero ($\searrow$).
8. Se sustituye "*Valencia*" por "*Valencia*" con coste cero ($\searrow$).

Al completar la matriz de costes y con ello la del camino seguido, se deshace el camino y se segmenta la frase de la siguiente manera.

- "*i wanna go*" $\rightarrow$ "*i want to go*".
- "*to Segovia*" $\rightarrow$ "*to Segovia*".
- "*from Valencia*" $\rightarrow$ "*from Valencia*".

De esta forma ya se tiene la frase segmentada y preparada para ser tratada por un *script* que la transforma en un formato compatible con el *toolkit* `CRF++`. Este proceso se repite para cada una de las frases del conjunto de entrenamiento traducido a inglés y francés por cada traductor *online*.

La implementación del algoritmo es accesible desde la dirección que se muestra a continuación.

`www.github.com/carmilso/TFG/blob/master/scripts/computeSegments.py`

En la tabla 4.3 se plasman la cantidad de frases, segmentos, vocabulario y número medio de palabras por segmento que hay para cada conjunto de entrenamiento traducido a inglés y francés después del proceso de segmentación. Además, se realiza una comparativa con el conjunto original de entrenamiento en castellano.

Inglés				
Traductor	Frases	Segmentos	Vocabulario	N. medio p. x s.
Apertium	4.887	14.521	590	2,85
Bing	4.887	13.892	771	2,61
Google	4.887	13.882	744	2,52
Lucy	4.887	14.457	660	2,83
Systranet	4.887	14.193	629	2,86
Francés				
Traductor	Frases	Segmentos	Vocabulario	N. medio p. x s.
Apertium	4.887	14.364	790	2,6
Bing	4.887	14.107	1.146	2,43
Google	4.887	14.002	1.029	2,39
Lucy	4.887	14.502	795	2,60
Systranet	4.887	14.517	819	2,56
Castellano				
	Frases	Segmentos	Vocabulario	N. medio p. x s.
	4.887	14.575	716	2,53

Tabla 4.3: Características de la segmentación

Como se puede observar en la tabla que se acaba de mostrar, el número de segmentos etiquetados en los conjuntos de entrenamiento en inglés y francés es diferente al de la referencia en castellano. Esto se debe a varias razones: hay palabras que no son entendidas por los traductores *online* y por tanto no se traducen correctamente; algunas frases son reordenadas de activo a pasivo; y se añaden algunas expresiones propias del lenguaje. Por estas razones hay segmentos que no se pueden alinear con la frase completa en el proceso de segmentación y se pierden.

En la figura 4.1 se puede ver un ejemplo de pérdida de segmentos con la frase de entrenamiento en castellano "quiero un billete de ida por la mañana el día dieciséis de junio en ave si puede ser no fumadores clase no preferente".

```
          quiero un billete:   consulta        |    i want a ticket:   consulta
                     de ida:   tipo_viaje      |            --------   --------
            por la mañana:     hora            |     in the morning:   hora
el día dieciséis de junio:     fecha           |  on june sixteenth:   fecha
                    en ave:    tipo_tren       |             in ave:   tipo_tren
              si puede ser:    consulta        |  if it can not be:    consulta
              no fumadores:    nada            |       non-smoking:    nada
       clase no preferente:    clase_billete   |     business class:   clase_billete
```

Figura 4.1: Ejemplo de pérdida de segmentos

Se puede ver como se ha perdido el segmento "de ida". Este segmento se ha traducido al inglés como "*outward*" y en el proceso de segmentación ha sufrido un borrado al no poder alinearse con ninguna secuencia de palabras de la frase completa.

4.4.5 Tratamiento de la salida

La salida que genera el *script* para la segmentación del ejemplo de la tabla 4.2 es "*i want to go | to Segovia | from Valencia*". En el caso del ejemplo de la pérdida de segmentos la salida es "*i want a ticket | | in the morning | on june sixteenth | in ave | if it can not be | non-smoking | business class*", donde se puede observar que hay un segmento vacío por no haberse podido segmentar.

Esta información es recogida por otro *script* que se encarga de asignar las etiquetas semánticas extraídas anteriormente a cada segmento calculado por el algoritmo que acaba de ser explicado. De esta forma se generan los ficheros de entrenamiento para los CRFs con cada traductor e idioma y otro con las traducciones obtenidas del conjunto de todos los traductores.

En las etiquetas semánticas se ha utilizado la notación IOB2. Por ello, aquellas palabras cuya etiqueta asociada comienza con una B (*Begin*) son las que marcan el inicio del concepto. Las que por el contrario tienen asociada una etiqueta que empieza con una I (*Inner*) denotan continuidad del concepto. Un ejemplo de los ficheros resultantes de entrenamiento con la frase que se ha utilizado para mostrar el proceso de segmentación se puede ver en la figura 4.2.

```
i               B_consulta
want            I_consulta
to              I_consulta
go              I_consulta
to              B_ciudad_destino
Segovia         I_ciudad_destino
from            B_ciudad_origen
Valencia        B_ciudad_origen
.               0
```

Figura 4.2: Ejemplo de frase de entrenamiento etiquetada para el *toolkit* `CRF++`

En el caso de los segmentos que no se consiguen alinear con la frase, se pierde también su correspondiente etiqueta semántica para el proceso de entrenamiento. Por ejemplo en el caso visto en la figura 4.1 no se entrenará ningún segmento de la frase con la etiqueta "tipo_viaje" a pesar de existir en la frase original en castellano.

El *script* que realiza la preparación de las frases segmentadas para ser compatible con el *toolkit* `CRF++` está disponible en el enlace siguiente.

`www.github.com/carmilso/TFG/blob/master/scripts/generateCRFTrain.sh`

Capítulo 5

Experimentación

5.1 Conjuntos de entrenamiento y prueba

El conjunto de entrenamiento consiste en un subconjunto del *corpus* `DIHANA` compuesto por 4.887 turnos y etiquetado y segmentado como se ha explicado en el Capítulo 2. Se han estimado dos sistemas de comprensión, uno para el inglés y otro para el francés. En ambos casos se ha definido un conjunto de prueba de 1.000 turnos de usuario. En la experimentación se usan entradas transcritas y de voz. Estas últimas han sido adquiridas por locutores nativos y reconocidas por el `Google ASR (Automatic Speech Recognizer)`. En el caso del inglés se ha adquirido las 1.000 frases de pruebas pero en el caso del francés solo se dispone de 500 frases adquiridas por nativos del total de frases del conjunto de pruebas transcrito.

El WER (*Word Error Rate*), cuya fórmula se muestra en la Ecuación 5.1 y que calcula la distancia de `Levenshtein` a nivel de palabra entre una frase y otra, calculado sobre las frases reconocidas con el `Google ASR` ha sido de 20,0 para el inglés y 19,5 para el francés.

$$WER = \frac{S + B + I}{N} \tag{5.1}$$

donde:

- S representa el número de palabras sustituidas
- B representa el número de palabras borradas
- I representa el número de palabras insertadas
- N es el número de palabras de la referencia

Para entrenar los modelos se han usado CRFs teniendo en cuenta dos palabras por delante y dos por detrás y se ha generado un modelo por cada idioma y por cada traductor *online* utilizado. Además se ha entrenado un modelo para cada idioma en el que se ha utilizado el conjunto de todas las traducciones obtenidas por los traductores *online*. De esta forma, los modelos se han entrenado con traducciones en el idioma objetivo, que coincide con el mismo en el que estará la prueba de entrada al sistema de comprensión. En la figura 5.1 se puede observar el sistema general del proceso de comprensión.

Cabe destacar que a las pruebas tanto de texto como de voz se les ha aplicado parte del postproceso explicado en la sección 3.4. Esto se debe a que ambos contienen signos de puntuación y letras en mayúsculas, además de números en el caso de las pruebas de voz.

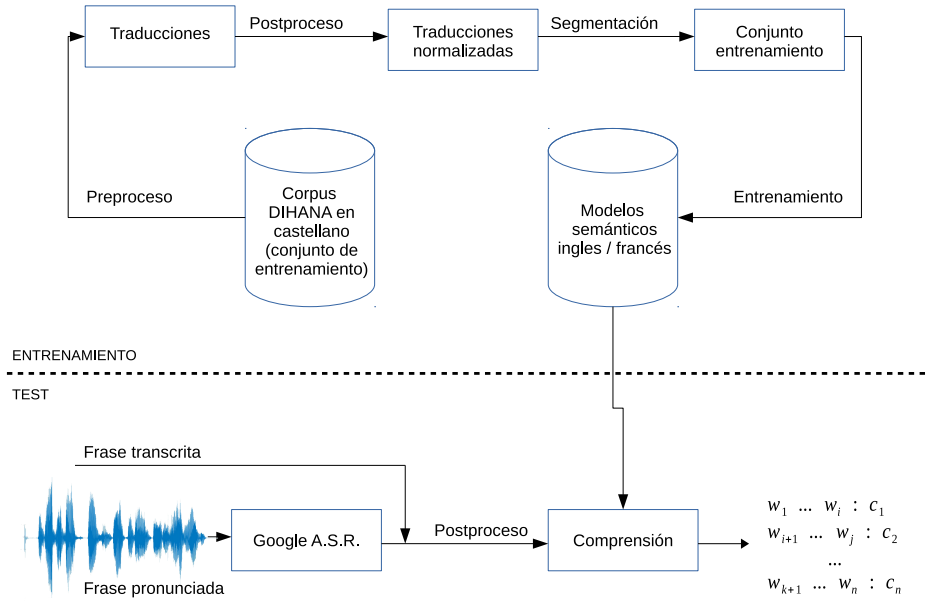

Figura 5.1: Esquema del sistema de comprensión usando *train-on-target*

Una vez las pruebas están normalizadas, se obtienen los resultados en los que se muestran los conceptos asociados a cada palabra de la frase tal y como se puede ver en la figura 5.2.

Los resultados se obtienen en dos columnas, de las cuales la primera es una palabra de la frase y la segunda es el concepto asociado a la palabra resultado de la com-

prensión. Los conceptos con los que se etiquetan las palabras de las frases de las pruebas siguen la notación IOB2, explicada anteriormente en la subsección 4.4.5. Cada frase del resultado se separa de la siguiente por un punto en la primera columna y un 0 en la última.

```
yes         B_<afirmacion>
i'd         B_<hora>
like        I_<hora>
times       I_<hora>
for         I_<hora>
trains      I_<hora>
going       I_<hora>
to          B_ciudad_destino
Cuenca      I_ciudad_destino
.           0
```

Figura 5.2: Ejemplo de frase resultado

5.2 Resultados de la experimentación

Para la experimentación se han utilizado cinco traductores *online* (`Apertium`, `Bing`, `Google`, `Lucy`, `Systranet`). Se han llevado a cabo los experimentos usando diferentes combinaciones de los traductores, que se codifican en las tablas de forma binaria (1 - se ha utilizado; 0 - no se ha utilizado) y se ordenan alfabéticamente, ocupando de esta manera `Apertium` el bit más significativo y `Systranet` el bit menos significativo.

El CER (*Concept Error Rate*), cuya fórmula se puede observar en la Ecuación 5.2 ha sido la medida utilizada para evaluar la calidad del sistema, que calcula la distancia de edición mínima entre la hipótesis (los conceptos asignados a cada frase por el sistema de comprensión) y la referencia (los conceptos reales de la frase) normalizado por la longitud de la referencia.

$$CER = \frac{S + B + I}{N} \tag{5.2}$$

donde:

- S representa el número de conceptos sustituidos
- B representa el número de conceptos borrados
- I representa el número de conceptos insertados
- N es el número de conceptos de la referencia

Además, con el fin de valorar la confianza de los resultados se ha calculado un intervalo de confianza (IC) del 95 % para cada uno de los resultados empleando la fórmula disponible en la Ecuación 5.3, donde la medida de la cual debe ser estimado su intervalo es el CER, que está representada por el parámetro p, y donde la suma de longitudes de las secuencias de conceptos de referencia se representa por el parámetro n.

$$IC = \pm 1,96 \cdot \sqrt{\frac{(p \cdot (1-p))}{n}} \tag{5.3}$$

En la tabla 5.1 se pueden ver los resultados obtenidos del inglés para las pruebas de 1.000 frases de texto y voz utilizando los modelos de cada traductor de forma individual y el de todos juntos.

	Texto		Voz	
Traductores	CER	IC±	CER	IC±
Apertium (10000)	30,5	1,66	37,6	1,75
Bing (01000)	24,8	1,56	30,4	1,66
Google (00100)	**23,0**	1,52	**28,1**	1,62
Lucy (00010)	34,8	1,72	40,4	1,77
Systranet (00001)	31,1	1,67	38,6	1,75
Todos (11111)	**21,9**	1,49	**27,0**	1,60

Tabla 5.1: Resultados individuales y totales de inglés con 1.000 frases

Como se puede observar, el mejor resultado tanto para las pruebas de texto como para las de voz en inglés se ha obtenido con el conjunto de todos los traductores. Este modelo semántico se ha adquirido a partir del entrenamiento de las traducciones normalizadas de todos los traductores. Se puede apreciar también que el modelo semántico obtenido a partir de las traducciones de `Google` es el que mejor resultado ha obtenido después de la combinación de todos los traductores, siendo especialmente notable en el caso de las pruebas de voz. Cabe destacar por tanto que éste ha sido el traductor que más ha aportado positivamente al modelo semántico obtenido de la combinación de todos los traductores

Por el contrario, se puede comprobar que `Lucy` es el traductor a partir del cual se ha generado el modelo semántico que peores aportaciones ha hecho al proceso de comprensión.

Con el objetivo de poder comparar las 500 frases de voz de francés con las de texto se ha generado una prueba de 500 frases de texto en francés producto del

subconjunto de las 1.000 de la prueba original. En la tabla 5.2 se pueden ver los resultados de las 500 frases de texto y voz en francés.

	Texto		Voz	
Traductores	CER	IC±	CER	IC±
Apertium (10000)	37,7	2,45	38,8	2,47
Bing (01000)	**22,8**	2,12	**25,9**	2,22
Google (00100)	**22,9**	2,13	**25,1**	2,19
Lucy (00010)	29,9	2,32	32,4	2,37
Systranet (00001)	27,6	2,26	31,7	2,35
Todos (11111)	**18,9**	1,98	**22,2**	2,10

Tabla 5.2: Resultados individuales y totales de francés con 500 frases

En la tabla que se acaba de mostrar se demuestra como el modelo semántico generado a partir de la combinación de todos los traductores vuelve a ser el que mejor resultados arroja para los dos tipos de prueba. En este caso tanto `Bing` como `Google` demuestran ser los traductores con los que mejor resultado se obtiene en las pruebas de francés.

Cabe señalar que en el caso del francés, el intervalo de confianza aumenta considerablemente con respecto al inglés. Esta situación provoca que los resultados obtenidos sean más relativos y la diferencia entre cada uno de los traductores con resultados aproximados no sea tan destacable como en el caso del inglés.

Se han calculado también los resultados para las pruebas de texto en francés con 1.000 frases que se pueden ver en la tabla 5.3.

	Texto	
Traductores	CER	IC±
Apertium (10000)	35,9	1,73
Bing (01000)	24,5	1,55
Google (00100)	**21,7**	1,49
Lucy (00010)	27,4	1,61
Systranet (00001)	26,9	1,60
Todos (11111)	**19,8**	1,43

Tabla 5.3: Resultados de texto en francés con 1.000 frases

En los resultados obtenidos para francés con 1.000 frases usando el modelo semántico de todos los traductores aumenta en nueve décimas el error con respecto a las pruebas con 500 frases. Sin embargo si nos fijamos en el intervalo de confianza se puede concluir que los resultados son similares a los obtenidos con 1.000 frases.

Las tablas que se muestran a continuación siguen la codificación binaria explicada anteriormente, con lo que por ejemplo la secuencia 00011 representa la combinación de traductores `Lucy` y `Systranet`, mientras que la secuencia 11100 representa la combinación de `Apertium`, `Bing` y `Google`.

Se ha realizado una experimentación que recoge todas las posibles combinaciones de uso de los cinco traductores *online* para la obtención del conjunto de entrenamiento en inglés y francés.

En la tabla 5.4 se pueden observar los resultados de las diferentes combinaciones realizadas para las pruebas tanto de texto como de voz con 1.000 frases cada una para el inglés.

	Texto		Voz	
Traductores	CER	IC±	CER	IC±
00011	25,7	1,57	33,3	1,70
00101	22,0	1,49	27,7	1,61
00110	22,2	1,50	27,8	1,61
01001	24,7	1,55	30,6	1,66
01010	24,1	1,54	30,0	1,65
01100	22,5	1,50	27,6	1,61
10001	26,4	1,59	33,6	1,70
10010	26,5	1,59	33,7	1,70
10100	21,9	1,49	27,6	1,61
11000	24,0	1,54	30,2	1,66
00111	22,0	1,49	27,7	1,61
01011	24,0	1,54	30,3	1,66
01101	22,2	1,50	27,6	1,61
01110	22,3	1,50	27,3	1,61
10011	25,3	1,57	32,7	1,69
10101	**21,5**	1,48	**27,1**	1,60
10110	**21,6**	1,48	**27,0**	1,60
11001	24,1	1,54	30,3	1,66
11010	22,9	1,51	29,5	1,64
11100	22,5	1,50	27,9	1,62
01111	22,2	1,50	**27,2**	1,60
10111	**21,5**	1,48	**27,1**	1,60
11011	23,5	1,53	30,1	1,65
11101	22,4	1,50	27,8	1,62
11110	22,4	1,50	27,4	1,61

Tabla 5.4: Combinación de traductores para inglés con 1.000 frases

En la tabla que se acaba de mostrar se puede ver como ha habido varias combinaciones de traductores en las que el resultado ha sido similar. Estas combinaciones

han sido las siguientes: `Apertium`, `Google` y `Systranet` (10101); `Apertium`, `Google` y `Lucy` (10110); `Apertium`, `Google`, `Lucy` y `Systranet`. Además en el caso de las pruebas de voz hay que destacar también la combinación de `Bing`, `Google`, `Lucy` y `Systranet` (0111).

En la tabla 5.1 se puede ver que la combinación de todos los traductores obtiene un CER de 21,9 en las pruebas de texto y un 27,0 en las de voz. Debido al intervalo de confianza se puede concluir que la mejor elección a la hora de entrenar un modelo semántico para inglés es utilizar todas las traducciones, lo cual evita tener que elegir entre una combinación u otra.

Siguiendo el mismo procedimiento que se ha realizado para calcular los resultados de la tabla 5.4, se muestran en la tabla 5.5 los resultados de diferentes combinaciones realizadas para las pruebas de texto y voz con 500 frases de francés.

	Texto		Voz	
Traductores	CER	IC±	CER	IC±
00011	24,3	2,17	26,0	2,22
00101	21,2	2,07	**22,0**	2,10
00110	21,6	2,08	23,1	2,13
01001	20,3	2,04	23,9	2,16
01010	21,4	2,08	23,5	2,14
01100	**19,9**	2,02	24,4	2,17
10001	25,0	2,19	27,6	2,26
10010	27,2	2,25	29,3	2,30
10100	22,2	2,10	23,3	2,14
11000	21,7	2,08	24,7	2,18
00111	20,9	2,06	**21,7**	2,08
01011	20,3	2,04	23,4	2,14
01101	**19,1**	1,99	23,1	2,13
01110	**19,3**	2,00	23,1	2,13
10011	24,8	2,19	26,2	2,23
10101	21,1	2,07	**22,1**	2,10
10110	21,5	2,08	23,0	2,13
11001	20,0	2,02	23,7	2,15
11010	21,2	2,07	23,9	2,16
11100	**19,1**	1,99	23,4	2,14
01111	**18,9**	1,98	**22,3**	2,11
10111	21,3	2,07	**22,0**	2,10
11011	20,8	2,05	23,5	2,15
11101	**19,1**	1,99	**22,7**	2,12
11110	**19,1**	1,99	**22,8**	2,12

Tabla 5.5: Combinación de traductores para francés con 500 frases

Como se muestra en la tabla anterior para las pruebas de texto en francés, la combinación de `Bing`, `Google`, `Lucy` y `Systranet` (0111) obtiene el mismo resultado que con la combinación de todos los traductores, que es de 19,8 tal y como se puede observar en la tabla 5.2. Además, en las pruebas de voz con la combinación de los traductores `Google` y `Systranet` (00101), y con la de `Bing`, `Google`, `Lucy` y `Systranet` (0111), se obtiene mejor resultado que con la combinación de todos ellos. Igual que en el caso del inglés y debido al intervalo de confianza, se puede asegurar que la mejor opción es usar la combinación de todos los traductores.

Por último, se muestra en la tabla 5.6 los resultados para las pruebas de texto en francés con un total de 1.000 frases.

	Texto	
Traductores	CER	IC±
00011	22,6	1,51
00101	20,1	1,45
00110	20,3	1,45
01001	21,5	1,48
01010	21,9	1,49
01100	21,1	1,47
10001	23,3	1,52
10010	24,8	1,56
10100	20,6	1,46
11000	22,2	1,50
00111	20,0	1,44
01011	21,1	1,47
01101	20,3	1,45
01110	20,2	1,45
10011	22,6	1,51
10101	**19,9**	1,44
10110	20,2	1,45
11001	20,6	1,46
11010	21,8	1,49
11100	**19,9**	1,44
01111	**19,8**	1,44
10111	20,1	1,45
11011	21,1	1,47
11101	**19,8**	1,44
11110	**19,8**	1,44

Tabla 5.6: Combinación de traductores para texto en francés con 1.000 frases

Ésta es la última tabla que se ha obtenido y en ella se puede ver como con una prueba de texto con 500 frases de prueba más de las que se han utilizado para la tabla 5.5 se empeoran ligeramente los resultados. Por ello tampoco se obtiene

ninguna combinación de traductores que funcione mejor que la combinación de todos ellos.

5.3 Valoración de los resultados

En el apartado anterior se han mostrado los resultados obtenidos por los sistemas de comprensión entrenados con CRFs a partir de diferentes combinaciones de traductores.

Como se ha explicado previamente, las combinaciones más interesantes de traductores son aquellas en las que todos participan. A pesar de haber combinaciones concretas que arrojan un resultado levemente más positivo, el intervalo de confianza indica que no son resultados que difieran significativamente de los que se obtienen con la combinación de todos los traductores.

Anteriormente en el Capítulo 4, se ha explicado que la aproximación que se ha seguido para construir el sistema de comprensión ha sido la del *train-on-target*. Por ello, todos los modelos de comprensión han sido entrenados a partir de traducciones normalizadas en el idioma destino.

Existe otra aproximación diferente que es la del *test-on-source* [5]. Esta técnica consiste en entrenar los modelos de comprensión en el idioma origen y traducir al idioma destino las pruebas de entrada al sistema. Un esquema básico de esta aproximación se puede ver en la figura 5.3.

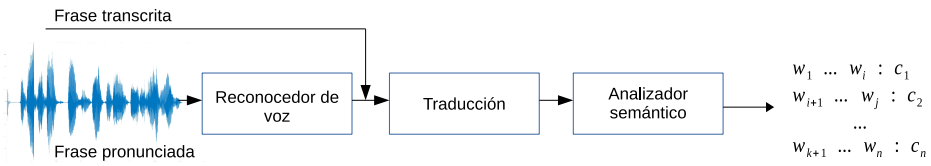

Figura 5.3: Esquema del sistema de comprensión usando *test-on-source*

Como se puede ver, la aproximación *test-on-source* centra su trabajo en la traducción de las frases de entrada al sistema. En el artículo citado se usan los mismos de conjuntos de prueba que en el presente proyecto. Además, se utilizaron diferentes aproximaciones para entrenar los modelos semánticos. Entre ellas se encuentra la misma que la que se ha utilizado en este proyecto para entrenar los modelos semánticos, es decir, los CRFs.

En la tabla 5.7 se pueden ver los resultados que se presentan en el artículo usando diferentes combinaciones de traductores y entrenando los modelos semánticos con CRFs. Concretamente se muestran las combinaciones de traductores que proporcionaron mejores resultados.

		test-on-source	*train-on-target*
		`CER`	`CER`
Inglés	`Texto`	22,1	21,9
	`Voz`	32,2	27,0
Francés	`Texto`	23,1	19,8
	`Voz`	28,9	22,2

Tabla 5.7: Comparación de resultados con CRFs usando las dos estrategias

Como se puede observar, los resultados obtenidos siguiendo la aproximación de *train-on-target* mejoran en todos los casos los obtenidos siguiendo la aproximación de *test-on-source*.

Mientras que en el artículo, siguiendo la aproximación *test-on-source*, se obtuvo una puntuación de 22,1 de CER para las pruebas de texto en inglés, con la aproximación utilizada en este proyecto se ha conseguido rebajar esta cifra hasta 21,9. En el caso de las pruebas de voz para este mismo idioma, en el artículo se muestra una puntuación de 32,2 de CER siendo de solo 27,0 la conseguida mediante la aproximación de *train-on-target*.

En el caso del francés también se han obtenido mejores resultados siguiendo la aproximación de *train-on-target*. En las pruebas de texto con 1.000 frases se obtuvieron en el artículo 23,1 puntos mientras que esa cifra se ve reducida hasta los 19,8 puntos en este proyecto. Cabe recordar que en las pruebas de voz para francés tanto en el artículo como en el presente proyecto se han utilizado tan solo 500 frases. En este último caso, la puntuación obtenida en el artículo fue de 28,9 de CER, que pasa a ser 22,2 siguiendo la aproximación de *train-on-target*.

Se puede apreciar que la diferencia de resultados siguiendo una aproximación u otra es significativa. A pesar de ser mejores los resultados, el proceso de obtención de los modelos semánticos en la aproximación de *train-on-target* es mucho más costosa que el tratamiento de la entrada al sistema de comprensión de la aproximación de *test-on-source*.

En la aproximación de *train-on-target* que se ha utilizado en el presente proyecto ha sido necesario traducir todas las frases y sus segmentos, para lo que es necesario una tarea previa de análisis de los traductores *online* a utilizar. A continuación se ha analizado y tratado cada una de las traducciones obtenidas de los diferentes traductores para hacerlas todas consistentes entre sí. Por último y previo al en-

trenamiento de los modelos semánticos se ha realizado un proceso de alineamiento de segmentos para poder etiquetar cada segmento de la frase con el concepto semántico que le corresponde. Todo este proceso está detalladamente explicado en los Capítulos 3 y 4.

Sin embargo, para la aproximación de *test-on-source* utilizada en el artículo no es necesario un proceso tan costoso. Esto se debe a que en la citada aproximación el entrenamiento de los modelos semánticos se realiza en el idioma destino, y como ya se ha explicado anteriormente es la entrada la que se encuentra en el idioma origen y la que debe ser tratada.

Por todo esto se puede concluir que la elección de una aproximación u otra dependerá de el nivel de error que se esté dispuesto a asumir, ya que como se acaba de explicar, la aproximación de *test-on-source* es mucho menos costosa que la de *train-on-target* aunque los resultados obtenidos sean unos puntos peores.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

En este proyecto se han tratado los problemas de la obtención de un sistema de comprensión del habla multilingüe. Tal y como se ha explicado al inicio del proyecto en el Capítulo 1, se parte de un sistema de comprensión en castellano y el objetivo es adquirir uno en inglés y francés.

Para conseguir el sistema de comprensión en inglés y francés se ha seguido la aproximación *train-on-target*. Lo primero que se ha conseguido es la obtención de un conjunto de entrenamiento en ambos idiomas, haciendo uso de cinco traductores *online* de propósito general (`Apertium`, `Bing`, `Google`, `Lucy` y `Systranet`), a partir de un subconjunto del *corpus* original en castellano. Estos conjuntos de entrenamiento han sido normalizados para eliminar las particularidades de cada traductor *online* tal y como se ha explicado en el Capítulo 3. Con los conjuntos de entrenamiento ya normalizados, se ha realizado un proceso de segmentación para alinear los segmentos con las frases enteras como se muestra en el Capítulo 4. De esta forma se obtiene la frase completa segmentada y preparada para el entrenamiento mediante CRFs. A partir de los conjuntos de entrenamiento segmentados se entrenan los modelos semánticos. Se ha obtenido un modelo por cada idioma y traductor *online* utilizado, pero además se han entrenado modelos con la combinación de las traducciones de diferentes traductores.

Con los modelos semánticos en inglés y francés entrenados se ha realizado pruebas por cada idioma con todas las combinaciones de modelos entrenados. Los resultados han sido comparados con los presentados siguiendo la aproximación *test-on-source*. Se ha podido demostrar en base a estos resultados que siguiendo la

aproximación *train-on-target* y utilizando diferentes combinaciones de traductores se obtienen resultados que mejoran ligeramente los obtenidos por la aproximación *test-on-source*. A pesar de esta mejora, se puede ver como la aproximación *train-on-target* conlleva mucho más trabajo y tiempo que la aproximación *test-on-source*.

Como se ha podido ver a lo largo del proyecto, es difícil evitar la acumulación de errores en el proceso de obtención del sistema de comprensión multilingüe. Los traductores *online* cometen errores semánticos que no pueden ser tratados en la normalización por la naturaleza de los mismos. Estos errores se propagan al proceso de segmentación donde tal y como se ha mostrado en la subsección 4.4.4 se pierden algunos de los segmentos. Esta acumulación de errores provoca que el entrenamiento de los modelos semánticos tampoco quede exento de ellos.

6.2 Trabajo futuro

En vista de que los resultados obtenidos con el sistema de comprensión siguiendo la aproximación *train-on-target* son satisfactorios, conviene estudiar diferentes propuestas de optimización.

Se puede hacer uso de la herramienta MOSES para la generación de un traductor automático a partir de las traducciones obtenidas por todos los traductores *online* para inglés y francés. Esto permitiría entrenar un nuevo modelo de comprensión con el que obtener nuevos resultados que se podrían comparar con los obtenidos en este proyecto. De esta forma se podría comprobar si las traducciones mejoran significativamente respecto a las de los traductores *online*. Además, se puede usar los resultados obtenidos por diferentes combinaciones de traductores para entrenar un traductor automático a partir de las mejores combinaciones de los mismos.

Respecto al proceso de segmentación, se pueden probar otras aproximaciones diferentes a la implementada en este proyecto. Segmentar correctamente es fundamental a la hora de conseguir un conjunto de entrenamiento de calidad. Esto se podría conseguir considerando la posible permutación de los segmentos para alinearlos con la frase debido a la conversión de algunas frases en voz activa a voz pasiva y la introducción de frases hechas propias del idioma. Otra posibilidad sería la de obligar a respetar el número de etiquetas por frase, con lo que no se perdería ninguna etiqueta semántica.

6.3 Asignaturas relacionadas

En este proyecto se ha hecho uso del material y los conocimientos adquiridos en el transcurso del grado. Han resultado muy relevantes asignaturas como "Sistemas de almacenamiento y recuperación de la información" o "Algorítmica" relacionadas con el tratamiento del lenguaje y la programación eficiente. Otras asignaturas como "Percepción" o "Aprendizaje automático" han servido también para asentar las bases para la comprensión de las técnicas de aprendizaje y clasificación usadas en el proyecto como son los CRFs.

Bibliografía

[1] S. Seneff. "TINA. A probabilistic syntactic parser for speech understanding systems". En: *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. Mayo de 1989, 711-714 vol.2. DOI: 10.1109/ICASSP.1989.266526 (vid. pág. 3).

[2] F. Mairesse y col. "Cross-Lingual Spoken language understanding from unaligned data using discriminative classification models". En: IEEE, abr. de 2009, págs. 4749-4752 (vid. pág. 4).

[3] Gregoire Mesnil y col. "Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding". En: *IEEE/ACM transactions on audio, speech, and language processing, vol. 23, no. 3*. Mar. de 2015 (vid. pág. 5).

[4] John D. Lafferty, Andrew McCallum y Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". En: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, págs. 282-289 (vid. págs. 6, 34).

[5] Marcos Calvo y col. "Multilingual Spoken Language Understanding Using Graphs and Multiple Translations". En: *Comput. Speech Lang.* 38.C (jul. de 2016), págs. 86-103. ISSN: 0885-2308. DOI: 10.1016/j.csl.2016.01.002 (vid. págs. 7, 53).

[6] E. Segarra y col. "Extracting Semantic Information Through Automatic Learning Techniques". En: *IJPRAI* 16.3 (2002), págs. 301-307 (vid. pág. 7).

[7] Lucía Ortega y col. "A Statistical Segment-Based Approach for Spoken Language Understanding". En: *Proc. of InterSpeech 2010*. Makuhari, Chiba, Japan, 2010, págs. 1836-1839 (vid. pág. 7).

[8] J. M. Benedí y col. "Design and acquisition of a telephone spontaneous speech dialogue corpus in Spanish: DIHANA". En: *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*. 24-25-26 May 2006. European Language Resources Association (ELRA). Genoa (Italy), mayo de 2006, págs. 1636-1639 (vid. pág. 11).

[9] Aurelio Jargas. *The sed $HOME*. URL: `http://sed.sourceforge.net` (vid. pág. 18).

[10] Free Software Foundation. *Gawk*. URL: `https://www.gnu.org/software/gawk` (vid. pág. 18).

[11] Python. *Python Software Foundation*. URL: `https://www.python.org` (vid. pág. 18).

[12] Tiobe. *TIOBE Index for May*. 2016. URL: `http://www.tiobe.com/tiobe_index` (vid. pág. 18).

[13] F. García y col. "Combining multiple translation systems for Spoken Language Understanding portability". En: *Proc. of IEEE Workshop on Spoken Language Technology (SLT 2012)*. Miami, 2012, págs. 282-289 (vid. pág. 34).

[14] Taku Kudo. *CRF++: Yet Another CRF toolkit*. URL: `https://taku910.github.io/crfpp/` (vid. pág. 36).

Apéndice A

Palabras invariantes

A_Coruña
Alacant
alaris
Albacete
Alicante
Almería
altaria
alta_velocidad
ave
Ávila
Badajoz
Barcelona
Bilbao
Burgos
Cáceres
Cádiz
Cambrils
Cartagena
Castellón
cataluña_expres
Ciudad_Real
Compostela
Córdoba
Coruña
Cuenca
Cullera
estrella
euromed
francia
Galicia
Gerona
Gijón
Girona
Granada
Guadalajara
Huelva
Huesca
intercity
Jaén
La_Coruña
León
Lérida
Lleida
Logroño
Lugo
Madrid
Málaga
marenostrum
mare_nostrum
Mérida
miguel_de_unamuno
Monzón
Murcia
Orense
Oviedo
Palencia
Pamplona
París
Peñíscola
Pontevedra
port_aventura
Real
regional
Salamanca
san_fermines
san_josé
San_Sebastián
Santander
Santiago
Santiago_de_Compostela
Sebastián
Segovia
semana_santa
Sevilla
Soria
Sueca
talgo
talgos
Tarragona
Teruel
Toledo

tren_hotel
triana
Valencia
Valladolid
viernes_santo
Vigo
Vitoria
Zamora
Zaragoza