

# Sistema de Diálogo Basado en VoiceXML para Proporcionar Información de Viajes en Tren

R. López-Cózar <sup>1</sup>, R. Granell <sup>2</sup>

<sup>1</sup>Dpto. Lenguajes y Sistemas Informáticos, E.T.S. Ingeniería Informática  
18071 Universidad de Granada, Tel.: +34 958 240579, Fax: +34 958 243179  
E-mail: rlopezc@ugr.es

<sup>2</sup>Dpto. Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia, Tel.: +34 96 387 9722, Fax: +34 96 387 7395  
E-mail: rgranell@dsic.upv.es

**Resumen:** En este artículo se presenta el resultado de nuestro trabajo actual en el proyecto DIHANA, consistente en implementar un sistema de diálogo basado en la tecnología VoiceXML que permita a los usuarios acceder telefónicamente a información de viajes en tren (p. e. precios de billetes, horarios de salida y llegada, etc.). En el artículo se presentan las principales características del sistema desarrollado, se describen las soluciones que aportamos en VoiceXML para hacer frente a los diversos problemas de implementación del sistema, y se aporta nuestra experiencia personal al usar este lenguaje en lugar de otros lenguajes de propósito general para realizar la implementación.

**Palabras clave:** Sistemas de diálogo, VoiceXML, reconocimiento automático del habla, procesamiento de lenguaje natural, gestión de diálogo, síntesis de habla.

**Abstract:** This paper presents our recent work in the DIHANA project, concerned with setting up a dialogue system based on VoiceXML to provide train travel information on the phone (e. g. ticket prices, departure and arrival timetables, etc.). The paper addresses the main features of the system developed and comments our solutions in terms of VoiceXML to deal with the typical problems of implementing a dialogue system. Finally, it addresses our own experience using VoiceXML instead of other general-purpose languages for the setting up.

**Keywords:** Dialogue systems, VoiceXML, automatic speech recognition, natural language processing, dialogue management, speech synthesis.

## 1 Introducción

Los sistemas de diálogo son programas informáticos que se diseñan con la finalidad de emular a un ser humano en un diálogo oral con otra persona. Actualmente, diversas empresas e instituciones usan estos sistemas para proporcionar información y otros servicios de forma automática, como por ejemplo, información y reserva de viajes en avión (Seneff y Polifroni, 2000), información meteorológica (Zue et al. 2000), compra de comida rápida (López-Cózar et al. 1997), información y reserva de viajes en tren (Billi et al. 1997), etc.

En los años ochenta, dada la inexistencia de herramientas de desarrollo, los implementadores de estos sistemas debían procesar a nivel de señal la voz proveniente de los usuarios, controlar el flujo de la

interacción y los posibles eventos que podían producirse, etc. En los años noventa aparecen diversas herramientas de desarrollo, como por ejemplo HTK (*Hidden Markov Model Toolkit*) (Hain et al., 1999), que facilitan la implementación de tales sistemas. En los años noventa aparecen en el mercado servidores web con capacidad de soportar la voz humana, propiciando la aparición de lenguajes basados en etiquetas que facilitan el desarrollo de sistemas capaces de interactuar oralmente con las páginas web. VoiceXML es uno de estos lenguajes, diseñado para aprovechar además las ventajas aportadas por los sistemas de transmisión de información en Internet (<http://www.voicexml.org>). Entre otras características, este lenguaje posibilita usar ficheros de audio digitalizados, reconocimiento automático del habla,

entradas mediante DTMF (*Dial Tone Multi Frequency*), grabación de entradas orales, gestión de terminales telefónicos, conversaciones de iniciativa mixta y conversión texto-habla. A principios de 2001, VoiceXML fue considerado por el W3C (World Wide Web Consortium) el lenguaje estándar basado en etiquetas para crear este tipo de sistemas. Poco después, el VBWG (*Voice Browser Working Group*), un subgrupo de trabajo del W3C dedicado a la evolución del estándar, comenzó a trabajar en la versión 2.0, cuya versión inicial apareció en octubre de 2001.

A modo ilustrativo, la Figura 1 muestra un breve fragmento de código escrito en VoiceXML para solicitar el número de teléfono del usuario, reconocerlo mediante un proceso de reconocimiento automático del habla (RAH) (Rabiner y Juang, 1993), almacenarlo en una determinada variable para su posterior uso y generar un mensaje de ayuda a petición del usuario. Como se puede observar en el ejemplo, el lenguaje presenta la ventaja de aislar al implementador de las cuestiones de bajo nivel relacionadas con el RAH y la síntesis de habla.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
  <form id="datos_usuario">
    <field name="telefono" type="digits">
      <prompt>Diga su número de teléfono
        dígito a dígito </prompt>
      <help> Debe pronunciar el número
        dígito a dígito, por ejemplo, 9, 5,
        8, 1, 3, 1, 4, 2, 6 </help>
    </field>
  </form>
</vxml>
```

Figura 1. Ejemplo de código VoiceXML

Respecto al RAH, el implementador sólo ha de especificar qué tipo de datos desea obtener del usuario y gestionar a alto nivel algunos eventos que se puedan producir durante dicho proceso (p. e. el usuario solicita ayuda o no responde en un determinado intervalo de tiempo, etc.). Respecto a la síntesis de habla, el implementador sólo ha de especificar el mensaje a generar, en forma de texto o mediante un fichero de voz pregrabado.

Tras esta breve introducción, el resto del artículo está estructurado de la siguiente forma. La sección 2 presenta brevemente el proyecto DIHANA, describiendo los corpus de diálogos usados en el proceso de creación del sistema de diálogo implementado. La sección 3 presenta dicho sistema, centrándose en su estructura modular y sus principales características. Asimismo, muestra un diálogo de ejemplo realizado en condiciones de laboratorio. Finalmente, la sección 4 presenta las conclusiones y comenta algunas líneas de trabajo futuro.

## 2 Proyecto DIHANA

El proyecto DIHANA (TIC2002/04103-C03), aprobado por la CICYT para el periodo 2003-06, tiene como finalidad desarrollar un sistema de diálogo para el acceso a la información mediante habla espontánea en diferentes entornos. El dominio de aplicación del sistema es la información de viajes en tren, tarea abordada previamente en otros proyectos de investigación como por ejemplo TABA (Aust et al. 1995), ARISE (Den Os et al. 1999) y BASURDE (Bonafonte et al. 2000). Se trata de un dominio de aplicación relativamente sencillo pues está bien delimitado semánticamente, lo que facilita el proceso de comprensión de las frases (Sanchís et al. 2000, Segarra et al. 2000). Además, el vocabulario es estable, regular y no contiene palabras en otros idiomas (lo que facilita el proceso de RAH). Asimismo, la estructura del diálogo es relativamente sencilla pues se basa en datos concretos que deben ser proporcionados por los usuarios para obtener la información deseada (p. e. nombres de ciudades de origen y destino, día de la semana en que desean viajar, etc.).

### 2.1 Corpus de diálogos

Para desarrollar el sistema de diálogo presentado en este artículo nos hemos basado en dos corpus de diálogos creados previamente en el proyecto BASURDE. Por una parte, un corpus consistente en 200 diálogos persona-persona en los que usuarios reales de este tipo de servicio solicitan información interactuando con operadores reales. El análisis de este corpus

nos ha permitido obtener información muy variada que incluye fenómenos propios del habla espontánea, como por ejemplo, interrupciones, correcciones (p. e. “el viernes – el sábado”), pausas rellenas (*filled pauses*, p. e. “eh”), falsos comienzos (*false starts*, p. e. “quiero ... me gustaría ...”), repeticiones de palabras (p. e. “el – el”), frases incorrectas gramaticalmente, etc. La información obtenida muestra que el habla espontánea constituye una forma de expresión muy compleja, incluso en escenarios poco complejos (en principio) como el que nos ocupa, excediendo en ocasiones las posibilidades de los sistemas actuales de RAH y comprensión del habla.

Teniendo en cuenta que los usuarios probablemente se comporten de forma distinta cuando interactúen con una máquina (afectando en consecuencia a su léxico y estructuras gramaticales), hemos usado un segundo corpus de diálogos con la finalidad de refinar los modelos obtenidos del primer corpus. Este segundo corpus contiene 215 diálogos persona-sistema obtenidos mediante la técnica conocida como Mago de Oz (Maulsby et al. 1993).

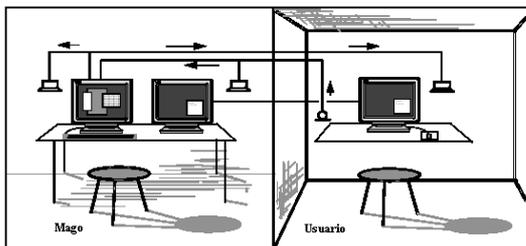


Figura 2. Escenario de la técnica Mago de Oz

Según esta técnica (Figura 2), una persona (Mago) simula el comportamiento del sistema haciendo creer a los usuarios que éstos realmente están interactuando con un ordenador. Los modelos obtenidos como resultado del análisis de ambos corpus nos han permitido conocer:

- El vocabulario de la aplicación.
- Los tipos de frases que probablemente usarán los usuarios para comunicarse con el sistema de diálogo.
- Las estrategias de diálogo que el sistema deben utilizar para interactuar con los usuarios.
- Los tipos de frases que éste debe generar como respuestas.

### 3 Sistema de diálogo basado en VoiceXML

En esta sección presentamos el resultado de nuestro trabajo en el proyecto DIHANA, relacionado con el estudio y uso de la tecnología VoiceXML con objeto de desarrollar un sistema de diálogo para una aplicación concreta. La Figura 3 muestra la estructura modular del sistema que hemos implementado usando un toolkit de desarrollo de este tipo de sistemas.

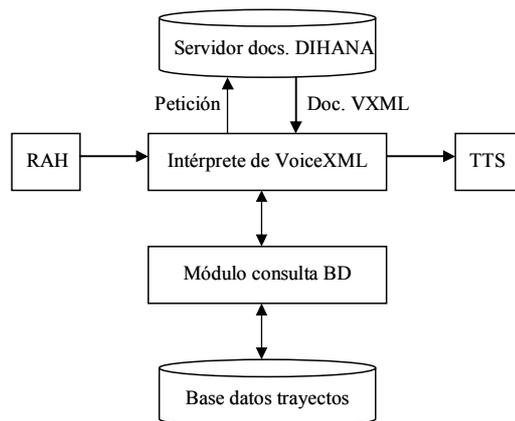


Figura 3. Sistema de diálogo basado en VoiceXML

El sistema está funcionando en un ordenador PC que, en el momento de escribir este artículo, no se encuentra conectado a la línea telefónica, por lo que la interacción ha de realizarse usando el micrófono y los altavoces.

#### 3.1 Interfaz de Entrada/Salida

La interfaz de Entrada/Salida del sistema está constituida por los módulos de RAH y conversión texto-habla (Text-To-Speech, TTS). En la versión actual del sistema, dichos módulos son sistemas de RAH y TTS desarrollados por IBM, concretamente incluidos en el SDK 3.1 de WebSphere Voice Server, aunque nuestra intención es incorporar próximamente otros módulos desarrollados en el marco del proyecto.

Para los procesos de RAH y comprensión del habla el sistema utiliza gramáticas en formato JSGF (Java Speech Grammar Format) (Figura 4). Una gramática de este tipo tiene la ventaja de combinar ambos procesos en un solo paso, ya que por una parte, determina la

secuencia de palabras reconocibles, y por otra, indica cómo deben ser interpretadas semánticamente determinadas secuencias de palabras.

```
#JSGF V1.0;

grammar de_a;

public <de_a> = [<deseo>]
[<viajar> <ciudad> {this.ciudadDestino=$ciudad}]
[<procede> <ciudad> {this.ciudadOrigen=$ciudad}];

<deseo> = quiero [saber] | me gustaría [saber] | quería
| yo quería | necesito | yo necesito | tengo que;

<viajar> = ir a | viajar a | para ir a | para viajar a;

<ciudad> = jaén | córdoba | sevilla | huelva | cádiz |
málaga | granada | almería | valencia | alicante |
castellón | barcelona | madrid;

<procede> = de | desde | salir desde | saliendo desde;
```

Figura 4. Gramática en formato JSGF

Por ejemplo, la regla “<viajar> <ciudad> {this.ciudadDestino=\$ciudad}” indica al intérprete de VoiceXML que si el nombre de una ciudad está precedido por un constituyente de la clase “viajar” (p. e. “para viajar a”), entonces dicha ciudad debe ser “comprendida” como la ciudad a la que el usuario desea viajar. Análogamente, la regla “<procede> <ciudad> {this.ciudadOrigen=\$ciudad}” indica que si el nombre de la ciudad está precedido por un constituyente de la clase “procedencia” (p. e. “saliendo desde”) entonces dicha ciudad debe ser “comprendida” como la ciudad desde la que el usuario desea iniciar el viaje. El resultado de la comprensión no es la creación de los frames semánticos (Allen, 1995) típicos de un gran número de sistemas de diálogo, sino la asignación de los datos obtenidos a los campos (*fields*) de un formulario (*form*) de un documento de la aplicación.

En cuanto a la salida, el sistema utiliza la técnica de conversión texto-habla mediante la que cualquier frase en formato de texto puede ser transformada a voz. Para generar las frases en formato de texto (respuestas del sistema) se usa un conjunto de patrones que son instanciados con los datos correspondientes. Por ejemplo, la Figura 5 muestra el patrón utilizado para solicitar la confirmación de los datos

obtenidos del usuario antes de efectuar la consulta a la base de datos.

```
<!-- comprobación ciudades y día -->
<field name="confirmacion" type="boolean"
modal="true">
  <prompt>
    Quiere ir a <value expr="ciudadDestino"/>
    saliendo desde <value expr="ciudadOrigen"/>
    el <value expr="diaSemana"/>? Sí o no.
  </prompt>
  ...
</field>
```

Figura 5. Ejemplo de patrón para generación de frases en VoiceXML

Para instanciar este patrón sólo son necesarios tres datos (ciudadDestino, ciudadOrigen y diaSemana), generándose por ejemplo la frase: “¿Quiere ir a Barcelona saliendo desde Madrid el viernes?”.

### 3.2 Base de datos y módulo de consulta a la BD

La base de datos a la que accede el sistema para proporcionar información a los usuarios está implementada mediante un fichero de MS Access en el que existe un registro por cada trayecto de tren que une directamente dos ciudades. La información almacenada está basada en el sistema ferroviario español, incluyendo nombres de ciudades de origen y destino, días (de la semana) de salida y llegada, horas de salida y llegada, tipos de trenes, clases y precios de billetes, etc.

El módulo de consulta a la BD está implementado mediante un script escrito en lenguaje Perl que se encarga de efectuar las consultas SQL a la base de datos y generar dinámicamente nuevos documentos VoiceXML con los datos obtenidos. Además, este script se encarga de transformar a lenguaje natural determinada información que se encuentra codificada en la base de datos; por ejemplo, los días de los trayectos están almacenados en el formato X-Y, donde X hace referencia al día de la semana (1 = domingo, 2 = martes, etc.) e Y hace referencia a la semana dentro del mes (1 = primera semana, 2 = segunda semana, etc.).

### 3.3 Servidor de documentos DIHANA

El servidor de documentos DIHANA procesa las peticiones enviadas por el intérprete de VoiceXML y proporciona como respuesta documentos VoiceXML estáticos (Abbot, 2002) que constituyen la aplicación. Tales documentos son usados por el intérprete para gestionar el diálogo con los usuarios, consistente básicamente en solicitar datos necesarios para efectuar consultas a la base de datos, realizar confirmaciones de datos poco fiables y proporcionar información con los datos obtenidos de las consultas.

La aplicación está implementada mediante un documento raíz, que contiene diversas variables globales y gramáticas a nivel de aplicación, y varios documentos hoja dependientes del documento raíz (p. e. existe un documento para gestionar las consultas de horarios, otro para gestionar las consultas de precios, etc.).

### 3.4 Intérprete de VoiceXML

El intérprete de VoiceXML procesa los comandos (etiquetas) de los documentos VoiceXML obtenidos del servidor de documentos, genera prompts y procesa las respuestas del usuario, ejecutando la lógica de la aplicación. En dicha lógica se han considerado tres aspectos básicos en cualquier sistema de diálogo: estrategia de interacción, estrategia de confirmación y gestión de eventos.

#### 3.4.1 Estrategia de interacción

Para lograr una alta efectividad de los diálogos sin restringir excesivamente la libertad de interacción de los usuarios, hemos implementado una estrategia de interacción mixta (Polifroni et al. 2003). Ello permite que el usuario pueda efectuar consultas proporcionando un número variable de datos (p. e. ciudad destino, ciudad destino + ciudad origen, ciudad destino + ciudad origen + día de la semana, etc.). Además, este tipo de estrategia permite que cuando el usuario se encuentre en un determinado estado del diálogo, pueda cambiar el foco actual de la conversación e iniciar un subdiálogo distinto. Para implementar esta estrategia usamos,

por una parte, la etiqueta <initial>, que permite generar prompts poco restrictivos, del tipo “¿Qué desea?, ¿Cómo puedo ayudarlo?”, y por otra, usamos gramáticas a nivel de aplicación, es decir, gramáticas que están activas en cualquier estado del diálogo, independientemente del prompt generado por el sistema en un momento dado.

La interacción se rige mediante reglas que determinan el siguiente estado a visitar en los formularios, y que son utilizadas por el algoritmo de interpretación de formularios FIA (*Form Interpretation Algorithm*) de VoiceXML.

#### 3.4.2 Estrategia de confirmación

Para gestionar las confirmaciones de los datos proporcionados por los usuarios, usamos los valores de confianza asignados a las palabras reconocidas por el módulo de RAH, que son números reales en el rango (0,1). El sistema utiliza confirmaciones explícitas e implícitas (López-Cózar et al. 2000). Las confirmaciones explícitas se usan para confirmar únicamente aquellas palabras cuyos valores de confianza no superan el valor del umbral de confianza. Un ejemplo de este tipo de confirmación es el siguiente: “¿Ha dicho que quiere viajar a Barcelona?”. En cambio, las confirmaciones implícitas se utilizan para confirmar datos a la vez que se solicitan otros datos. Un ejemplo de este tipo de confirmación es el siguiente: “De acuerdo, destino Barcelona. ¿Desde qué ciudad desea salir?”. Combinando ambos tipos de confirmaciones (explícitas e implícitas) evitamos sobrecargar el diálogo con confirmaciones, lo que provocaría el desagrado de los usuarios.

#### 3.4.3 Gestión de eventos

El sistema también incorpora el código necesario para gestionar los diversos tipos de eventos que se pueden producir durante el diálogo (p. e. el usuario solicita ayuda o no responde a un determinado prompt del sistema, el módulo de RAH no proporciona ninguna salida aceptable, etc.). Para estas situaciones VoiceXML proporciona un mecanismo de tratamiento general, liberando así al programador de la

necesidad de definirlo; no obstante, es posible modificar el tratamiento de los eventos incluyendo las etiquetas correspondientes en el código de la aplicación.

En la implementación del sistema hemos usado las etiquetas <help>, <noinput> y <match> para gestionar los eventos de forma sensible al contexto. Así, cuando el usuario solicita ayuda, no proporciona ninguna entrada o no se reconoce con suficiente confianza la frase que pronuncia, el sistema proporciona una respuesta u otra dependiendo del estado del diálogo en que se produce dicho evento. Por ejemplo, si al inicio del diálogo el usuario solicita ayuda, el sistema proporciona el mensaje siguiente: “Puede solicitar información sobre horarios, precios, servicios de los trenes, ...”; en cambio, si solicita la ayuda cuando el sistema le pide que indique la ciudad de destino, el mensaje de ayuda es el siguiente: “Debe decir el nombre de la ciudad a la que desea viajar, por ejemplo, Barcelona”.

Asimismo, tenemos previsto que la respuesta del sistema ante el evento producido cuando el usuario no proporcione respuesta (o ésta no pueda ser reconocida) varíe en función de la repetición de dicho evento. Por ejemplo, en caso de no recibir una respuesta aceptable, el mensaje puede ser: “Por favor, diga claramente el nombre de la ciudad a la que desea viajar”, y tras el siguiente intento para obtener ese mismo dato se puede generar el mensaje: “Un momento por favor, le pasamos con un operador”, transfiriendo seguidamente la llamada entrante a un determinado número de teléfono.

### 3.5 Diálogo de ejemplo

Finalmente, mostramos un ejemplo de diálogo realizado en condiciones de laboratorio (con escaso ruido), usando el micrófono y los altavoces (S = sistema, U = usuario).

S1: Buenas tardes. Bienvenido a X. ¿Cómo puedo ayudarle?  
U1: sí, quería saber qué trenes hay para ir a Barcelona este viernes  
S2: ¿desde qué ciudad desea salir?  
U2: desde Madrid

S3: hay trenes que van a Barcelona saliendo desde Madrid el viernes a las siguientes horas: 9.00, 10.55, 14.00, 15.45, 22.00 y 23.00. ¿Desea algo más?

U3: ¿puede repetir?

S4: hay trenes que van a Barcelona saliendo desde Madrid el viernes a las siguientes horas: 9.00, 10.55, 14.00, 15.45, 22.00 y 23.00. ¿Desea algo más?

U4: sí, ¿cuánto vale el billete en el de las 14 horas?

S5: ¿en qué clase desea viajar? preferente, turista o coche cama.

U5: en turista

S6: en clase turista cuesta 40.50 euros. ¿Algo más?

U6: ¿cuánto tarda?

S7: 7 horas y 3 minutos. ¿Desea algo más?

U7: no, muchas gracias.

S8: de acuerdo, muchas gracias por usar nuestro servicio. Buenas tardes.

Figura 6. Diálogo de ejemplo

Como se puede observar en este diálogo, la interacción se realiza eficientemente pues los valores de confianza de las palabras reconocidas superan el valor del umbral de confianza, y por consiguiente, no son necesarios turnos del diálogo destinados a realizar confirmaciones explícitas de datos.

También se puede observar en los turnos S3, S4 y S6 que el sistema incluye en sus respuestas los datos utilizados para realizar las consultas a la base de datos. De esta forma se evita que se pueda proporcionar información errónea al usuario a causa de las llamadas *aceptaciones falsas*, que son inevitables. Tales aceptaciones se producen si los valores de confianza de palabras incorrectamente reconocidas superan el valor del umbral de confianza. Dado que estas palabras no son confirmadas explícitamente, es posible que se realicen consultas a la base de datos usando datos erróneos (por ejemplo, el usuario puede solicitar horarios para el trayecto Madrid-Barcelona, y el sistema puede proporcionar los horarios para el trayecto Madrid-Gerona). Incluyendo en las respuestas los datos usados para consultar la base de datos, el usuario puede detectar tales errores en caso de producirse (es decir, sabrá que la información que recibe no es la que ha

solicitado), pudiendo realizar de nuevo la consulta.

#### 4 Conclusiones y trabajo futuro

En este artículo se ha presentado el resultado de nuestro trabajo en el proyecto DIHANA relacionado con el estudio y aplicación de la tecnología VoiceXML para implementar un sistema de diálogo que proporcione información de viajes en tren. Esta tecnología ha sido diseñada para simplificar la implementación de sistemas de diálogo, aislando al programador de las cuestiones de bajo nivel relacionadas con el RAH y la síntesis del habla. La implementación se realiza usando etiquetas similares a las de HTML, intuitivas y fáciles de recordar. No obstante, nuestra experiencia personal tras implementar el sistema descrito en este artículo es que el proceso de implementación no es tan fácil ni tan inmediato como cabría esperar, pues difiere bastante en algunas cuestiones respecto a la metodología de programación usada en otros lenguajes de uso general, como por ejemplo C. Una diferencia importante radica en que algunas etiquetas son altamente *contextuales*, es decir, se pueden usar únicamente dentro del ámbito de otras, lo que causa cierta confusión inicialmente. Por otra parte, usando VoiceXML no se tiene el control *de grano fino* proporcionado por los lenguajes de propósito general, restando potencial al programador.

No obstante, a pesar de estas desventajas, consideramos que VoiceXML constituye una herramienta muy útil para gestionar la interacción mediante diálogos basados en transacciones simples (p. e. relacionados con viajes en avión o tren, información meteorológica, venta de productos, etc.), gestionando fácilmente algunos de los diversos eventos que se pueden producir durante la interacción (p. e. el usuario no contesta o el sistema no se comprende lo que dice, etc.).

Algunas líneas de trabajo futuro relacionadas con el desarrollo del sistema son las siguientes:

- Configuración de un ordenador que actúe *voice gateway*, en el que se instalará el interprete de VoiceXML y el hardware telefónico, permitiéndose

así la recepción de llamadas de los usuarios.

- Modificación del módulo de consulta a la base de datos, para que las consultas se efectúen en una página web en la que la información esté siempre actualizada.
- Implementación de estrategias de confirmación e interacción adaptativas. Tenemos previsto implementar estrategias que tengan en cuenta el éxito del diálogo a lo largo de la interacción, para que el sistema pueda adaptar su funcionamiento de forma automática. De esta forma, podrá usar confirmaciones explícitas en lugar de implícitas si el usuario indica repetidamente la existencia de errores. Asimismo, podrá limitar la libertad de interacción del usuario (deshabilitando las gramáticas a nivel de formulario o aplicación) para obtener en cada momento un único dato. Creemos que usando este tipo de estrategias el sistema podrá gestionar más eficientemente el diálogo en los diversos entornos en que éste se podrá llevar a cabo (p. e. usando un teléfono fijo o móvil, hablando desde casa o desde un vehículo en movimiento, etc.).

#### Bibliografía

- Abbott, K. R. 2002. Voice enabling web applications. VoiceXML and beyond. a! press
- Allen, 1995. Natural language understanding. The Benjamin/Cummings Publishing Company Inc.
- Aust, H., Oerder, M., Seide, F., Steinbiss, V. 1995. The Phillips automatic train timetable information system. Speech Communication, 17, pág. 249-262
- Billi, R., Castagneri, G., Danielli, M. 1997. Field trial evaluations of two different information inquiry systems. Speech Communication, 23, 1-2, pág. 83-93
- Bonafonte, A., Aibar, P., Castell, E., Lleida, E., Mariño, J. B., Sanchís, E., Torres, M. I. 2000. Desarrollo de un sistema de diálogo oral en dominios restringidos. I Jornadas en Tecnología del Habla, Sevilla

- Den Os, E., Boves, L., Lamel, L., Baggia, P. Overview of the ARISE Project. 1999. Proc. Eurospeech, pág. 1527-1530
- Hain T., Woodland P.C., Niesler T.R., Whittaker E.W.D. 1999. The 1998 HTK System for Transcription of Conversational Telephone Speech; Proc. International Conference on Acoustics, Speech and Signal Processing
- López-Cózar, R., García, P., Díaz, J., Rubio, A. J. 1997. A voice activated dialogue system for fast-food restaurant applications. Proc. Eurospeech, pág. 1783-1786
- López-Cózar R. 2000. Test de estrategias de diálogo en un sistema conversacional. I Jornadas en Tecnología del Habla, Sevilla
- Maulsby, D., Greenberg, S., Mander, R. 1993. 1993. Applying the Wizard of Oz technique – To the study. Proc. InterCHI, pp. 277-284
- Polifroni, J., Chung, G., Seneff, S. 2003. Towards the automatic generation of mixed-initiative dialogue systems from web content. Proc. Eurospeech, pág. 193-196
- Rabiner L. R., Juang B. H. 1993. Fundamentals of Speech Recognition. Prentice-Hall
- Sanchís, E., Segarra, E., Galiano, M., García, F., Hurtado, L. 2000. Modelización de la comprensión mediante técnicas de aprendizaje automático. I Jornadas en Tecnología del Habla, Sevilla
- Segarra, E., Sanchís, E., Galiano, I., García, F., Hurtado, L. 2002. Extracting semantic information through automatic learning. International Journal of Pattern Recognition and Artificial Intelligence, 16(3), pág. 301–307
- Seneff, S., Polifroni, J. 2000. Dialogue management in the Mercury flight reservation system. Proc. ANLP-NAACL 2000 Satellite Workshop, pág. 1-6
- Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T., Hetherington, L. 2000. Jupiter: A telephone-based conversational interface for weather information. IEEE Trans. on Speech and Audio Proc., 8(1), pág. 85-96