Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación

Un modelo estadístico para la gestión de sistemas de diálogo hablado

Máster en
Inteligencia Artificial, Reconocimientos de Formas e Imagen Digital

Febrero de 2011

Dirigido por:
Dr. Lluís F. Hurtado Oliver
Dr. Emilio Sanchis Arnal

Presentado por:
Joaquín Planells Lerma

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La capacidad de entablar una conversación o dialogar es una de las actividades fundamentales del ser humano. Cuando somos pequeños aprendemos nuestro primer idioma a través del diálogo y aún para la mayoría de adultos, dialogar es una de las actividades fundamentales en el día a día. Cuando pedimos la comida en un restaurante, llamamos por teléfono a un amigo o pedimos información sobre horarios de trenes estamos participando en un diálogo en el que dos o más interlocutores intercambian información en lenguaje natural. Utilizamos el diálogo para resolver un gran número de situaciones cotidianas y uno de los objetivos de la Ingeniería del Lenguaje Natural es el de conseguir que una máquina sea capaz de interactuar con personas de la misma forma. Idealmente sin que el usuario humano sepa que está hablando con un ordenador, aunque esto está fuera de nuestro alcance por el momento.

Entendemos por diálogo una situación en la que dos o más agentes intercambian frases[1] en lenguaje natural con el fin de alcanzar un objetivo concreto. En el presente texto nos interesa el caso en el que uno de estos agentes es una máquina que debe dar servicio a un usuario.

El desarrollo de sistemas de diálogo hablado (spoken dialog system, SDS) es uno de los principales objetivos de la tecnología del lenguaje hablado. Un SDS se puede ver como una interfaz hombre-máquina que reconoce y entiende comandos de voz y genera respuestas utilizando una voz sintética. Mediante sucesivos turnos de diálogo el usuario trata de alcanzar un objetivo como obtener información o realizar cierta acción. La mayoría de los sistemas de diálogo están pensados para ser utilizados telefónicamente y están adaptados para tareas con dominios restringidos.

Normalmente están formado por una serie de módulos que se ejecutan uno tras otro de manera secuencial: reconocimiento de voz, compresión del lenguaje natural, gestión del diálogo, generación de respuestas y síntesis de voz. En la siguiente sección veremos una breve explicación de cada uno de

[1] También llamadas turnos o actos de diálogo.

estos módulos.

Hoy en día podemos encontrar sistemas de diálogo en los teléfonos de atención al cliente de la mayoría de grandes empresas. Estos sistemas tratan de obtener información del usuario para poder dirigirle a el teleoperador apropiado o en algunos casos, resolver la incidencia o duda del usuario.

Durante los últimos años se ha tratado de incorporar técnicas estadísticas a cada una de estas subtareas que forman un sistema de diálogo hablado (Young, 2002). Este tipo de metodologías se han aplicado con éxito desde hace años en los campos de reconocimiento y comprensión del lenguaje natural (Griol et al., 2008; Williams and Young, 2007; Levin and Pieraccini, 1995; Minker, 1999; Segarra et al., 2002; He and Young, 2003; Esteve et al., 2003).

Este trabajo se encuadra dentro del proyecto *SD-TEAM: Sistemas basados en la interacción oral dinámicamente mejorables y adaptables a nuevos contextos* financiado por el Ministerio de Ciencia e Innovación (MICINN). Se presenta aquí una nueva aproximación para realizar la gestión de diálogo mediante un modelo estadístico y una plataforma para la generación automática de diálogos. Para la evaluación se ha utilizado un tarea diseñada en el seno de los proyectos *EDECAN* y *SD-TEAM* llamada *EDECAN-SPORT*. Esta tarea modela un sistema de diálogo hablado que permite reservar y anular pistas deportivas en un campus universitario.

1.1. Sistemas de diálogo

En la Figura 1.1 vemos un diagrama de los módulos que intervienen en un sistema de diálogo junto con las posibles fuentes de información que proveen a cada uno. La ejecución clásica de un sistema como el mostrado comenzaría cuando el usuario pronuncia una frase. Esta frase se convierte a texto utilizando el módulo de reconocimiento automático del habla y se envía al módulo de comprensión que transforma el lenguaje humano en una representación más manejable por la máquina. Esta representación, llamada lenguaje semántico intermedio se envía al gestor de diálogo que selecciona la siguiente acción del sistema teniendo en cuenta lo dicho por el usuario hasta el momento. Una vez la respuesta ha sido seleccionada el generador de respuestas se encarga de hacérsela llegar al usuario mediante los altavoces o la pantalla.

A continuación se explican brevemente las funciones de cada uno de los módulos mencionados. En general, todas las aplicaciones que utilizan sistemas de diálogo incorporan estos módulos aunque pueden presentar más fases o nombrarlos de distintas formas.

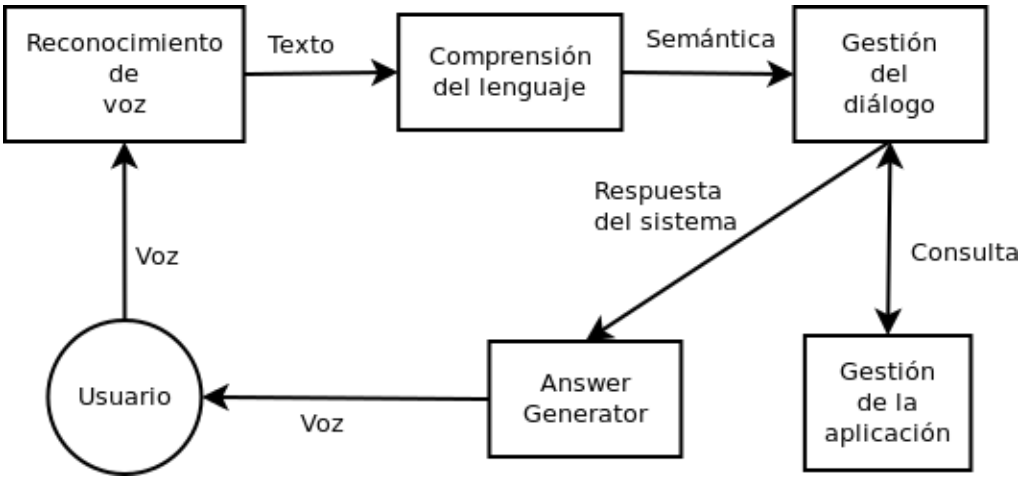

Figura 1.1: Diagrama de módulos de un sistema de diálogo.

1.1.1. Reconocimiento automático del habla

El reconocedor automático del habla es el módulo encargado de convertir la señal acústica que proviene del micrófono en texto. Un reconocedor debe ser capaz de discernir si la señal es habla humana (señal vocal) o por el contrario se trata de ruido. En caso de detectar señal vocal, el reconocedor debe especificar qué palabras han sido pronunciadas. Este proceso es muy complicado ya que cada persona pronuncia una misma palabra de manera distinta. Incluso una misma persona no siempre pronuncia igual una palabra si no que depende del estado de salud, de su ánimo o de la posición del cuello. Para resolver este problema se han diseñado técnicas estadísticas que permiten tratar esta variabilidad en las voces durante reconocimiento de las palabras (Rabiner et al., 1996).

El campo del reconocimiento del habla ha ido perfeccionándose a lo largo de los años. En un principio se trató de detectar palabras aisladas como *Sí*, *No* o los números del 0 al 9. Una vez se consideró resuelto este problema se trató de reconocer frases completas pero en dominios restringidos. Este tipo de reconocimiento presenta el problema añadido de que normalmente no separamos las palabras al hablar y que las sílabas de una palabra modifican ligeramente la pronunciación de las palabras adyacentes. Hoy en día se intenta reconocer cada vez mayores vocabularios y además hacerlo en entornos donde puede haber ruido (Aubert and Ney, 1995; Rose and Lleida, 1997).

En el Cuadro 1.1.1 se muestran las tasas de error para algunas tareas usadas habitualmente como benchmark. Los datos han sido extraídos de (Jurafsky, 2009) y representan el estado del arte del reconocimiento en el año 2006.

Tarea	Vocabulario	Tasa de error (%)
TI Digits	11 (zero-nine, oh)	.5
Wall Street Journal read speech	5,000	3
Wall Street Journal read speech	20,000	3
Broadcast News	64,000+	10
Conversational Telephone Speech (CTS)	64,000+	20

Cuadro 1.1: Tasas de error de reconocimiento en 2006 para algunas tareas.

Este módulo obtiene la señal de un micrófono y debe ser capaz de devolver una o más frases con lo que estima que ha dicho el usuario. Como ya hemos dicho, es una tarea compleja y no existe un reconocedor perfecto, por tanto a cada frase generada se le suele asignar una valor de confianza. Este valor numérico indica la seguridad que tiene el algoritmo de que el resultado que nos devuelve es el resultado real. En este trabajo se trata fundamentalmente sobre la gestión del diálogo y asumimos que disponemos de un reconocedor automático del habla con una cierta calidad que es capaz de darnos la frase que se ha pronunciado con alguna medida de confianza fiable.

1.1.2. Comprensión del habla

El módulo de comprensión del habla se encarga de traducir un texto o frase en lenguaje a natural a un lenguaje semántico definido para la tarea que representa el significado de la frase. Este lenguaje depende de la tarea a considerar y debe ser definido por un experto en la misma.

En muchas de las aplicaciones relacionadas con la interacción oral hombre máquina, el proceso de comprensión del habla tiene una gran relevancia. Tal es el caso de los sistemas de diálogo hablado, en que el módulo de comprensión debe extraer tanto la intención como la información proporcionada por el usuario. En los últimos años se han hecho grandes esfuerzos en el desarrollo de sistemas de diálogo, lo que ha impulsado también los trabajos en el área de comprensión de habla (De Mori et al., 2008).

Habitualmente estas aplicaciones se limitan a dominios restringidos dado que las capacidades de representación y modelización están muy condicionadas por la cantidad y variabilidad de campo semántico que se quiere modelizar. Ejemplo típicos de aplicaciones de diálogo y comprensión son los sistemas de acceso a un servicio de información considerando habla espontánea, iniciativa mixta y un vocabulario de talla media. Al igual que en otras modelizaciones en el campo del reconocimiento del habla, como son la modelización acústica o los modelos de lenguaje, los métodos estadísticos han sido ampliamente usados en comprensión del habla (Minker, 1999). Entre estas aproximaciones se encuentran aquellas basadas en clasificadores aprendidos con criterios discriminativos (Hahn et al., 2009; Dinarelli et al.,

2009), las basadas en modelos estocásticos (He and Young, 2006; Segarra et al., 2002), y las basadas en transductores (Raymond et al., 2006). La representación semántica escogida en la mayoría de las propuestas es la de frame, consistentes en un conjunto de conceptos y pares atributo-valor. Esta representación es especialmente útil cuando la aplicación consiste en un servicio de información que requiere el acceso a una base de datos y por tanto hay que completar una serie de campos para realizar un requerimiento (slot-filling). Sin embargo, en los últimos años se ha empezado a estudiar la posibilidad de trabajar con estructuras semánticas jerárquicas más complejas (Quarteroni et al., 2009; Pérez et al., 2006).

Al igual que el módulo de reconocimiento del habla, en este trabajo se da por supuesto que el problema de la comprensión del habla está resuelto. De todas las técnicas expuestas arriba, el formato utilizado aquí será el del frame. Cada frase que pronuncie el usuario será traducida a un frame que incluirá uno o más conceptos y sus atributos.

Por ejemplo, para la frase *Quería reservar una pista de baloncesto para mañana*, el módulo de comprensión semántica nos devolvería:

```
BOOKING
   SPORT: basketball
   DATE: tomorrow
```

Con este frame tenemos una representación semántica de la frase anterior sin tener en cuenta las palabras concretas. Frases como *Me gustaría reservar baloncesto mañana si es posible* o *¿Podría reservar baloncesto para mañana?* tienen la misma representación semántica para utilizan expresiones distintas.

Para modelar posibles errores de interpretación semántica del módulo de comprensión, cada concepto y atributo utilizado viene acompañado con una o más medidas de confianza.

1.1.3. Gestión del diálogo

El módulo de gestión del diálogo[2] se encarga de determinar qué acciones realizará el sistema. Estas acciones pueden consistir en ofrecer información al usuario o bien llevar a cabo algún trabajo o modificación del sistema. Por ejemplo, emitir por los altavoces el mensaje *Gracias por usar este servicio* sería la acción elegida por el gestor en el momento que el usuario ha terminado de interactuar con el sistema. Tradicionalmente, los gestores de diálogo se han diseñado manualmente, pero a medida que las aplicaciones que integran sistemas de diálogo se vuelven más ambiciosas el diseño manual se vuelve inviable. En los últimos años se han obtenido resultados interesantes diseñando los gestores de diálogo utilizando metodologías estadísticas (Levin

[2] A veces nos referimos a él como gestor de diálogo o simplemente gestor.

et al., 2000; Torres et al., 2003; Lemon et al., 2006; Hurtado et al., 2006; Williams and Young, 2007).

Estas aproximaciones consiguen resultados razonables en el laboratorio pero presentan problemas cuando se aplican en entornos más realistas. Tienen que lidiar con locuciones inesperadas de los usuarios, ser robustos frente a los errores de reconocimiento o comprensión, etc. Algunos de estos defectos se deben a la falta de muestras de aprendizaje con las que estimar los parámetros del modelo estadístico.

Durante un diálogo, el gestor no trabaja directamente con la frase pronunciada por el usuario si no que utiliza la representación semántica en formato frame generada por el módulo de comprensión. Este trabajo trata sobre la gestión del diálogo y se hablará en profundidad sobre el tema en capítulos posteriores.

1.1.4. Gestor de aplicación

El gestor de aplicación es el módulo que implementa la lógica de la tarea que se quiere resolver. Este módulo es absolutamente dependiente de la tarea y no suele ser posible utilizar técnicas estadísticas en este módulo ya que tiene una funcionalidad específica que debe definir un experto en la materia.

Por ejemplo, en un sistema de diálogo hablado para la obtención de información sobre trenes y rutas, el gestor de aplicación constaría de una base de datos con las estaciones, horarios y toda la información relativa a los viajes en tren. Además contaría con una serie de algoritmos para tratar temas como trasbordos.

Generalmente, el gestor de aplicación consta de una base de datos con toda la información referente a la tarea a resolver. Suele incluir además una serie de reglas para el control de algunos aspectos de la aplicación como cumplimiento de normativas, verificaciones de seguridad, etc.

El gestor de aplicación puede ser un módulo independiente o bien estar integrado dentro del módulo de gestión de diálogo.

1.1.5. Generación multimodal de respuestas

El generador de respuestas se encarga de hacer llegar al usuario información sobre las acciones del sistema. En un sistema de diálogo hablado el usuario recibe normalmente la información mediante voz sintetizada, aunque se le puede presentar también en pantalla mediante texto o utilizando un avatar (Griol et al., 2010).

En el campo de la síntesis de voz durante los últimos años se han venido aplicando técnicas estadísticas para mejorar la calidad de las voces generadas. En concreto los sintetizadores basados en modelos ocultos de Markov (**?**; Ljolje and Fallside, 1986; Yu and Young, 2010) se encuentran entre los que mejores resultados obtienen.

1.2. Contenido de este trabajo

En el Capítulo 2 se hará una revisión del estado del arte en los campos tratados en este trabajo: la gestión estadística del diálogo y la simulación de usuarios.

La tarea *EDECAN-SPORT* que será la que se utilice para la evaluación del modelo se presenta en el Capítulo 3. En esta tarea el usuario puede reservar o anular pistas deportivas de un campus universitario.

El modelo estadístico propuesto para la gestión del diálogo se encuentra en el Capítulo 4. Este modelo está basadoen un transductor estocástico de estados finitos que permite la fácil adaptación del modelo a nuevas tareas.

En el Capítulo 5 se presenta la plataforma de generación automática de diálogos utilizada para entrenar el modelo. Gracias a la posibilidad de obtener diálogos etiquetados sin necesidad de intervención humana podemos entrenar nuestros modelos probabilísticos.

Posteriormente, en el Capítulo 6 se muestran los resultados obtenidos aplicando el modelo a la tarea *EDECAN-SPORT*.

En el Capítulo 7 se comenta la estructura y los módulos de un sistema de diálogo completo para esta tarea.

El Capítulo 8 contiene las conclusiones, publicaciones científicas relacionadas y el trabajo futuro.

Finalmente, el Apéndice A detalla la evaluación del sistema presentado con una tarea más sencilla para mostrar que es sencillo adaptar el modelo a nuevas tareas.

Capítulo 2

Estado del arte

En este capítulo se van a revisar algunos de los últimos trabajos publicados en las áreas tratadas en este texto: La gestión del diálogo y la simulación de usuarios.

2.1. Gestión del diálogo

La aplicación de aproximaciones de aprendizaje automático para el aprendizaje de estrategias de gestión de diálogo es una línea de investigación muy popular en los últimos años.

El modelo Partially Observable Markov Decision Process (POMDP) (Young, 2000; Young, 2006) es una generalización del clásico Markov Decision Process en el que tenemos cierta incertidumbre sobre el entorno que nos impide estar totalmente seguros de lo que observamos. La aplicación a los sistemas de diálogo es inmediata, ya que tomamos los turnos de usuario como las observaciones del modelo. POMDP nos permite manejar los posibles errores de reconocimiento y comprensión. La potencia de esta aproximación implica que el aprendizaje de los parámetros es costoso en tiempo y en número de muestras de entrenamiento necesarias. Para conseguir que el modelo sea usable en la práctica para tareas realistas se ha estado trabajando en la búsqueda de nuevos algoritmos de entrenamiento (Thomson and Young, 2009; Young et al., 2010; Jurčíček et al., 2010). Concretamente, el uso de redes bayesianas está bastante extendido (Thomson et al., 2008; Raux et al., 2010).

Otra aproximación reciente en la combinación de un modelo POMDP con uno diseñado manualmente (Williams, 2008). El gestor de diálogo creado artesanalmente emite varias posibles hipótesis cada turno del sistema y el POMDP se encarga de seleccionar la mejora acción.

Un modelo alternativo a los MDP es el Module-Variable Decision Process (MVDP) (Laroche et al., 2010). MVDP no necesita un estado global del sistema, si no que permite trabajar directamente con los campos o atributos

y tomar las decisiones a partir de estos. (Heintze et al.,)

En (Schlangen and Skantze, 2009) se presenta una nueva arquitectura incrementar para un sistema de diálogo. Su principal característica es que un módulo emite hipótesis temporales, los siguientes módulos trabajan con ellas pero sabiendo que en cualquier momento el creador de la hipótesis puede revocarla. Esta idea se asemeja a las transacciones de los sistemas gestores de bases de datos. Aunque un sistema incremental puede utilizar los modelos estadísticos comentados anteriormente,es preferible la utilización de métodos que permitan trabajar con información parcial como Support Vector Machines o Markov Random Fields.

2.2. Técnicas de simulación de usuario

El intenso trabajo en modelos estadísticos para la gestión trae consigo la necesidad de obtener una gran cantidad de diálogos etiquetados. También existe la necesidad de evaluar los modelos obtenidos y utilizar humanos para ello haría muy costoso el desarrollo de los prototipos. Para paliar esta necesidad de usuarios que interactúen con el sistema se han desarrollado multitud de técnicas de simulación. La mayor parte de estas sustituyen a los módulos de reconocimiento y comprensión.

El trabajo (Schatzmann et al., 2006) es una reseña de la técnicas de simulación de usuario más importantes de los últimos años. En él se tratan modelos basados en MDP, HMM (Cuayahuitl et al., 2005) aprendidos a partir de diálogos etiquetados, grafos de decisión (Scheffler and Young, 2000; Scheffler and Young, 2001) o redes bayesianas (Pietquin and Beaufort, 2005; Pietquin and Dutoit, 2005).

El mismo autor propone también una aproximación basada en agenda (Schatzmann et al., 2007). La agenda es una estructura similar a una pila en la que el usuario simulado va almacenando las tareas pendientes de realizar en el diálogo (informar del valor de un atributo, solicitar información, etc). Este método se ha utilizado con éxito para entrenar modelos basados en POMDP (Keizer et al., 2010).

En (Quarteroni et al., 2010) se presenta una técnica modular que incluye un simulador de usuario junto con un simulador de error que modela el canal de comunicación. La emisión de los turnos se realiza utilizando un modelo de bigramas.

En (Ito et al., 2006) se describe un simulador de usuario diseñado especialmente para la evaluación de sistemas que utilizan VoiceXML[1].

[1] VoiceXML es un estándar comercial basado en XML para la implementación de interacción mediante voz.

Capítulo 3

La tarea *EDECAN-SPORT*

La tarea *EDECAN-SPORT* se definió como banco de pruebas para la arquitectura *EDECAN* (Lleida et al., 2006) desarrollada en el seno de los proyectos *EDECAN* y *SD-TEAM*. Consiste en un sistema de diálogo de iniciativa mixta que permite reservar pistas deportivas utilizando el habla espontánea. La arquitectura permite una interacción multimodal: el sistema proporciona información mediante la voz o por pantalla y el usuario puede aportar datos hablar o haciendo clicks en un dispositivo táctil. En este capítulo se explicarán las etiquetas semánticas definidas para *EDECAN-SPORT* y como se ha adquirido el corpus.

La semántica de la tarea se ha obtenido partiendo de un conjunto de diálogos persona-persona proporcionados por el Área de Deportes de la Universidad Politécnica de Valencia. Se han definido un conjunto de etiquetas para los actos de diálogo del usuario, otro para las respuestas del gestor de diálogo y por último las etiquetas que devuelve el gestor de aplicación.

A continuación se muestra una interacción típica entre el usuario y el sistema. Los turnos del sistema está indicados con la letra `S` y los de usuario con la letra `U`.

S0: Bienvenido al servicio de consulta, reserva y anulación de pistas deportivas de la UPV. ¿En qué puedo ayudarle?
U0: Quiero reservar una pista de baloncesto para el viernes 7.
S1: ¿Quiere reservar la pista 3 de baloncesto el 7 de noviembre a las nueve en punto de la mañana?
U1: Sí me viene bien gracias
S2: Le confirmo que la pista 3 de baloncesto ha sido reservada el 7 de noviembre a las 9 en punto de la mañana. ¿Desea algo más?
U2: Sí dime todas las reservas que tenga hechas.
S3: En la pantalla le indicamos las pistas que tiene reservadas. ¿Desea algo más?

U3: Nada más gracias.
S4: Gracias por utilizar este servicio.

Los diálogos siempre comienzan con el sistema saludando al usuario y preguntándole qué desea hacer. Los siguientes turnos dependen de la información que vaya proporcionando el usuario hasta que el usuario decide terminar la interacción y el sistema emite un mensaje de despedida.

3.1. Turnos de usuario

El usuario puede realizar 4 acciones en el sistema: consultar las pistas que hay disponibles, reservar una pista, consultar las pistas que tiene reservadas y cancelar una reserva.

Podemos dividir las acciones anteriores en 2 tipos: las que son meras consultas a la base de datos y las que implican una modificación de la misma. Además, la reserva o cancelación requieren que el usuario haya realizado previamente las respectivas consultas para que el sistema tenga constancia de qué pista debe reservar o anular.

Para modelar la funcionalidad expuesta arriba se han definido las siguientes etiquetas para los actos de diálogo del usuario. Se muestra la etiqueta, la acción asociada y un ejemplo de frase a la que podríamos asociar el concepto.

1. Conceptos dependientes de la tarea (acciones)
    - `AVAILABILITY`: Consultar las pistas disponibles. *Quiero saber lo que hay libre.*
    - `BOOKING`: Reservar una pista. *Reservame la pista, por favor.*
    - `BOOKED`: Consultar pistas reservadas. *¿Me puede decir qué pistas tengo reservadas?*
    - `CANCELLATION`: Cancelar una reserva. *Anulame esa pista.*
2. Conceptos independientes de la tarea (meta-acciones)
    - `ACCEPTANCE`: Afirmación del usuario. *Sí, gracias.*
    - `REJECTION`: Negación del usuario. *No, no, no, el martes no.*
    - `NOT-UNDERSTOOD`: Solicitud de repetición. *¿Puede repetirme lo último que ha dicho?*
    - `NONE`: No se ha podido asociar ninguno de los conceptos anteriores con el segmento. *No estoy muy seguro del día.*

Cada uno de los conceptos anteriores puede tener asignada información complementaria en forma de atributos. Los atributos representan restricciones a la acción del usuario. A continuación se muestran los 6 atributos definidos en esta tarea. Se muestra la etiqueta y posibles valores para los atributos:

- `SPORT`: Deporte. *Tenis.*
- `HOUR`: Hora. *A las 10 de la mañana.*
- `DATE`: Fecha. *El viernes que viene.*
- `COURT-TYPE`: Tipo de pista. *La pista del pabellón.*
- `COURT-NUMBER`: Número de pista. *La pista 3.*
- `ORDER-NUMBER`: Selección de pista relativa. *La primera pista.*

La semántica de la mayoría de atributos es obvia a partir de su nombre. Algunos se utilizan muy raramente como `COURT-NUMBER` mientras que otros como `SPORT` aparecen obligatoriamente en cada diálogo. Para dar libertad se permite que cualquier concepto esté acompañado de un número arbitrario de atributos. La única excepción es el atributo `ORDER-NUMBER`. Este permite al usuario realizar una selección de pista relativa a una oferta anterior del sistema. Por ejemplo, durante un diálogo el sistema realiza una consulta a la base de datos e indica al usuario que tiene 3 pistas disponibles y se las muestra por pantalla. El usuario en este momento puede seleccionar una de las 3 pistas indicando el orden de la misma dentro del listado, generando un atributo de tipo `ORDER-NUMBER` (*Quiero reservar la primera pista*), o bien puede aportar algún tipo de información que permita seleccionar una sola pista (*La de por la mañana*).

Utilizando los conceptos y atributos definidos, podemos etiquetar cada locución del usuario con uno o más conceptos con sus respectivos atributos. A este proceso se le llama interpretación semántica.

En el siguiente ejemplo vemos una locución de un usuario y su interpretación semántica. Se muestra el concepto y debajo los atributos asociados y su valor.

Quiero reserva una pista de pádel para mañana.

BOOKING
  SPORT: padel
  DATE: tomorrow

3.2. Turnos del sistema

El gestor de diálogo puede realizar diversas acciones para ayudar al usuario a cumplir su objetivo.

1. Acciones independientes de la tarea:
   - `OPENING`: Emite el mensaje de bienvenida al sistema y pregunta al usuario qué desea hacer.
   - `CLOSING`: Emite el mensaje de despedida y cierra del diálogo.
   - `ASK`: Pregunta al usuario qué desea hacer.
2. Acciones que ofrecen información al usuario:
   - `SHOW-AVAILABILITY`: Muestra o dice al usuario las pistas disponibles que cumplen los requisitos.
   - `SHOW-BOOKED`: Muestra o dice al usuario las pistas que tienen reservas y que cumplen los requisitos.
   - `NO-ROWS`: Indica al usuario que no hay ninguna pista con las características seleccionadas.
   - `BOOKING-OK`: Indica al usuario que la reserva se ha realizada correctamente.
   - `BOOKING-ERROR`: Indica al usuario que la reserva no se ha podido realizar.
   - `CANCELLATION-OK`: Indica al usuario que la cancelación se ha realizado correctamente.
   - `CANCELLATION-ERROR`: Indica al usuario que la cancelación no se ha podido realizar.
3. Acciones que solicitan información al usuario:
   - `SPORT`: Pregunta al usuario el deporte.
   - `DATE`: Pregunta al usuario la fecha.
   - `HOUR`: Pregunta al usuario la hora.
   - `COURT-NUMBER`: Pregunta al usuario el número de pista.
   - `COURT-TYPE`: Pregunta al usuario el tipo de pista.
4. Acciones que solicitan confirmaciones:
   - `CONFIRMATION-SPORT`: Solicita confirmación de que el deporte es correcto.
   - `CONFIRMATION-DATE`: Solicita confirmación de que la fecha es correcta.

- `CONFIRMATION-HOUR`: Solicita confirmación de que la hora es correcta.
- `CONFIRMATION-COURT-NUMBER`: Solicita confirmación de que el número de pista es correcto.
- `CONFIRMATION-COURT-TYPE`: Solicita confirmación de que el tipo de pista es correcto.
- `CONFIRMATION-BOOKING`: Solicita confirmación de que el usuario quiere reservar la pista seleccionada.
- `CONFIRMATION-CANCELLATION`: Solicita confirmación de que el usuario quiere cancelar la pista seleccionada.

5. Acciones dependientes de la tarea sin interacción con el usuario:

- `AVAILABILITY-QUERY`: Consulta al gestor de aplicación sobre disponibilidad de pistas.
- `BOOKING-QUERY`: Orden al gestor de aplicación para que reserva una pista.
- `BOOKED-QUERY`: Consulta sobre pistas reservas.
- `CANCELLATION-QUERY`: Orden de que anulación de una reserva.

Dentro del conjunto de confirmaciones, las acciones `CONFIRMATION-BOOKING` y `CONFIRMATION-CANCELLATION` tienen una semántica ligeramente distinta a las confirmaciones de atributos. Para estas últimas tratamos de confirmar que el valor que nuestro sistema ha obtenido del usuario es correcto (*¿Ha dicho que quiere jugar al baloncesto?*) mientras que la confirmación de acciones lo que nos permite es averiguar si el usuario desea realmente realizar la acción (*¿Quiere reservar la pista seleccionada?*).

Las acciones que interactúan con el gestor de aplicación (punto 5 de la lista anterior) no emiten ningún mensaje al usuario, si no que funcionan de la siguiente manera: cuando el sistema realiza una acción de este tipo (por ejemplo `AVAILABILITY-QUERY`) el gestor de la aplicación genera una consulta SQL para obtener de la base de datos las pistas disponibles atendiendo a los requisitos introducidos por el usuario. El resultado de la acción se empaqueta y se devuelve al gestor de diálogo que se encargará de emitir una nueva respuesta al usuario. Esta segunda acción del sistema se ha definido mediante reglas sencillas. En caso de que el gestor de aplicación devuelva una sola tupla como resultado, el gestor de diálogo selecciona como respuesta `CONFIRMATION-BOOKING` o `CONFIRMATION-CANCELLATION`, dependiendo de cual sea la acción que esté realizando el usuario. Si ha devuelto más de una emitirá `SHOW-AVAILABILITY` o `SHOW-BOOKED`. En caso de haber 0 tuplas como resultado de la consulta emitirá `NO-ROWS`. En caso de que la interacción no sea para modificar la base de datos, el AM devuelve un

sólo valor indicando si la acción se ha podido realizar o no (`BOOKING-OK`, `BOOKING-ERROR`, `CANCELLATION-OK`, `CANCELLATION-ERROR`).

A continuación se muestra el diálogo mostrado al principio de este capítulo etiquetado semánticamente. En los turnos del sistema se muestran las acciones que realiza contra el gestor de aplicación y los mensajes que emite al usuario.

S0: Bienvenido al servicio de consulta, reserva y anulación de pistas deportivas de la UPV. ¿En qué puedo ayudarle?
`OPENING`
U0: Quiero reservar una pista de baloncesto para el viernes 7.
`BOOKING`
`SPORT:basketball`
`DATE: friday, 07-??-????`
S1: ¿Quiere reservar la pista 3 de baloncesto el 7 de noviembre a las nueve en punto de la mañana?
`AVAILABILITY-QUERY`
(El gestor de aplicación ha devuelto una tupla)
`CONFIRMATION-BOOKING`
U1: Sí me viene bien gracias
`ACCEPTANCE`
S2: Le confirmo que la pista 3 de baloncesto ha sido reservada el 7 de noviembre a las 9 en punto de la mañana. ¿Desea algo más?
`BOOKING-QUERY`
(el gestor de aplicación ha devuelto `BOOKING-OK`*)*
`BOOKING-OK`
U2: Sí dime todas las reservas que tenga hechas.
`ACCEPTANCE`
`BOOKED`
S3: En la pantalla le indicamos las pistas que tiene reservadas. ¿Desea algo más?
`BOOKED-QUERY`
(el gestor de aplicación ha devuelto una tupla)
`SHOW-BOOKED`
U3: Nada más gracias.
`REJECTION`
S4: Gracias por utilizar este servicio.
`CLOSING`

3.3. Adquisición de un corpus mediante Mago de Oz

El conjunto de diálogos persona-persona etiquetados suponían un conjunto de entrenamiento muy pobre para la tarea *EDECAN-SPORT*. Utilizando este pequeño corpus se aprendió una versión preliminar del gestor de diálogo usando la técnica de clasificación basada en redes neuronales (Hurtado et al., 2006). Este gestor de diálogo se utilizó como prototipo en el proceso supervisado de adquisición de un corpus mayor utilizando la técnica del Mago de Oz, ya empleada anteriormente para el proyecto DIHANA (Miguel et al., 2003).

Durante la adquisición se utilizó a un experto (Mago de Oz) que realizaba la interpretación semántica de las frases y a un segundo que hacía el rol de gestor de diálogo. El primer Mago escucha la frase pronunciada por el usuario y simula el comportamiento del reconocedor de voz y el módulo de comprensión, el resultado lo obtienen simultáneamente el prototipo de gestor aprendido y un segundo Mago. Este último puedo aceptar o corregir la acción propuesta por el gestor y enviar la respuesta al usuario que no sabe que está interactuando con humanos en vez de máquinas. Utilizar la técnica del Mago de Oz nos permite obtener a la vez un corpus de turnos de diálogo que incluye la pronunciación del usuario con 4 canales de audio, la transcripción y el etiquetado semántico de las frases y la acción realizada por el sistema con lo que posteriormente podemos aprender modelos estadísticos para los módulos del sistema de diálogo.

Utilizando el doble Mago se obtuvo un conjunto de 143 diálogos, con 16 interlocutores distintos de diferentes partes de España. Los idiomas utilizados fueron castellano, catalán y vasco[1]. Se diseñó manualmente un conjunto de 15 escenarios que cubrían todos los posibles casos de uso de la aplicación. Hay de media 4.9 turnos de usuario por diálogo y 6.7 palabras por turno de usuario.

[1]Los diálogos en vasco no han sido utilizados para la evaluación presentada en este trabajo.

Capítulo 4

Gestión estadística del diálogo

En un sistema de diálogo hablado, el gestor del diálogo (dialog manager, DM) es el módulo encargado de elegir cuál es la mejor acción del sistema en un momento dado teniendo en cuenta el desarrollo del diálogo. En los primeros sistemas de diálogo, esta gestión se realizaba mediante reglas implementadas a mano por el programador o por un experto en la materia, esta aproximación se sigue utilizando hoy en día aunque a medida que los sistemas de diálogo se hacen más y más complejos esta tarea es cada vez más costosa. Durante los últimos años se han venido estudiando métodos estadísticos para la gestión del diálogo. En este capítulo se presenta una nueva aproximación a la tarea.

Se ha desarrollado un gestor de diálogo basado en la modelización estadística de las secuencias de actos de diálogo (turnos de usuario y de sistema). Por tanto necesitamos un corpus de diálogos etiquetados para estimar el modelo, asunto que se tratará en el siguiente capítulo. Dependiendo del número de actos de diálogo diferentes que tenga el gestor de diálogo, la cantidad de muestras de entrenamiento necesarias para obtener un buen modelo varía. Si consideramos sólo un pequeño número de acciones generales en el diálogo (por ejemplo, juntando todas las confirmaciones de fechas en una sola acción `CONFIRMATION-DATE`) podemos obtener un buen modelo utilizando menos diálogos etiquetados que si consideramos acciones más específicas en el modelo (como confirmar una fecha concreta `CONFIRMATION-DATE-11-01-2011`[1]).

Por otra parte, podemos realizar una aproximación mixta en la que el diálogo está gestionado en su mayor parte por métodos estadísticos y algunas situaciones concretas se implementan mediante reglas diseñadas a mano.

[1] Etiqueta ficticia. Las acciones instanciadas específicamente para una fecha requieren una cantidad de mayor de muestras de entrenamiento.

4.1. Modelización estadística del diálogo

Consideramos que un diálogo es una secuencia de pares $(u_i,\ s_{i+1})$, $i = 1...n$ donde u_i es la locución del usuario en el instante i y s_{i+1} es la respuesta del sistema en ese turno.

El diálogo es por tanto una secuencia

$$s_0, (u_0, s_1), (u_1, s_2), (u_2, s_3), \ldots, (u_i, s_{i+1})$$

La respuesta s_{i+1} se selecciona no sólo teniendo en cuenta la última locución del usuario u_i, si no también toda la información proporcionada por el usuario durante el diálogo. Para cada turno del sistema la mejor acción $\hat{s}_{i+1}$ se puede obtener estadísticamente como:

$$\hat{s}_{i+1} = \underset{s \in S}{\operatorname{argmax}} P(s|s_0, u_0, s_1, \ldots, s_i, u_i)$$

Para poder estimar correctamente estas probabilidades condicionales necesitaríamos una cantidad exponencial de diálogos etiquetados. No suele ser posible obtener una cantidad tan elevada de muestras de entrenamiento por lo que nos vemos obligados a asumir que podemos resumir todos los turnos anteriores del diálogo y tomar la decisión a partir de este resumen. En el modelo propuesto, toda esta información (en términos de conceptos y atributos de la tarea) se almacena en una estructura de datos que llamamos registro de diálogo (dialog register, DR). Por tanto, el DR_i es un resumen de lo que ha dicho el usuario hasta el momento i. Ahora sólo necesitamos conocer el registro del diálogo y la información proporcionada en el turno para tomar la decisión:

$$\hat{s}_{i+1} = \underset{s \in S}{\operatorname{argmax}} P(s|DR_i, u_i)$$

Además, asumimos que el valor concreto de un atributo en el DR no es necesario para elegir la siguiente acción del sistema. Por tanto para cada atributo sólo nos interesa saber si ha sido proporcionado o no y su valor de confianza. Para reducir aún más la variabilidad en el espacio de valores del DR se cuantifican el valor de confianza de cada atributo y se consideran solamente tres valores: atributo no proporcionado, baja confianza y alta confianza.

Para la tarea con la que estamos trabajando, *EDECAN-SPORT*, el registro del diálogo contiene los siguientes campos:

1. Acción nombrada por el usuario con su valor de confianza. Uno entre:
   - `AVAILABILITY`
   - `BOOKING`
   - `BOOKED`

- `CANCELLATION`

2. Meta-acciones. Una entre:
   - `ACCEPTANCE`
   - `REJECTION`
   - `NONE`
3. Confianza obtenida para cada uno de los siguientes atributos:
   - `SPORT`
   - `DATE`
   - `HOUR`
   - `COURT-TYPE`
   - `COURT-NUMBER`

Se muestra a continuación un ejemplo de DR para un diálogo en el que el usuario quiere reservar una pista de baloncesto podría ser:

1. Acción: `AVAILABILITY` (confianza: alta)
2. Meta-acción: `NONE`
3. Atributos:
   - `SPORT: basketball` (confianza: alta)
   - `DATE: tomorrow` (confianza: baja)
   - `HOUR: NONE`
   - `COURT-TYPE: NONE`
   - `COURT-NUMBER: NONE`

Como hemos dicho, el contenido real de los atributos no se tiene en cuenta a la hora de elegir la siguiente acción del sistema por tanto para el gestor de diálogo el DR anterior sería equivalente al siguiente, ya que todos los conceptos y atributos coinciden en tipo y valor de confianza aunque no en valor.

1. Acción: `AVAILABILITY` (confianza: alta)

2. Meta-acción: `NONE`
3. Atributos:
   - `SPORT: football` (confianza: alta)
   - `DATE: friday` (confianza: baja)
   - `HOUR: NONE`
   - `COURT-TYPE: NONE`
   - `COURT-NUMBER: NONE`

El atributo `ORDER-NUMBER` no se almacena en el DR ya que tiene un semántica distinta al resto de atributos y se utiliza únicamente a la hora de la actualización de los valores del registro.

El número de registros de diálogo distintos que podemos obtener con esta definición es de 10935.

4.2. Estado del diálogo

En muchas aplicaciones, sobre todo en las que el sistema se limita a ofrecer la información requerida por el usuario, el gestor de diálogo tiene suficiente con el registro de diálogo para inferir la acción que debe llevar a cabo a continuación. Pero en el caso de *EDECAN-SPORT*, disponemos de un gestor de aplicación que nos permite reservar y cancelar pistas y, por tanto, modificar la información almacenada en el sistema. Además, una misma secuencia de turnos puede dar lugar a acciones distintas dependiendo de las respuestas del gestor de aplicación. Por ejemplo, el turno *Quiero reservar una pista de baloncesto el lunes por la tarde* puede generar la acción de reserva de pista o bien emitir un error al usuario indicándole que no hay pistas disponibles o que está sancionado. Estas circunstancias se dan también en el corpus adquirido y ocasiona que secuencias de turnos iguales en cuanto a su DR generen acciones totalmente distintas, dificultando el aprendizaje de modelos estadísticos.

La solución utilizada ha sido ampliar el registro de diálogo con nueva información que permita diferenciar más fácilmente las acciones a tomar. Se ha creado para ello una estructura llamada estado del diálogo (Dialog State, DS). Esta estructura contiene toda la información del DR y añade los siguientes campos:

1. El número de tuplas que ha devuelto la última consulta a la base de datos. Un valor entre:
   - Ningún resultado (0)

   - Un sólo resultado (1)
   - Dos o más resultados (2)

2. El tipo de la última consulta.

   - `AVAILABILITY`
   - `BOOKED`

3. El último turno del sistema

Consideramos que es conveniente diferenciar si el gestor de aplicación no ha devuelto ningún resultado (para mostrar un error, por ejemplo), si ha devuelto sólo uno (para preguntar si desea reservar la pista seleccionada) o si ha devuelto más de uno (para mostrarle las opciones disponibles). Más allá de 2 resultados, el gestor de aplicación siempre ha de realizar la misma acción por lo que no necesita conocer la cardinalidad real de la consulta realizada.

Con este registro ampliado tenemos mucha más información a la hora seleccionar la acción a realizar y aumentamos la riqueza de situaciones posibles ya que ahora disponemos de $2,263,545$ estados distintos[2].

4.3. Gestión de diálogo mediante un transductor estocástico

El modelo de gestión propuesto en este trabajo está basado en el modelo clásico de transductor estocástico de estados finitos (Stochastic Finite-State Transducer, SFST) (Casacuberta and Vidal, 2004). Un SFST es una tupla $(Q, \Sigma, \Delta, q_0, p, f)$ donde:

- Q es el conjunto de estados del transductor.
- Σ es el alfabeto de entrada.
- Δ el alfabeto de salida.
- $q_0 \in Q$ es el estado inicial.
- $p : Q \times \Sigma \times \Delta \times Q \rightarrow [0, 1]$ es la función estocástica de emisión y transición. Nos da la probabilidad de pasar de un estado q_i a otro q_j $(q_i, q_j \in Q)$ dada una entrada $u \in \Sigma$ emitiendo el símbolo de salida $s \in \Delta$.
- $f : Q \rightarrow [0, 1]$ es la probabilidad de los estados finales.

[2]Este número es una cota máxima. Puede haber alguna combinación de valores de los campos del estado que sea imposible de encontrar en la práctica.

Con el fin de utilizar un modelo SFST para la gestión de diálogo proponemos algunas modificaciones al modelo. Para empezar, asimilamos el concepto de estado del SFST a nuestro DS. Tal y cómo hemos definido el estado de diálogo, el espacio de valores es muy grande aunque finito y enumerable por lo que podemos utilizarlo como espacio de estados en nuestro transductor. Además, no necesitamos almacenar explícitamente la tabla de transiciones entre estados ya que el siguiente estado se obtiene actualizando valores en el DS. Sólo necesitamos una función de emisión de símbolos, siendo la transición entre estados un proceso determinista.

Por tanto:

- El conjunto de estados Q está formado por todos los posibles valores que puede tomar el DS.
- El alfabeto de entrada Σ está formado por las posibles acciones del usuario.
- El alfabeto de salida Δ está formado por las acciones del sistema.
- El estado q_0 es aquel consistente en un DS con todos sus valores por defecto. Todos los diálogos comienzan en el mismo estado.
- La función de transición se simplifica y solamente requiere actualizar algunos valores del DS para obtener el nuevo estado.
- La función de emisión p nos da la probabilidad de que un cierto símbolo de salida s sea emitido en un cierto estado q dada una entrada u.
- Los estados finales son aquellos en los que el modelo nos haga emitir una acción `CLOSING`.

La función de emisión $p(q, u, s) \in [0, 1]$ nos proporciona la probabilidad de que un símbolo s sea emitido cuando el sistema se encuentra en el estado q y se ha recibido una entrada de usuario u. En el turno de usuario nos encontramos en cierto estado q (que equivale a un determinado DS), el usuario pronuncia una frase representada por el símbolo u. Seleccionamos la acción a realizar como el máximo de entre todas las posibles:

$$\hat{s} = \underset{s \in \Delta}{\operatorname{argmax}}\, p(q, u, s)$$

Ahora podemos actualizar el DS a partir del turno de usuario y del sistema para generar el nuevo estado q' al que se mueve el transductor. Es decir, para cada posible turno $q \in Q$ almacenamos una matriz donde cada fila es un posible turno del usuario u y cada columna la probabilidad de que se de cierta respuesta del sistema s. Es una aproximación por máxima verosimilitud, estando en el estado q y con la entrada u buscaríamos la

respuesta s con una mayor probabilidad y sería la acción ejecutada por el sistema.

En la Figura 4.1 vemos un ejemplo. Para cada estado q se almacena una matriz cuyas filas representan los turnos del usuario y cuyas columnas representan las posibles acciones del sistema. El modelo recorre la fila correspondiente a u y selecciona la acción s que maximiza la probabilidad. En el ejemplo, `DATE` sería el siguiente turno del sistema.

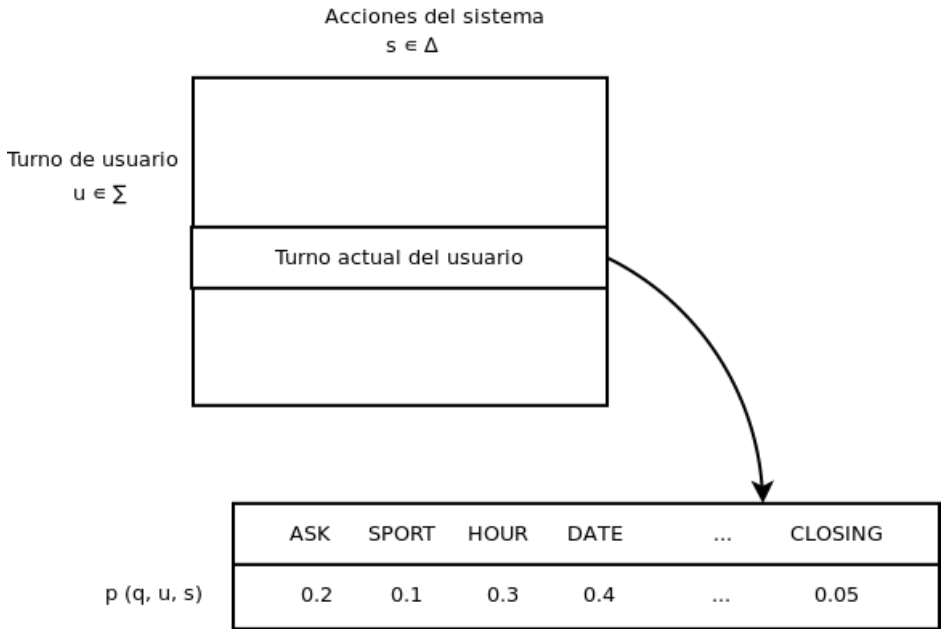

Figura 4.1: Descripción gráfica de la función de emisión de símbolo.

Dado el determinismo en las transiciones del modelo, p puede ser aprendido utilizando un corpus etiquetado como

$$p(q,u,s) = Pr(u,s|q) = \frac{C(q,u,s)}{C(q,u)}$$

donde $C(q,u,s)$ es el número de veces que estando en el estado del diálogo q se ha observado la acción del usuario u y la respuesta del sistema fue s y $C(q,u)$ es un factor de normalización que indica el número de veces que estando en q se ha observado u.

Con esta aproximación, la selección de la mejor acción en el turno i (s_{i+1}) viene dada por una maximización local:

$$\hat{s_{i+1}} = \underset{s_i \in \Delta}{\operatorname{argmax}}\, p(q_{i-1}, u_i, s_i, q_i) = \underset{s_i \in \Delta}{\operatorname{argmax}}\, Pr(u_i, s_i | q_{i-1})$$

El diálogo finaliza cuando la función p devuelve como acción a realizar la emisión del mensaje de despedida `CLOSING`.

4.4. Consideraciones prácticas para el aprendizaje del modelo

Tal y cómo hemos definido el estado del diálogo tenemos $2,263,545$ estados distintos y para cada uno hemos de estimar la probabilidad de emisión de los símbolos de salida. Necesitamos una gran cantidad de diálogos etiquetados para poder aprender los parámetros del modelo y aún con un gran número de estos[3] es probable que algún estado no haya sido visto. Si alcanzamos un estado no visto no tenemos información para emitir ninguna acción. Para evitar este problema que se puede dar en la práctica hemos definido un método heurístico de suavizado que nos permite recuperarnos cuando se den este tipo de situaciones.

Si nos encontramos en el estado q_i y obtenemos la entrada u_i, como no tenemos ninguna salida s_i para la tupla (q_i, u_i)., transitamos para buscar la posible acción al estado anterior (q_{i-1}, u_i). Si en este tampoco tenemos muestras transitamos recursivamente hacia atrás hasta encontrar un q_j que sí haya sido visto[4]. Una vez se ha obtenido una acción para el turno actual s_i, modificamos el estado del diálogo utilizando (q_i, u_i, s_i). Es decir, el estado q_j con el que hemos encontrado la acción a realizar no interviene para la transición al nuevo estado. Si usáramos q_j en lugar del actual q_i, estaríamos perdiendo información ya que el estado tiene una correspondencia biyectiva con nuestro DS que almacena todo el historial del diálogo.

[3] 200,000 diálogos etiquetados, como veremos en capítulos posteriores.

[4] En el caso peor, las transiciones hacia atrás llegarían hasta el estado inicial y no tendríamos forma de saber qué acción realizar. En este caso sólo podemos preguntar al usuario qué quiere hacer, pero esta situación es muy poco probable.

Capítulo 5

Generación automática de diálogos

Para poder aprender los parámetros del SFST tal y cómo hemos visto en el capítulo anterior, necesitamos adquirir un gran corpus etiquetado. Debido al alto coste de la adquisición de diálogos con usuario reales, se propone una nueva aproximación para la generación automática de diálogos que sólo requiere la definición semántica de la tarea y un conjunto de criterios que definan la corrección de un diálogo.

Esta aproximación está basada en la simulación de los principales módulos que participan en un sistema de diálogo hombre-máquina. Utiliza un simulador de usuario y del canal de comunicación por un lado y por otro simuladores de gestor diálogo y de aplicación. En la Figura 5.1 podemos var un esquema de la simulación. Durante la creación automática de diálogos, un módulo llamado gestor de simulación irá solicitando turnos alternativamente al simulador de usuario y al simulador del sistema hasta completar un diálogo. Una vez completado el diálogo se realizará un test de coherencia en el que con una serie de reglas se comprueba que el usuario ha alcanzado su objetivo y que el sistema no ha emitido ninguna acción considerada incorrecta. La implementación realizada no necesita ningún conocimiento sobre la tarea a simular excepto en el módulo que usamos para la evaluación del diálogo.

En las siguientes secciones se explica detalladamente cada uno de estos módulos. Aunque el sistema de generación de diálogos es en su mayor parte independiente de la tarea, se explicará su funcionamiento tomando como ejemplo la tarea *EDECAN-SPORT*.

5.1. El simulador de usuario

El módulo de simulación de usuario nos permite entrenar y/o evaluar un sistema de diálogo ahorrándonos el coste de utilizar usuarios reales. En un

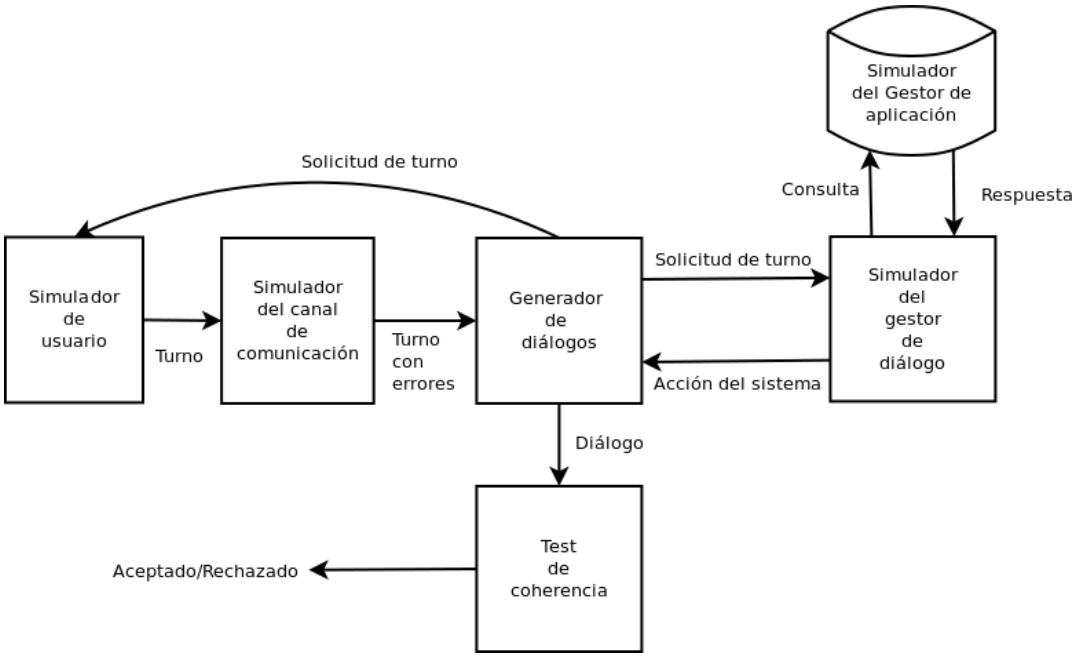

Figura 5.1: Esquema de los módulos implicados en la generación automática de diálogos.
El Generador de diálogos se encarga de solicitar turnos alternativamente al Simulador de usuario y al Simulador del gestor de diálogo.

sistema de diálogo hablado el usuario puede verse como un agente que emite en cada turno un acto de diálogo. Esta locución del usuario se convertirá a texto utilizando un reconocedor automático del habla y posteriormente se traducirá a un frame gracias al módulo de interpretación semántica. Una aproximación muy utilizada para evitar lidiar con texto en lenguaje natural es permitir que el simulador de usuario genere directamente el frame.

El simulador de usuario propuesto en este trabajo genera en cada turno un frame siguiendo la definición semántica de la tarea a utilizar. No requiere ninguna otra información sobre la tarea aparte de los conceptos y atributos de los frames de usuario. Este modelo sin información alguno supone que todos los turnos posibles de usuario son equiprobables. Asumimos además que el usuario actúa en 2 fases: Antes de iniciarse la interacción, el usuario decide cuál es su objetivo y durante el desarrollo del diálogo espera su turno y emite un frame.

5.1.1. El objetivo del usuario

Antes de iniciar, el diálogo el simulador debe tener un objetivo y para alcanzarlo debe de interactuar con el sistema de diálogo. En el caso de

nuestra tarea de reserva de pistas deportivas, los 4 objetivos básicos que puede tener el usuario son:

- Ver las pistas disponibles.
- Reservar una pista.
- Ver las pistas reservadas por el usuario.
- Cancelar una reserva.

Estas posibles metas coinciden con los conceptos que habíamos definido para los frame de usuario: `AVAILABILITY`, `BOOKING`, `BOOKED` y `CANCELLATION`. Para obtener el objetivo del simulador de usuario elegimos aleatoriamente una de estas acciones. Debemos elegir además atributos para concretar el objetivo del usuario por ejemplo, un usuario real podría tener como objetivo *Reservar una pista de baloncesto para mañana a las 9*. Podemos modelar este tipo de objetivos escogiendo aleatoriamente un conjunto de atributos para el concepto objetivo. Estos objetivos serán: `SPORT`, `DATE`, `HOUR`, `COURT-TYPE` y `COURT-NUMBER`.

Por tanto, podemos elegimos al azar uno de los 4 posibles objetivos y le añadimos un subconjunto de los atributos del concepto elegido. Tal y cómo hemos definido el transductor estocástico en el capítulo anterior, no necesitamos generar aleatoriamente valores específicos para los atributos. En nuestro generador simplemente les asignamos el valor *CORRECT*.

Asumimos que si un atributo no aparece en el objetivo significa que el usuario es flexible en el valor de ese atributo. Por ejemplo, para la meta *Reservar una pista de baloncesto para mañana* tendríamos el siguiente frame objetivo:

```
BOOKING
   SPORT:CORRECT
   DATE:CORRECT
```

Los atributos `HOUR`, `COURT-TYPE` y `COURT-NUMBER` no aparecen en el objetivo y cualquier posible valor para estos sería compatible con el objetivo. El siguiente frame sería coherente con la meta anterior:

```
BOOKING
   SPORT:CORRECT
   DATE:CORRECT
   HOUR:CORRENT
```

Mientras que un frame con menos campos no lo sería:

```
BOOKING
   SPORT:CORRECT
```

En la sección dedicada al test de coherencia del simulador veremos cómo se utiliza este frame objetivo para evaluar la corrección de los diálogos.

Diálogos multiobjetivo

Combinando varios frames objetivo podemos crear situaciones en la que el usuario tiene múltiples objetivo. La concatenación de frames se puede interpretar como subobjetivos que se deben cumplir secuencialmente en un mismo diálogo. Tomemos el siguiente frame como ejemplo:

```
CANCELLATION
   SPORT:CORRECT
BOOKING
   SPORT:CORRECT
   DATE:CORRECT
```

Este frame modela una objetivo como *Cancelar la pista de tenis que tengo reservada y reservar una de baloncesto para mañana.*

Este formato para los objetivos del usuario no permite representar ciertas intenciones como podrían ser disyunciones *Reservar tenis o squash.* Pero es suficientemente expresivo para modelar la mayoría de situaciones reales y obtener un buen modelo como veremos en el capítulo de evaluación.

5.1.2. Turnos de usuario

Un usuario real pronunciará una o más frases por turno que serán posteriormente interpretadas semánticamente. Asumimos que un turno de usuario en formato frame constará de a lo sumo un metaconcepto (`ACCEPTANCE` o `REJECTION`) y un concepto dependiente de la tarea (`AVAILABILITY`, `BOOKING`, `BOOKED` o `CANCELLATION`). Además se añade al turno un subconjunto aleatorio de atributos asignándoles el valor *CORRECT* tal y cómo hemos visto en la generación del objetivo.

Los pasos seguidos en el algoritmo de generación de un turno de usuario son los siguientes:

1. Seleccionar un metaconcepto entre { `ACCEPTANCE`, `REJECTION`, `NONE` }. Donde `NONE` es el metaconcepto nulo que indica que en la locución no se ha realiza ninguna afirmación o negación.

2. En caso de haber seleccionado un metaconcepto, asignarle un valor de confianza aleatorio en el rango $[0, 1]$.
3. Seleccionar un elemento entre { `AVAILABILITY`, `BOOKING`, `BOOKED`, `CANCELLATION`, `NONE` }. El concepto nulo indica que no se ha encontrado ningún concepto.
4. En caso de haber seleccionado un concepto, asignarle un valor de confianza aleatorio en el rango $[0, 1]$.
5. Seleccionar un subconjunto aleatorio de { `SPORT`, `DATE`, `HOUR`, `COURT-TYPE`, `COURT-NUMBER` }.
6. Para cada atributo seleccionado, asignarle el valor *CORRECT* y dos valores de confianza aleatorio en el rango $[0, 1]$.

Con este algoritmo podemos obtener con igual probabilidad cualquier turno de usuario $u \in \Sigma$[1]. En el paso 6 generamos 2 números aleatorios para cada atributo, esto simula el comportamiento del módulo de comprensión. Necesitamos conocer la probabilidad de que el segmento de texto dado sea un atributo de cierto tipo; y además la probabilidad de que siendo de ese tipo de atributo, el valor sea el obtenido.

Veamos algunos ejemplos de turnos generados aleatoriamente con los valores de confianza redondeados a 2 decimales y una posible representación en lenguaje natural.

Quería reservar la pista 3 de baloncesto.
`AVAILABILITY` 0,73
`SPORT`:*CORRECT* 0,91 0,41
`COURT-NUM`:*CORRECT* 0,34 0,52

Como vemos, los atributos `SPORT` y `COURT-NUM` tiene 2 valores de confianza. El primero indica que hay un segmento de texto (*de baloncesto*) y que ha sido etiquetado como atributo de tipo SPORT con una confianza 0,91 mientras que el segundo indica la confianza que tiene el traductor a frame del valor concreto del atributo.

De acuerdo, ¿qué tengo reservado?
`ACCEPTANCE` 0,85

[1]Según la definición de Σ del capítulo anterior, es el conjunto que contiene todos los posibles turnos de usuario

BOOKED 0,67

En el ejemplo anterior vemos un turno de usuario que contiene simultáneamente un metaconcepto y un concepto pero ningún atributo.

5.2. Simulador del canal de comunicación

Al generar los turnos de usuario directamente en formato frame, perdemos posibles fuentes de error como son el módulo de reconocimiento del habla y el de comprensión del lenguaje. Para simular el error producido por éstos, definidos el simulador del canal de comunicación. Este módulo toma como entrada un frame generado por el simulador de usuario y devuelve un nuevo frame modificado mediante la introducción de errores.

Todo elemento del frame que lleva adjunto un valor de confianza es susceptible de ser modificado. Cada tipo de campo (metaconceptos, conceptos o atributos) se modifica de una forma distinta, como veremos a continuación, pero la regla utilizada para decidir si se modifica o no es la misma para todos: Generamos un número aleatorio en el rango $[0, 1]$ y lo comparamos con el valor de confianza original, si el nuevo valor supera el original se modifica y si no dejamos el valor actual.

Las modificaciones realizadas por este módulo dependen del tipo de campo a modificar.

- Si hemos de modificar un concepto, cambiamos el tipo de concepto por otro elegido aleatoriamente entre el resto de conceptos (incluyendo el nulo `NONE`[2]. El valor de confianza se queda intacto.
- Los metaconceptos se modifican de la misma forma que los conceptos, intercambiando el tipo de otro metaconcepto o el concepto nulo.
- En caso de modificar el valor de un atributo, cambiamos el valor de *CORRECT* a *ERROR* y dejamos el valor de confianza intacto.
- Si hay que modificar el tipo de un atributo, cambiamos el tipo de atributo por otro y cambiamos el valor de *CORRECT* a *ERROR*. En este caso la confianza del tipo de atributo queda intacta pero la confianza del valor se cambia por un valor muy bajo[3].

Con esta aproximación los campos generados por el simulador de usuario con baja confianza tienen más probabilidad de ser modificados por el canal

[2]Para simular, por ejemplo, que el módulo de comprensión no ha sido capaz de asociar ningún concepto con el turno del usuario.

[3]En las pruebas se le asignada un valor de 0.1.

de comunicación que los que tiene alta confianza. Esto no impide que un campo con un valor alto de confianza introduzca un error o que uno con baja confianza tenga el valor correcto y nos interesa la aparición de estos fenómenos ya que se dan en la práctica por errores en los diferentes módulos y nos permiten obtener un modelo más robusto. En las pruebas realizadas, se introducen alrededor de un 20 % de cambios o errores en los atributos. Es decir, Esta tasa comparable a los valores incorrectos que introducen los módulos de reconocimiento y comprensión cuando se utilizan con usuarios reales.

Modificación avanzada de campos

El simulador de canal comunicación implementado permite además modificar los conceptos (y metaconceptos y tipos de atributos) de forma realista. Se puede utilizar una matriz de confusión para indicar las probabilidades de intercambio de un concepto o atributo por otro. Por ejemplo, es fácil imaginar que el atributo SPORT de *una pista de baloncesto* es poco probable que se confunda por uno de tipo `DATE`. En cambio, los atributos `DATE`, `HOUR` y `COURT-NUMBER` es más fácil que se confundan entre ellos al contener números.

Esta modificación de atributos no ha sido utilizada en la evaluación presentada en este trabajo por falta de datos. No disponemos de un corpus de diálogos en los que conste el valor obtenido por el modelo automático y el real asignado por el Mago de Oz.

5.3. Simulador del gestor de aplicaciones

El gestor de aplicaciones en la tarea *EDECAN-SPORT* realiza consultas a la base de datos y devuelve las tuplas resultantes al gestor de diálogo. A efectos de simulación, y como no almacenamos valores concretos para los atributos, el simulación del gestor se limita a devolver aleatoriamente un entero indicando el número de tuplas que ha devuelto la consulta. Según la definición del estado del diálogo que hemos hecho en el capítulo anterior, sólo tenemos en cuenta tres posibles como resultado de la consulta: ninguna fila devuelva, una sola fila o más de una fila.

La implementación de este módulo es sencilla, para cada consulta que nos envíe el gestor de diálogo devolvemos un entero entre 0 y 2 indicando el número de tuplas que tiene el resultado.

5.4. Simulador del gestor de diálogo

El simulador del gestor actúa de forma similar al simulador de usuario. Cada turno selecciona al azar y con igual probabilidad una acción de las de su lista de 26 acciones. En este caso no necesitamos generar valores confianza

ya que asumimos que el canal de comunicación no perturba el mensaje y el usuario entiende lo que le dice el sistema.

Si la acción seleccionada por el simulador implica una interacción con el gestor de aplicación, se realiza una llamada al simulador del gestor de aplicación que devuelve un entero con la cardinalidad de la supuesta consulta a la base de datos.

5.5. Test de coherencia

Cuando se ha terminado de simular un diálogo, el test de coherencia se encarga de determinar si el diálogo es correcto o no. Esta comprobación se realiza en 2 fases que explicamos a continuación.

Para la primera fase se ha definido una serie de criterios que permiten decidir automáticamente la validez del diálogo. Estos criterios son genéricos e independientes de la tarea concreta. Un diálogo se considera incorrecto si se da alguna de las siguientes situaciones:

- El número de turnos de diálogo excede un umbral fijado manualmente. Este límite es ligeramente superior a la media de turnos en los diálogos adquiridos con usuarios reales.
- El diálogo no empieza con una acción de `OPENING` o no termina con la acción `CLOSING`.
- El gestor de diálogo ha intentado interactuar con el gestor de aplicación modificando algún atributo en su registro con el valor *ERROR*. Por ejemplo, hacer una consulta de disponibilidad teniendo el `SPORT` equivocado no implica un error mientras que realizar una reserva sí.
- El gestor de diálogo ha realizado alguna acción con el gestor de aplicación que aún no ha sido proporcionada por el usuario. Es decir, si por ejemplo el usuario no ha dicho explícitamente `BOOKED`, el gestor no puede realizar una acción de consultar las pistas reservadas.
- El gestor ha intentado realizar alguna acción que requiere información que aún no tiene. Ejemplos de este tipo de error son intentar confirmar un atributo que no ha dicho el usuario aún o preguntar si desea reservar la pista seleccionada sin haber realizado ninguna consulta al gestor de aplicación.

La segunda fase consiste en un análisis del diálogo completo para determinar si el usuario ha cumplido o no su objetivo. En la sección 5.1.1 hemos definido el objetivo de usuario.

Para la tarea *EDECAN-SPORT* la comprobación del objetivo del usuario se realiza a partir del simulador del gestor de aplicación. El gestor almacena todas las consultas que realiza en formato frame.

El simulador de usuario ha creado un frame objetivo que podemos comparar con el frame almacenado en el gestor de aplicación. En caso de que ambos sean iguales el objetivo se ha cumplido y en caso de ser distintos se rechaza el diálogo. Existe una excepción: si el frame obtenido es más específico que el frame objetivo (es decir, es igual al objetivo pero añadiendo algún atributo más) también se considera válido el diálogo.

Todos los diálogos etiquetados que han sido validados por el test de coherencia se almacenan y puede ser usados para entrenar un modelo. En nuestro caso, un transductor estocástico.

5.6. Adaptación del simulador de diálogos a nuevas tareas

Una de las características más destacadas del generador de diálogos presentado en este capítulo es que no necesita mucha información sobre la tarea. Sólo la generación del objetivo del usuario y la comprobación de que el objetivo se ha cumplido son dependientes de la tarea.

Por contra, los simuladores de usuario y del gestor de diálogo utilizan un modelo plano[4] a la hora de emitir sus respectivos turnos. Esto ocasiona que la gran mayoría de los diálogos generados sean después rechazados por el test de coherencia. Para obtener $200,000$ diálogos aceptados en *EDECAN-SPORT* hemos generado más de $9,000,000,000$ diálogos. Por tanto necesitamos generar unos $50,000$ diálogo para obtener uno sólo que cumpla las condiciones.

La adaptación a una nueva tarea requiere que la definamos utilizando el formato de frame con conceptos y atributos tanto para los turnos de usuario como para los del sistema. Necesitamos además definir los campos del registro de diálogo y, si vamos a utilizar los diálogos para entrenar un transductor estocástico, el estado del diálogo. Podemos implementar también las funciones de generación y comprobación del objetivo del usuario, siendo estas tan complejas como necesitemos.

En el Apéndice A se detalla una evaluación utilizando una nueva tarea. Esta tarea modela un contestador automático (o buzón de voz) accesible a través de teléfono.

[4]Llamado así porque todas las posibles respuestas son equiprobables.

Capítulo 6

Evaluación

Se ha aprendido un gestor de diálogo basado en el transductor estocástico presentado en el capítulo 4 utilizando un corpus de diálogos sintéticos creado con el generador automático que hemos detallado en el capítulo 5.

Como conjunto de evaluación hemos tomado los 143 diálogos etiquetados manualmente que se adquirieron con la técnica del Mago de Oz.

Para comprobar la calidad del modelo se han definido las siguientes medidas:

- $\#D$: Diálogos correctos. Número de diálogos aceptados por el test de coherencia que se han utilizado para entrenar el modelo estadístico.
- $|Q|$: Número de estados del transductor. Es decir, el número de estados de diálogo distintos vistos durante la simulación.
- *Out*: Número de estados fuera del vocabulario. Porcentaje de turnos para los cuales el modelo no puede emitir una respuesta.
- ETR (*Exact Turn Rate*): Porcentaje de turnos en los diálogos de test para los cuales el modelo produce exactamente la misma respuesta que el Mago de Oz.
- EDR (*Exact Dialog Rate*): Porcentaje de diálogos de test en los que el modelo ha seleccionado la misma acción que el mago para todos los turnos.
- ADR (*Accepted dialog rate*): Porcentaje de diálogos de test que son aceptados por el transductor. Es decir, existe un camino en el modelo desde el estado inicial hasta algún final, aunque puede que no con las acciones del sistema correctas.

Se han hecho pruebas variando el número de diálogos de entrenamiento. Además se ha probado también el uso del suavizado propuesto en la sección

4.4. El modelo se ha obtenido utilizando exclusivamente diálogos generados sintéticamente y se ha evaluado con todos los 143 diálogos etiquetados manualmente.

A la hora de generar un diálogo correcto se seleccionó aleatoriamente un frame objetivo. Entonces se comenzaron a generar diálogos para ese mismo objetivo hasta que se consiguió uno válida para ese frame, pasándose a seleccionar un nuevo objetivo. Esta aproximación hace más costosa la generación pero permite que los frames objetivo más complicados tengan la misma probabilidad de aparecer en el corpus final que los frames sencillos. Con nuestro test de coherencia, es mucho más sencillo que un diálogo cumpla un objetivo muy genérico como *Reservar una pista de baloncesto* que uno más específico como *Reservar una pista de baloncesto mañana a las 9 de la mañana*. Cuántos más atributos tiene, más complicado será obtener un diálogo que los contenga con valor correcto.

Para la obtención de los 200,000 diálogos utilizados en las pruebas se utilizó un ordenador de sobremesa corriente. Fueron necesarias 24 horas aproximadamente para recopilar los diálogos requeridos. El generador es capaz de simular y evaluar miles de diálogos por segundo aunque el tiempo requerido para obtener uno válido varía dependiendo de la complejidad del objetivo del usuario.

6.1. Resultados sin suavizado

La tabla 6.1 muestra la evolución de los resultados del modelo a medida que se añaden más diálogos sintéticos.

$\#D$	$\|Q\|$	Out	ETR	EDR	ADR
1,000	3,121	0.456	0.343	0.007	0.084
5,000	9,543	0.241	0.573	0.105	0.259
10,000	14,375	0.176	0.640	0.168	0.399
20,000	20,894	0.116	0.640	0.161	0.545
50,000	32,332	0.096	0.701	0.245	0.601
100,000	43,163	0.069	0.753	0.266	0.727
120,000	46,249	0.069	0.734	0.245	0.727
150,000	50,211	0.069	0.749	0.266	0.727
170,000	52,538	0.069	0.749	0.266	0.727
200,000	55,645	0.067	0.750	0.266	0.734

Cuadro 6.1: Resultados del modelo variando el número de diálogos utilizados para el entrenamiento.

Vemos que el número de turnos fuera del modelo va reduciéndose a medida que aumentamos el número de diálogos pero llegada una cierta cantidad

el proceso de mejora muy lentamente. Para poder obtener muestras que recorran los más de 2 millones de estados posibles necesitaríamos millones de diálogos. Por ello, parece necesario utilizar algún método de suavizado que nos evite la necesidad de haber visto todos los estados durante el entrenamiento.

6.2. Resultados con suavizado

Se han realizado las mismas pruebas que en la sección anterior añadiendo el método de suavizado presentado en la sección 4.4. En la tabla 6.2 podemos ver los resultados.

$\#D$	$\|Q\|$	Out	ETR	EDR	ADR
1,000	3,121	0.276	0.423	0.014	0.196
5,000	9,543	0.159	0.647	0.182	0.441
10,000	14,375	0.053	0.746	0.343	0.783
50,000	32,332	0.023	0.769	0.371	0.895
100,000	43,163	0.000	0.817	0.413	1.000
150,000	50,211	0.000	0.813	0.413	1.000
200,000	55,645	0.000	0.813	0.413	1.000

Cuadro 6.2: Resultados del modelo variando el número de diálogos utilizando suavizado.

Como en el método sin suavizado, todas las medidas mejoran cuando se aumenta el número de diálogos, aunque a partir de 100,000 la mejora se frena.

6.3. Comentarios

El modelo obtiene un porcentaje de turnos correctos relativamente alto a pesar de no haber utilizado muestras etiquetadas manualmente. Aunque quizá un 19 % de error pueda ser algo elevado para una aplicación comercial, este modelo puede utilizarse para obtener fácilmente un prototipo que se permita realizar una adquisición de diálogos con usuarios reales.

El simulador del canal de comunicación (que introduce errores dependiendo del valor de confianza generado previamente) junto con el test de coherencia, parecen inducir al modelo a *aprender* que debe confirmar los atributos con confianza baja.

Creemos que aumentar aún más el número de muestras no mejorará sensiblemente los resultados de cobertura del modelo. En este sentido, los resultados son mejores utilizando suavizado con el que obtenemos una cobertura total a partir de un número relativamente bajo de diálogos.

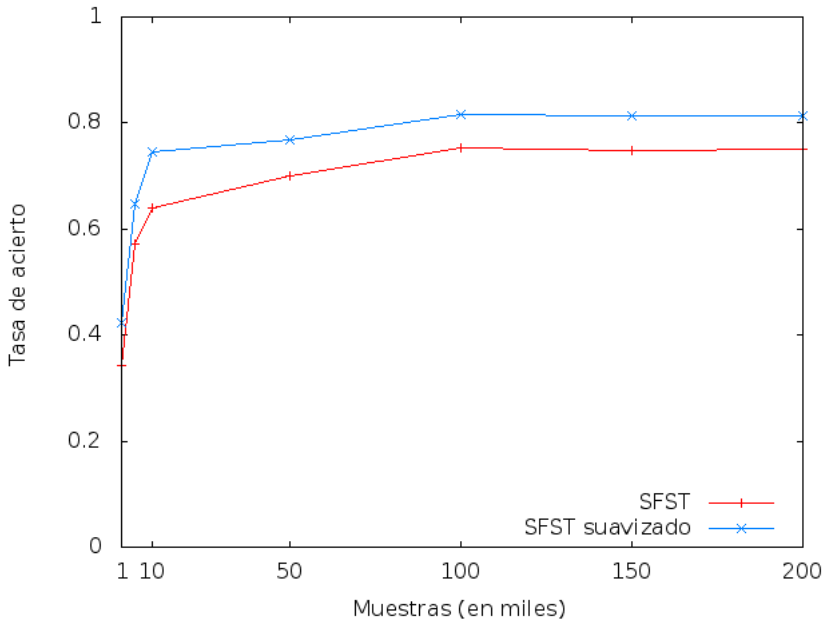

Figura 6.1: Comparación entre el modelo normal y el modelo con suavizado

Cabe destacar que al utilizar suavizado, la tasa de turnos correctos aumenta (ver Figura 6.1). Esto nos indica que además de conseguir evitar estados nos vistos, seleccionando la respuesta a partir de un estado anterior, la acción elegida en ciertos casos es la correcta.

Aunque el porcentaje de turnos correctos es del 81 % utilizando suavizado, el número de diálogos en los que cada turno del modelo coincide con los turnos que utilizó el Mago de Oz es relativamente bajo (41.3 %). Esto se debe parcialmente a que, en algunas situaciones, el modelo ha realizado alguna acción que aún siendo diferente a la referencia, no es incorrecta. Veamos un ejemplo: en un turno del diálogo, el usuario solicita reservar una pista de baloncesto el sábado. Por problemas de reconocimiento, los dos atributos (`SPORT` y `DATE`) tienen asignada una baja confianza. En esta situación, el modelo podría pedir confirmación de cualquiera de los 2 valores. Mediante la revisión manual de las acciones emitidas por el modelo se detectaron casos en los que esto ocurría, pero se ha querido utilizar un método de evaluación totalmente automático.

Capítulo 7

Sistema de diálogo para EDECAN-SPORT

Con el fin de evaluar el desarrollo de los distintos módulos que se han desarrollado en el seno del proyecto de investigación *SD-TEAM* se ha construido un sistema de diálogo completo para la tarea *EDECAN-SPORT*. Para la interconexión de los módulos se ha utilizado la arquitectura y los protocolos desarrollados en el proyecto *EDECAN*. Esta arquitectura permite que cada módulo se ejecute independientemente del resto, posiblemente en una máquina distinta, y que la comunicación entre módulos se realice de forma sencilla utilizando un formato XML (Lleida, 2005). Permite la sustitución rápida de las diferentes partes que componen el sistema y es por tanto una plataforma ideal para evaluar sistemas de diálogo.

La arquitectura contiene los módulos típicos de un sistema de diálogo hablado (ver Figura 7.1): reconocimiento de voz, comprensión del lenguaje, gestor de diálogo, generador de respuestas, sintetizador de voz, además de módulos específicos para el gestor de aplicación y la interacción mediante pantalla táctil.

7.1. Módulo de reconocimiento de voz

La arquitectura *SD-TEAM* permite la integración de múltiples reconocedores de voz. Se puede intercambiar fácilmente utilizando un fichero de configuración y el resto del sistema no se ve afectado por ello. Actualmente se utilizan 2 reconocedores: Loqueando ASR y un reconocedor desarrollado en nuestro grupo llamado MASR.

La señal que se obtiene del micrófono se escala un factor de pre-énfasis utilizando un filtro paso-alto FIR $H(z) = 1 - 0{,}95z^{-1}$ y posteriormente preprocesado para obtener una secuencia de tramas o vectores acústica. Cada 10ms se aplica una ventana Hamming de tamaño 20ms para obtener cada trama, que contiene 39 parámetros: energía, los primeros 12 MFCC y

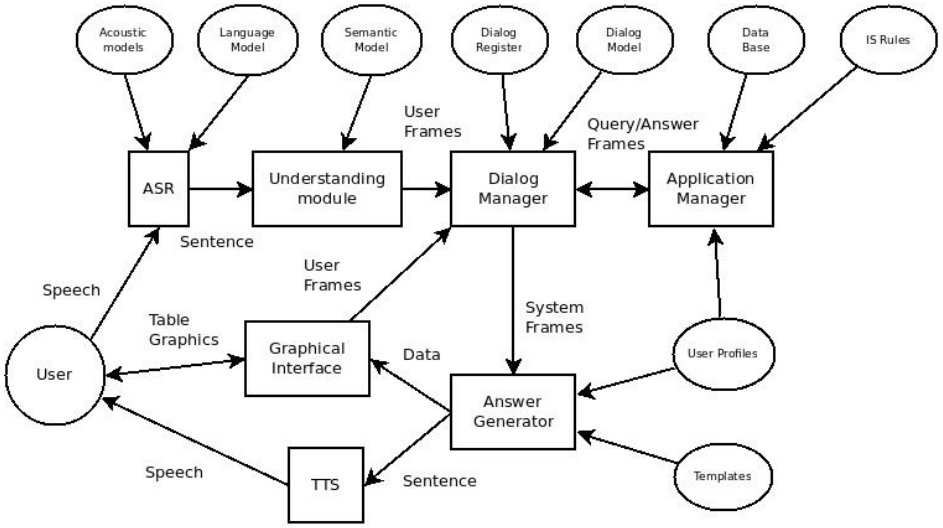

Figura 7.1: Diagrama del sistema de diálogo desarrollado para la tarea EDECAN-SPORT.
Figura extraída de (Segarra et al., 2010).

sus respectivas primera y segunda derivadas.

Los HMM se entrenaron utilizando el software HTK y el corpus en castellano de Albayzin. Este corpus está balanceado fonéticamente y consta de 6 horas de voz (Moreno et al., 1993).

Para el aprendizaje de los modelos de lenguaje de n-gramas utilizados por Loquendo y MASR se han utilizado los diálogos persona-persona de *EDECAN-SPORT* obtenidos mediante la técnica del Mago de Oz. Por tanto el modelo de lenguaje está especializado en la tarea.

Ambos reconocedores son capaces de devolver como resultado tanto la frase más probable como las n mejores (n-best).

7.2. Módulo de comprensión del lenguaje

Se propone un proceso de comprensión del lenguaje que funciona en 2 fases (Segarra et al., 2002; Hurtado et al., 2004).

La primera fase consiste en una transducción de la frase de entrada en formato texto que ha devuelto el reconocedor a una representación intermedia que segmenta la frase en subcadenas y asigna a cada una etiqueta de concepto o atributo. Existe también la etiqueta especial `NULL` que indica que esa subcadena no aporta información semántica.

El objetivo de esta primera fase es encontrar la mejor secuencia de unidades semánticas a partir de la frase de entrada. En la Figura 7.2 se muestra un diagrama del modelo de comprensión. Dada la frase:

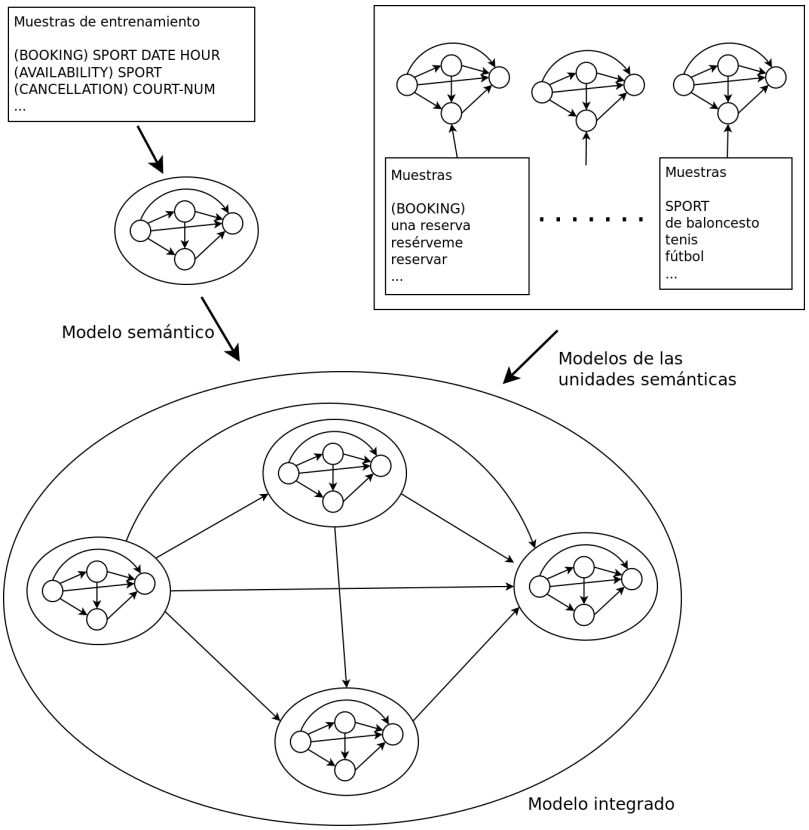

Figura 7.2: Modelo de comprensión.

Quiero reservar una pista de baloncesto el jueves por la mañana.

La representación intermedia en la primera fase de comprensión sería:

Quiero:`NULL`
reservar:`BOOKING`
una pista de baloncesto:`SPORT`
el jueves:`DATE`
por la mañana:`HOUR`

Como la representación intermedia es ya bastante próxima a lo que sería un frame semántico, la segunda fase consta simplemente de una serie de reglas que convierten cada segmento encontrado en la primera fase a su representación canónica en formato frame. El resultado de esta segunda fase de comprensión para el ejemplo anterior sería:

```
BOOKING
   SPORT:basketball
   DATE:[thursday-??-??-????]
   HOUR:morning
```

Esta segunda fase se limita a eliminar segmentos etiquetados como `NULL`, a reordenar los atributos y a instanciarlos con un valor correcto. Por ejemplo convirtiendo la fecha *el jueves* al valor `[thursday-??-??-????]`.

Del corpus anotado de entrenamiento se aprenden dos tipos de modelos: uno que representa la concatenación de unidades semánticas y otro que modela el contenido léxico de cada unidad[1]. En ambos casos se ha utilizado un modelo de bigramas. El proceso de decodificación se realiza utilizando Viterbi que nos proporciona la mejor secuencia de segmentos.

Es decir, dada la secuencia de entrada $w = w_1 w_2 \cdots w_n \in W$, el proceso consiste en buscar la secuencia de unidades semánticas $v = v_1 v_2 \cdots v_k \in V$ que maximiza la probabilidad:

$$\hat{v} = \underset{v}{\operatorname{argmax}}\, P(w|v)P(v)$$

El término $P(w|v)$ es la probabilidad de la secuencia de palabras w dada la unidad semántica v. Estas probabilidades se estiman utilizando el algoritmo de Viterbi como el máximo de todas las posibles segmentaciones de w en $|v|$ segmentos.

$$P(w|v) = \underset{\forall l_1, l_2, \ldots, l_{t-1}}{\text{máx}} \Big\{ P(w_1, \ldots, w_{l_1}|v_1)P(w_{l_1}) \cdot P(w_{l_1+1}, \cdots, w_{l_2}|v_2) \cdot \cdots \cdot P(w_{l_{k_1}+1}, \ldots, w_n|v_k) \Big\}$$

Si se usan modelos de bigramas, la probabilidad de cada segmento dada la unidad semántica asociada es:

$$P(w_i, \ldots, w_j|v_s) = \prod_{k=i}^{j} P(w_k|w_{k-1}, v_s)$$

donde el término $P(v)$ es la probabilidad en el modelo de bigramas de la secuencia v.

$$P(v) = \prod_{i-1}^{k} P(v_i|v_{i-1})$$

[1] Es decir, los segmentos y las palabras asociados a cada unidad semántica.

Figura 7.3: Fotografía del quiosco utilizado durante las pruebas.
El quiosco incorpora una pantalla táctil, micrófonos y altavoces. El teclado se encuentra deshabilitado.

7.3. Gestor de diálogo

Para la gestión de diálogo se utiliza el modelo de transductor estocástico presentado en los capítulos anteriores de este trabajo. Se ha aprendido el modelo utilizando los diálogos adquirido utilizando la técnica del Mago de Oz y un corpus de diálogos sintéticos generados con nuestro simulador de diálogos.

7.4. Interfaz multimodal

La respuesta del sistema le llega al usuario por dos canales: voz y texto. Por un lado tenemos el sintetizador que se encarga de convertir la respuesta del sistema a voz y la reproduce utilizando los altavoces. Las respuestas del sistema utilizan un sistema de plantillas donde cada mensaje depende de la acción que haya emitido el gestor de diálogo, la respuesta del gestor de aplicación y el contenido del registro de diálogo.

A continuación se muestra un pequeño fragmento del fichero de plantillas:

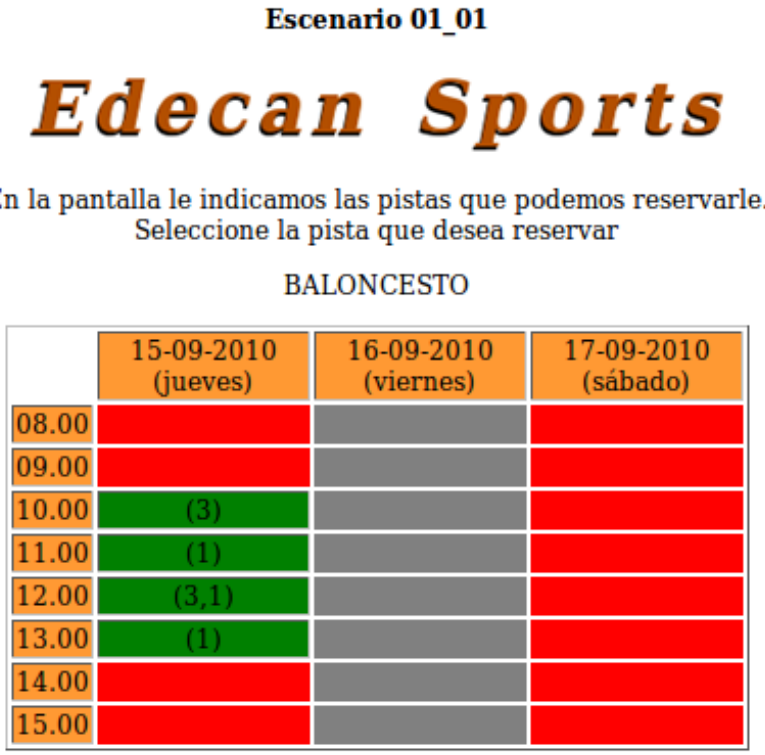

Figura 7.4: Captura de pantalla de la información mostrada al usuario. El usuario ha solicitado reservar una pista baloncesto jueves o viernes. Las celdas en verde representan las horas disponibles, las rojas que no hay pistas disponibles. Las celdas grises están fuera de la selección del usuario y no se consideran.

OPENING # Bienvenido al servicio de consulta, reserva y anulación de pistas deportivas de la UPV. ¿En qué puedo ayudarle?
OPENING # Bienvenido a EDECAN SPORT. ¿En qué puedo ayudarle?
ASK # Por favor, dígame qué es lo que desea.
ASK # ¿En qué puedo ayudarle?
CANCELLATION-OK # La cancelación se ha realizado correctamente.
CANCELLATION-OK # Su pista de [SPORT] [DATE] ha sido cancelada.
(CANCELLATION-OK) # La pista indicada se ha liberado.
...

Cada acción del sistema dispone de varias posibles frases para tratar de hacer más natural la interacción con el usuario. En el momento de sintetizar una respuesta, se selecciona aleatoriamente entre las candidatas. Vemos

que por ejemplo una de las acciones `CANCELLATION-OK` tiene 2 campos para instanciar valores de los atributos. El generador de respuestas tiene acceso al registro de diálogo y en caso de no disponer de algunos de esos atributos (`SPORT` y `DATE`) no considerará esa frase a la hora de seleccionar el mensaje a sintetizar.

La otra vía de comunicación con el usuario es la pantalla táctil de quiosco (ver Figuras 7.3 y 7.4). Se muestran por pantalla los resultados que devuelve el gestor de aplicación para hacer más sencillo al usuario la selección de la pista deportiva que desea.

La pantalla táctil permite además que el usuario pulsé en la pista que desea y esta se seleccione sin necesita de utilizar la voz. En la Figura 7.4 vemos que hay pistas disponibles de baloncesto para el jueves 15 de septiembre por la mañana. Si el usuario pulsa en uno de los recuadros verdes, el módulo de interfaz gráfica emitirá un frame de usuario con los datos de la pista seleccionada.

```
SPORT:basketball
   DATE:[15-09-2010]
   HOUR:10:00:am
```

Gracias a la implementación desacoplada de la arquitectura *EDECAN*, el frame llega al gestor de diálogo de forma transparente como si fuera una locución de usuario. El gestor de diálogo actúa como si el usuario hubiera pronunciado *La de las 10 de la mañana*.

7.5. Resumen

En este capítulo hemos repasado los módulos que componen el sistema de diálogo desarrollado por el grupo de investigación ELiRF para la tarea *EDECAN-SPORT*.

Las principales características del sistema es que posee una arquitectura desacoplada que permite que cada módulo sea desarrollado independientemente del resto. La interacción es multimodal, ya que permite al usuario introducir información mediante voz y utilizando la pantalla táctil y el sistema puede responderle por esos mismos canales. Demuestra además que es posible desarrollar sistemas de diálogo en el que los módulos estén basados en modelos estadísticos con la ventaja que esto supone en cuanto a facilidad de diseño, al no tener que implementar mediante reglas el comportamiento del sistema.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En este trabajo se ha presentado una nueva aproximación estadística para la gestión del diálogo. El modelo propuesto está basado en un transductor estocástico de estados finitos (SFST) que decide en cada turno, la acción que debe realizar el sistema de diálogo. En vez de trabajar con todo el historial del diálogo se define un registro de diálogo que contiene la información aportada por el usuario con su valor de confianza (en términos de alta o baja confianza). Este registro se enriquece con información sobre el gestor de aplicación y el último turno del sistema para dar lugar al estado del diálogo. Con esto conseguimos un espacio de estados para el transductor lo suficientemente grande como para poder representar un amplio abanico de situaciones sin necesidad de almacenar un número exponencial de estados.

Para el aprendizaje de los parámetros del modelo se propone una nueva plataforma para la generación automática de diálogos que está formada por simuladores de los principales módulos que forma un sistema de diálogo. El simulador de usuario y del gestor no utilizan ningún tipo de información sobre la tarea si no que cada turno emiten aleatoriamente un posible turno. Gracias a este comportamiento nos aseguramos que al simular un gran número de diálogos tendremos muestras de la mayoría de situaciones que se pueden dar en el sistema. Se dan además casos en los que el modelo aprende por su cuenta que debe confirmar valores de los atributos que tienen baja confianza.

Ambas técnicas se han probado en una tarea de reserva y cancelación de pistas deportivas. Los resultados de la evaluación muestran que el modelo es capaz de obtener unos resultados prometedores si tenemos en cuenta que se ha utilizado la mínimo información posible relativa a la tarea.

8.2. Publicaciones relacionadas

El modelo estadístico para la gestión del diálogo y el método de generación automática de diálogos han sido presentados en el principal conferencia internacional relacionada con las tecnologías del habla.

L.Hurtado, J.Planells, E.Segarra, E.Sanchis, D.Griol.
A Stochastic Finite-State Transducer Approach to Spoken Dialog Management.
In Proceedings of Interspeech'2010, pag. 3002-3005, 2010.

Además, se presentó en una conferencia nacional un prototipo funcional del sistema de diálogo para *EDECAN-SPORT* el cual incorpora como modelo de gestión de diálogo el presentado en el presente trabajo.

E.Segarra, L.Hurtado, J.A. Gómez, F.García, J.Planells, J.Pastor, L.Ortega, M.Calvo, E.Sanchis.
A prototype of spoken dialog system based on statistical models.
In Proceedings of FALA 2010, Vigo, Spain, November 2010.

8.3. Trabajo futuro

El modelo descrito en este texto abre varias posibles líneas de investigación. Estamos interesados en buscar nuevas formas de suavizar el modelo para lidiar con estados no vistos durante el entrenamiento. Uno de los métodos que se van a evaluar en un futuro es la utilización de un algoritmo de búsqueda que trate de optimizar la respuesta para concluir el diálogo lo más rápidamente posible.

También sería interesante estudiar formas de combinación de diálogos etiquetados manualmente con diálogos creados automáticamente. Actualmente se puede añadir estos diálogos correctos al modelo, pero al haber generado cientos de miles de muestras, los diálogos con usuario reales pierden cualquier influencia estadística que pudieran tener. Por ello, se debe definir algún tipo de ponderación que permita dar más importancia a unos sobre otros.

El transductor presentado en este trabajo utiliza como entrada el turno de usuario que ha obtenido el reconocedor de voz y que posteriormente ha traducido a frame el módulo de comprensión del lenguaje. Esto ocasiona que, en un momento dado, el transductor se encuentre en un único estado. Pero la mayoría de reconocedores son capaces de devolver las n mejores hipótesis (n-best) por lo que podríamos definir un transductor no determinista en el que cada una de las hipótesis generara una transición a un nuevo estado y utilizar las tablas de emisión de cada estado para obtener la acción a realizar.

Otra de las líneas interesantes de mejora del gestor de diálogo es utilizar técnicas de aprendizaje activo que permitan adaptar el modelo mientras se utiliza. Utilizando algún tipo de métrica, el sistema debería ser capaz de detectar estados en los que falla reiteradamente y avisar al diseñador para que éste le oriente sobre qué acción realizar. Con esto podríamos partir de un prototipo e ir mejorándolo con el uso, reduciendo el tiempo necesario de entrenamiento.

Apéndice A

Evaluación con la tarea *VOICEMAIL*

En este capítulo se va a presentar una tarea llamada *VOICEMAIL*. Una tarea muy sencilla que se ha utilizado en los últimos años como ejemplo en modelos como el POMDP (Young, 2000). Se muestran brevemente las etiquetas utilizadas y su significado. Posteriormente se comenta la evaluación realizada.

A.1. Semántica de la tarea

Esta tarea modela un contestador de voz que se accede por vía telefónica. Al acceder, el sistema reproduce un mensaje de los almacenados en memoria. Cuando termina, pregunta al usuario si quiere conservar el mensaje o eliminarlo. El diálogo finaliza cuando el sistema guarda o borrar el mensaje.

A.1.1. Etiquetas de usuario

El usuario sólo tiene dos acciones posibles: decir que quiere almacenar el mensaje o borrarlo. Se necesita añadir además una etiqueta para indicar que no se ha podido reconocer el turno del usuario.

- `save`: El usuario quiere almacenar el mensaje.
- `delete`: El usuario quiere eliminar el mensaje.
- `mumble`: Ruido.

La etiqueta `mumble` se utiliza cuando el reconocedor o el módulo de comprensión no han sido capaces de discernir la acción que ha pedido el usuario. En esta tarea no hay atributos en los conceptos.

A.1.2. Etiquetas del sistema

El sistema sólo puede realizar 3 acciones:

- `ask`: Preguntar al usuario si quiere guardar o borrar el mensaje.
- `doSave`: Guardar el mensaje y finaliza el diálogo.
- `doDelete`: Sistema borrar el mensaje y finaliza el diálogo.

Vemos que no existen confirmaciones explícitas de la intención del usuario. El sistema sólo es capaz de preguntar si desea guardar o borrar.

A.1.3. Diálogos de ejemplo

Se muestran a continuación 2 diálogos de ejemplo para ilustrar la tarea presentada. Se muestran los turnos del sistema (etiquetados con la letra S y el número de turno) y los turnos del usuario (letra U). En los turnos de usuario se muestra además el valor de confianza del concepto.

En el ejemplo siguiente el usuario emite el turno `delete` con confianza suficiente y el sistema decide borrar el mensaje.

S0: `ask`
U0: `delete` (confianza alta)
S1 `doDelete`

En este, el sistema no tiene claro lo que dice el usuario y continúa preguntando hasta que obtiene un turno con confianza alta.

S0: `ask`
U0: `mumble` (confianza baja)
S1: `ask`
U1: `save` (confianza alta)
S2: `doSave`

La simplicidad de la tarea no permite mucha variabilidad en la interacción del usuario con el sistema.

A.2. Estrategia óptima de gestión del diálogo

La estrategia óptima para la tarea consiste en preguntar al usuario (`ask`) hasta que éste emite una acción `save` o `delete` con confianza alta. Puede darse el caso que el usuario diga que quiere guardar el mensaje pero, por errores en el canal de comunicación, el módulo de comprensión decida que el usuario quiere borrar y le asigne una confianza alta. Ante este tipo de errores nada podemos hacer desde el punto de vista del gestor y necesitaríamos mejorar los módulos de reconocimiento y comprensión para evitarlos.

A.3. Evaluación

Se generaron 1,000 diálogos sintéticos utilizando la plataforma de simulación presentada en el presenta trabajo, con la única modificación del Test de Coherencia, que se ha simplificado.

Las condiciones que se comprueban son:

- Sólo se permite una acción de usuario y del sistema por turno, debido a las restricciones de la tarea.
- El diálogo se finaliza cuando el sistema emite un turno `doSave` o `doDelete`.
- Para determinar si el diálogo ha acabado con éxito, se compara el frame objetivo del usuario con la acción del sistema. En caso de coincidencia se acepta el diálogo, si no se rechaza.

No se disponen de ejemplos de test para realizar la evaluación, pero dado que el modelo consta únicamente de 6 estados se procedió a su revisión manual. El modelo obtenido sigue en todo momento la estrategia óptima.

A.4. Conclusiones

En este Capítulo se ha mostrado que el modelo puede utilizarse para aprender la estrategia óptima para la gestión de un sistema de diálogo hablado. Si bien estos resultados no son concluyentes ya que la tarea es muy sencilla, sirve como punto de partida para trabajar con tareas más complejas.

Referencias

Aubert, X. and Ney, H. (1995). Large vocabulary continuous speech recognition using word graphs. In *Proc. of ICASSP 95*.

Casacuberta, F. and Vidal, E. (2004). Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2):205–225.

Cuayahuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-computer dialogue simulation using hidden markov models. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding'05*.

De Mori, R., Bechet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., and Tur, G. (2008). Spoken language understanding: A survey. *IEEE Signal Processing magazine*, 25(3):50–58.

Dinarelli, M., Moschitti, A., and Riccardi, G. (2009). Concept Segmentation And Labeling For Conversational Speech. In *Interspeech*, Brighton, U.K.

Esteve, Y., Raymond, C., Bechet, F., and Mori, R. D. (2003). Conceptual Decoding for Spoken Dialog systems. In *Proc. of EuroSpeech'03*, pages 617–620.

Griol, D., Hurtado, L. F., Segarra, E., and Sanchis, E. (2008). A statistical approach to spoken dialog systems design and evaluation. In *Speech Communication*, volume 50, pages 666–682.

Griol, D., Patricio, M., Molina, J., Arroyo, A., Callejas, Z., and López-Cózar, R. (2010). Integración de los sistemas de diálogo para la interacción en redes sociales. *Procesamiento del lenguaje natural*, (44):107–114.

Hahn, S., Lehnen, P., Heigold, G., and Ney, H. (2009). Optimizing CRFs for SLU Tasks in Various Languages Using Modified Training Criteria. In *Interspeech*, Brighton, U.K.

He, Y. and Young, S. (2003). A data-driven spoken language understanding system. In *Proc. of ASRU'03*, pages 583–588.

He, Y. and Young, S. (2006). Spoken language understanding using the hidden vector state model. *Speech Communication*, 48:262–275.

Heintze, S., Baumann, T., and Schlangen, D. Comparing Local and Sequential Models for Statistical Incremental Natural Language Understanding.

Hurtado, L., Griol, D., Segarra, E., and Sanchis, E. (2006). A Stochastic Approach for Dialog Management based on Neural Networks. In *Procs. of InterSpeech'06*, pages 49–52, Pittsburgh, USA.

Hurtado, L., Segarra, E., García, F., and Sanchis, E. (2004). Language understanding using n-multigram models. In *Advances in Natural Language Processing, Proceedings of 4th International Conference EsTAL*, volume 3230 of *Lecture Notes in Computer Science*, pages 207–219. Springer-Verlag.

Ito, A., Shimada, K., Suzuki, M., and Makino, S. (2006). A User Simulator Based on VoiceXML for Evaluation of Spoken Dialog Systems. In *Proc. of InterSpeech 2006 - ICSLP*, Pittsburgh, USA.

Jurafsky, D. (2009). Speech and language processing.

Jurčíček, F., Thomson, B., Keizer, S., Mairesse, F., Gašić, M., Yu, K., and Young, S. (2010). Natural Belief-Critic: A Reinforcement Algorithm for Parameter Estimation in Statistical Spoken Dialogue Systems. In *Eleventh Annual Conference of the International Speech Communication Association*. ISCA.

Keizer, S., Gašic, M., Jurcıcek, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010). Parameter estimation for agenda-based user simulation.

Laroche, R., Putois, G., and Bretier, P. (2010). Optimising a handcrafted dialogue system design. In *Procs. of InterSpeech'10*, pages 3010–3013, Makuhari Chiba, Japan.

Lemon, O., Georgila, K., and Henderson, J. (2006). Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the talk towninfo evaluation. In *Proc. of SLT'06)*, Aruba.

Levin, E. and Pieraccini, P. (1995). Concept-based spontaneous speech understanding system. In *Proc. of EuroSpeech'95*, pages 555–558.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. In *IEEE Transactions on Speech and Audio Processing*, pages vol. 8 (1), 11–23.

Ljolje, A. and Fallside, F. (1986). Synthesis of natural sounding pitch contours in isolated utterances using hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(5):1074 – 1080.

Lleida, E. (2005). Librería de comunicaciones basada en paquetes XML: Dihana v2.0. Technical report.

Lleida, E., Segarra, E., Torres, M. I., and Macías-Guarasa, J. (2006). EDECÁN: sistEma de Diálogo multidominio con adaptación al contExto aCústico y de AplicaciónN. In *IV Jornadas en Tecnologia del Habla*, pages 291–296, Zaragoza, Spain.

Miguel, A., Galiano, M., Granell, R., Hurtado, L., and Sanchis, E. (2003). La plataforma de adquisición de diálogos en el proyecto DIHANA. In *Actas del XIX Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural*, pages 343–344, Alcala de Henares (España).

Minker, W. (1999). Stocastically-based semantic analysis. In *Kluwer Academic Publishers*, Boston, USA.

Moreno, A., Poch, D., Bonafonte, A., Lleida, E., Llisterri, J., Marino, J., and Nadeu, C. (1993). Albayzin speech database: Design of the phonetic corpus. In *Third European Conference on Speech Communication and Technology*. ISCA.

Pietquin, O. and Beaufort, R. (2005). Comparing ASR modeling methods for spoken dialogue simulation and optimal strategy learning. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech/Eurospeech'05)*, Lisbon (Portugal).

Pietquin, O. and Dutoit, T. (2005). A probabilistic framework for dialog simulation and optimal strategy learning. In *IEEE Transactions on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog*.

Pérez, G., Amores, G., and Manchón, P. (2006). A multimodal architecture for home control by disabled users. In *Procs. of IEEE/ACL Workshop on Spoken Language Technology (SLT)*, Aruba.

Quarteroni, S., González, M., Riccardi, G., and Varges, S. (2010). Combining user intention and error modeling for statistical dialog simulators. In *Procs. of InterSpeech'10*, pages 3022–3025, Makuhari Chiba, Japan.

Quarteroni, S., Riccardi, G., and Dinarelli, M. (2009). What's In An Ontology For Spoken Language Understanding. In *Interspeech*, Brighton, U.K.

Rabiner, L., Juang, B., and Lee, C. (1996). An overview of automatic speech recognition. In Publishers, K. A., editor, *Automatic Speech and speaker Recognition: Advanced Topic*, pages 1–30.

Raux, A., .Mehta, N., Ramachandran, D., and Gupta, R. (2010). Dynamic language modeling using bayesian networks for spoken dialog systems. In *Procs. of InterSpeech'10*, pages 3030–3033, Makuhari Chiba, Japan.

Raymond, C., Bechet, F., De Mori, R., and Damnati, G. (2006). On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48:288–304.

Rose, R. and Lleida, E. (1997). Speech recognition using automatically derived acoustic baseforms. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1271 –1274 vol.2.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. (2007). Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 149–152, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schatzmann, J., Weilhammer, K., Stuttle, M., and Young, S. (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.*, 21:97–126.

Scheffler, K. and Young, S. (2000). Probabilistic simulation of human-machine dialogues. In *Proc IEEE ICASSP*, Istanbul (Turkey).

Scheffler, K. and Young, S. (2001). Corpus-based Dialogue Simulation for Automatic Strategy Learning and Evaluation. In *Proc NAACL-2001 Workshop on Adaptation in Dialogue Systems*, Pittsburgh, USA.

Schlangen, D. and Skantze, G. (2009). A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics.

Segarra, E., Hurtado, L., Gómez, J., García, F., Planells, J., Pastor, J., Ortega, L., Calvo, M., and Sanchis, E. (2010). A prototype of spoken dialog system based on statistical models. In *Proceedings of FALA 2010*, Vigo, Spain.

Segarra, E., Sanchis, E., García, F., and Hurtado, L. (2002). Extracting semantic information through automatic learning techniques. In *International Journal of Pattern Recognition and Artificial Intelligence*, pages 16(3):301–307, Salt Lake City (USA).

Thomson, B., Schatzmann, J., and Young, S. (2008). Bayesian update of dialogue state for robust dialogue systems.

Thomson, B. and Young, S. (2009). Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. In *Computer Speech & Language*.

Torres, F., Sanchis, E., and Segarra, E. (2003). Development of a stochastic dialog manager driven by semantics. In *Proc. of EuroSpeech'03*, pages 605–608.

Williams, J. (2008). The best of both worlds : Unifying conventional dialog systems and pomdps. In *International Conference on Speech and Language Processing*.

Williams, J. and Young, S. (2007). Partially Observable Markov Decision Processes for Spoken Dialog Systems. In *Computer Speech and Language 21(2)*, pages 393–422.

Young, S. (2000). Probabilistic Methods in Spoken Dialogue Systems. In *Philosophical Trans Royal Society (Series A)*, pages 358(1769): 1389–1402.

Young, S. (2002). The Statistical Approach to the Design of Spoken Dialogue Systems. Technical report, Cambridge University Engineering Department.

Young, S. (2006). Using POMDPs for Dialog Management. In *Proc. of IEEE-ACL Workshop on Spoken Language Technology (SLT 2006)*, Aruba.

Young, S., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150 – 174.

Yu, K. and Young, S. (2010). Continuous f0 modelling for hmm based statistical parametric speech synthesis. *Audio, Speech, and Language Processing, IEEE Transactions on*, PP(99):1.