ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

**UNIVERSIDAD DE GRANADA**

**Trabajo Fin de Máster**

# Técnicas de Evaluación de Sistemas de Diálogo: Aplicación al sistema *Mayordomo*

**Autora**

Nieves Ábalos Serrano

**Tutores**

Ramón López-Cózar Delgado

Zoraida Callejas Carrión

Máster en Desarrollo de Software, 2009/2010

Dpto. Lenguajes y Sistemas Informáticos

**Granada, Diciembre 2010**

# Índice general

Índice general ..... 3
1. Motivación de este trabajo ..... 5
1.1 Principales contribuciones ..... 5
2. Introducción ..... 7
2.1 Sistemas de diálogo ..... 7
2.2 Evaluación de sistemas de diálogo ..... 9
2.2.1 Medidas de evaluación ..... 9
2.2.2 Simulador de usuarios ..... 10
2.3 Tecnologías del habla ..... 11
2.3.1 Windows Speech Recognition ..... 11
2.3.2 Herramienta para desarrolladores ..... 15
3. Sistema de diálogo *Mayordomo* ..... 25
3.1 Introducción ..... 25
3.2 Interacción mediante habla ..... 25
3.2.1 Reconocimiento automático de habla (RAH) ..... 26
3.2.2 Comprensión de habla ..... 27
3.2.3 Gestión del diálogo ..... 29
3.2.4 Generación de frases ..... 31
3.2.5 Síntesis de habla ..... 31
3.3 Interacción gráfica ..... 31
3.4 Diseño actual ..... 32
3.4.1 Electrodomésticos, atributos, valores y acciones. ..... 32
3.5 Clases ..... 37
3.5.1 Electrodomestico.h ..... 37
3.5.2 Habitacion.h ..... 38
3.5.3 Hogar.h ..... 39
3.6 Archivos de configuración ..... 40
3.6.1 Archivo de configuración del hogar ..... 40
3.6.2 Archivos de configuración de los electrodomésticos ..... 41
4. Gestor del corpus ..... 43
4.1 Introducción ..... 43
4.1.1 Obtención del corpus ..... 43

4.1.2 Transcriptor Ortográfico Automático ..... 48
4.1.3 Transcriptor Semántico Automático ..... 49
4.1.4 Gestor del Corpus ..... 50
4.1.5 Simulación de usuarios ..... 52
4.2 Análisis ..... 59
4.2.1 Diagrama de clases ..... 59
4.3 Implementación ..... 59
4.3.1 Librerías ..... 59
4.3.2 Estructura wordAccuracy ..... 60
4.3.3 GestorHogar ..... 60
4.3.4 Semantica ..... 62
4.3.5 Objetivo ..... 65
4.3.6 SimuladorUsuario ..... 65
4.3.7 GestorCorpus ..... 68
4.4 Ficheros de resultados ..... 77
4.4.1 Gestor del Corpus ..... 77
4.4.2 Simulador de Usuarios ..... 81
5. Experimentos realizados ..... 87
5.1 Perfiles de Reconocimiento de Voz ..... 87
5.2 Análisis del corpus de frases ..... 90
5.2.1 Escenario 1 ..... 91
5.2.2 Escenario 2 ..... 91
5.2.3 Corpus de frases completo ..... 92
5.3 Simulador de usuarios ..... 93
6. Conclusiones ..... 99
7. Trabajo futuro ..... 101
8. Apéndices ..... 103
A) Glosario de términos ..... 103
B) Windows Speech Recognition (Windows 7) ..... 104
10. Bibliografía ..... 113
11. Índices de tablas y figuras ..... 115
11.1 Índice de tablas ..... 115
11.2 Índice de figuras ..... 116

# 1. Motivación de este trabajo

La principal motivación de este Trabajo Fin de Máster es realizar un estudio en el campo de la evaluación de sistemas de diálogo hablado (Spoken Dialogue Systems, SDS), y aplicarlo al desarrollo de un sistema de diálogo adaptando las tecnologías estudiadas.

Los SDS son sistemas software complejos que conjugan diversas tecnologías, lo que hace muy complicada su evaluación, que tradicionalmente se haya llevado a cabo como pruebas de módulos individuales. Sin embargo, se hace necesario utilizar técnicas de evaluación global en que se mida el rendimiento y la usabilidad del sistema como un todo que integre cada uno de los módulos de su arquitectura. La sección 2 presenta los conceptos clave para comprender la complejidad de la evaluación de estos sistemas.

A partir del estudio realizado acerca de los criterios y métodos de evaluación que se emplean en la actualidad para los sistemas de diálogo, hemos implementado una metodología de evaluación para nuestro sistema de diálogo multimodal *Mayordomo*. En la sección 3, se presenta el sistema *Mayordomo* y se explican los detalles más relevantes de su implementación. En la Sección 4, presentamos las herramientas desarrolladas para llevar a cabo la evaluación de sistemas de diálogo, prestando especial atención al gestor del corpus y simulador de usuario. A continuación, en la sección 5, detallamos los experimentos realizados con estas herramientas en su aplicación particular al sistema *Mayordomo*. Para finalizar, en las secciones 6 y 7 respectivamente, se desarrollan las conclusiones obtenidas y se mencionan los posibles puntos a tratar en trabajos futuros con el fin de obtener una metodología de evaluación que permita el desarrollo de sistemas de diálogo más robustos y eficientes.

## 1.1 Principales contribuciones

Como fruto de este análisis se ha desarrollado un conjunto de herramientas para llevar a cabo dicha evaluación, que han sido aplicadas al caso específico de un sistema desarrollado en mi Proyecto Fin de Carrera, llamado *Mayordomo*.

En concreto, las aportaciones realizadas permiten llevar a cabo la evaluación de una manera sencilla, usando dos enfoques:

- Evaluación global del sistema. Para realizar este tipo de evaluación se ha desarrollado una herramienta de simulación de usuarios.
- Evaluación del reconocedor de habla, del procesador del lenguaje natural y del gestor del diálogo. Para realizar este tipo de evaluación se ha desarrollado un gestor del corpus de frases, que consta de un transcriptor ortográfico, transcriptor semántico, y una serie de herramientas secundarias para calcular medidas estadísticas.

Gracias a estas herramientas, podemos obtener medidas que calculan la exactitud de palabras (Word Accuracy), la exactitud de conceptos (Keyword Accuracy), la tasa de comprensión de frases (Sentence Understanding), la tasa de reconocimiento de frases

(Sentence Recognition), la tasa de tareas completadas (Task Completion) y la tasa de recuperación implícita (Implicit Recovery).

Estas medidas nos permiten obtener resultados experimentales de los cuales poder obtener conclusiones, como por ejemplo, qué módulos del sistema de diálogo debemos mejorar, o por qué es importante reentrenar los perfiles de voz de los reconocedores de habla, entre otras.

# 2. Introducción

## 2.1 Sistemas de diálogo

Los sistemas de diálogo o sistemas conversacionales (SLS, Spoken Language Systems) son una tecnología concebida para facilitar la interacción natural mediante el habla entre una persona y un ordenador (Llisterri & Machuca, 2006). Estos sistemas pueden ser considerados una interfaz o un intermediario entre un usuario y un sistema informático, que tiene la ventaja de no requerir el uso de una pantalla, un teclado o un ratón y de recurrir, en cambio, al medio de comunicación propio de los seres humanos. Por ello, suelen emplearse para servicios que se ofrecen a través del teléfono, aunque también se usan en todos aquellos casos en los que la expresión oral permite llevar a cabo una determinada tarea de forma más flexible y más eficaz que la escrita.

Los sistemas de diálogo que podemos utilizar hoy en día están sujetos a las limitaciones de la tecnología del reconocimiento automático del habla (RAH). En este sentido, no disponemos aún de reconocedores que puedan procesar con un cien por cien de fiabilidad la variación propia del habla completamente espontánea, la multiplicidad de realizaciones fonéticas de locutores de diversa procedencia geográfica y social, y las interferencias producidas por el entorno cuando el sistema tiene que utilizarse, por ejemplo, dentro de un vehículo en marcha o hablando por un teléfono móvil en plena calle.

Aun suponiendo que pudiéramos contar con los mejores reconocedores, un sistema de diálogo debería comprender cualquier enunciado, con independencia del tema o de su complejidad sintáctica, semántica y pragmática. La comprensión del lenguaje, es decir, la creación automática de una representación del contenido de un enunciado a partir de los resultados del reconocedor de habla, está limitada, por el momento, a dominios restringidos, ámbitos muy concretos a los que se aplica un determinado sistema de diálogo, como pueda ser la información de horarios de trenes, la reserva de billetes de avión o la información ciudadana.

En tercer lugar, debería disponerse de un sofisticado sistema de generación del lenguaje natural capaz de emular los mecanismos mediante los que las personas conseguimos producir enunciados que respondan acertadamente a cualquier situación, algo que todavía no está al alcance de la tecnología actual.

Finalmente, los métodos de conversión de texto en habla, pese a haber experimentado mejoras muy importantes en los últimos años, aún no permiten generar una voz sintetizada de igual calidad que la de una persona. Tampoco son capaces, por el momento, de ofrecer una realización prosódica de los enunciados idéntica a la del habla natural. Este es uno de los principales motivos por los que un usuario se percata inmediatamente de que, al llamar a un determinado servicio, quien le responde no es un interlocutor humano, sino un sistema de diálogo.

Para llevar a cabo las funciones descritas anteriormente, un sistema de diálogo debe constar de una serie de módulos, cada uno especializado en un aspecto concreto de la interacción entre la persona y el ordenador.

El sistema de diálogo tendrá como primera tarea analizar frases pronunciadas por el usuario para intentar extraer de ellas el contenido de la información solicitada. Este proceso inicial se realizará en el módulo de reconocimiento de habla, cuya tarea es convertir la onda sonora en una representación escrita que puedan utilizar los demás módulos del sistema. Para ello se necesita haber desarrollado previamente un conjunto de modelos fonéticos que recojan la variabilidad de cada segmento mínimo del habla –es decir, de cada alófono– y un modelo de lenguaje que establezca la probabilidad de combinación de una palabra con otra. Para obtener estos modelos se recurre al uso de corpus hablados, representativos tanto de los hablantes como del dominio de aplicación del sistema de diálogo. La tecnología utilizada en este módulo es, pues, el reconocimiento automático de habla (RAH o ASR, Automatic Speech Recognition), empleada también en otro tipo de aplicaciones como el dictado automático de textos, y considerada una parte esencial de las tecnologías del habla.

Sin embargo, una representación escrita de lo dicho por el usuario no es suficiente para responder a su pregunta, pues además del reconocimiento es necesario que el sistema de diálogo alcance un cierto grado de comprensión. Debe señalarse aquí que cuando en el contexto de las tecnologías lingüísticas se habla de comprensión, se hace referencia a la generación automática de una representación semántica abstracta de un enunciado. Esta tarea la realiza el módulo de comprensión o de interpretación semántica, recurriendo a las técnicas propias de la comprensión del lenguaje natural (NLU, Natural Language Understanding) o comprensión de habla (SLU, Spoken Language Understanding).

Una vez que el sistema de diálogo ha averiguado qué tipo de información desea recibir el usuario, entran en juego los mecanismos que le permiten acceder a bases de datos o bases de conocimiento que previamente se hayan creado. Con la información que de allí se extraiga, es necesario construir una respuesta, que deberá corresponder a una frase gramaticalmente bien formada. Esto supone una nueva tecnología: la generación del lenguaje (NLG, Natural Language Generation), que también forma parte del procesamiento del lenguaje natural y que constituye el núcleo de generación de respuestas del sistema de diálogo.

En una última etapa, la frase creada por el módulo de generación de respuesta tendrá que convertirse en su equivalente sonoro para que el usuario del sistema de diálogo pueda escucharla. De esta tarea se ocupa la conversión de texto en habla (TTS, Text-to-Speech Synthesis), una rama de las tecnologías del habla que tiene como objetivo transformar textos escritos en su realización oral. El módulo de conversión de texto en habla es, por tanto, el último paso de un sistema de diálogo y el que se encarga, finalmente, de hacer llegar la información a la persona que espera una respuesta.

Todas estas tareas están, en cierto modo, coordinadas o controladas por un módulo específico, denominado con gestor del diálogo, entre cuyas atribuciones se cuenta el seguimiento de la interacción entre el usuario y el de diálogo.

Un sistema de diálogo consiste, por lo tanto, en una estructura modular en la que cada módulo se ocupa de unas determinadas tareas, e interactúa con los demás módulos.

## 2.2 Evaluación de sistemas de diálogo

La evaluación de sistemas de diálogo es un campo de investigación muy importante actualmente. Por tanto, deber realizarse tanto la evaluación de componentes individuales como la evaluación de todo el sistema (López-Cózar & Araki, 2005; Möller, 2010).

El objetivo de la evaluación es obtener descripciones cuantitativas del rendimiento del sistema o de la calidad percibida por el usuario. En estos procesos de medida la herramienta de medición debe ser complementada por un elemento de medición, es decir, una persona que participa realizando pruebas del sistema. En este sentido, las mediciones son "subjetivas", en contraste con las mediciones instrumentales, que se basan sólo en un instrumento de medición física. La noción de "subjetividad", sin embargo, no implica que tales métodos sean menos fiables o válidos. Al contrario, las medidas de calidad fiables y válidas sólo pueden obtenerse con la ayuda de estas personas que participan en la evaluación.

Otro dato a tener en cuenta es que la evaluación, por lo general, requiere la compilación de un corpus importante de palabras y frases pronunciadas por los usuarios, relacionadas con el dominio de aplicación para el que se ha diseñado el sistema.

Otra técnica, que hace posible la reutilización de un corpus para la evaluación y para comprobar el rendimiento del sistema, es la llamada simulación de usuarios (López-Cózar et al., 2003). Ésta técnica se basa en la generación automática de conversaciones entre el sistema de diálogo y el simulador de usuarios el cual representa al usuario que está interactuando con el sistema de diálogo.

A continuación nos centramos en explicar una serie de medidas para evaluar algunos de los componentes del sistema.

### 2.2.1 Medidas de evaluación

#### *Evaluación del reconocimiento de habla*

La evaluación del reconocimiento de habla compara la hipótesis del reconocedor con una frase de referencia.

Con el fin de comparar las hipótesis con las frases de referencia, ambas tienen que estar alineadas. Para ello se puede usar un algoritmo de correspondencias de programación dinámica que asigna diferentes penalizaciones a palabras borradas, sustituidas o insertadas (Boros et al, 1996). Teniendo como base la alineación, el número de palabras correctas, sustituidas, borradas, e insertadas se cuenta y se determina la tasa de error de palabras (word error rate) o la exactitud de palabras (word accuracy).

Métricas similares también se pueden emplear a nivel de frase, dando como resultado tasas de error de frases (sentence error rate) o exactitud de frases (sentence accuracy). Métricas alternativas a nivel de frase incluyen el número promedio de errores por frase, y el error de palabras por frase. Por lo general, la exactitud de frases (sentence accuracy) es inferior a la exactitud de palabras, porque una sola palabra mal reconocida puede afectar a toda una frase. Sin embargo, algunos errores de reconocimiento no afectan al flujo del diálogo, pueden ser excluidos aplicando técnicas robustas de análisis parcial de frases o pueden ser completamente ignorados en caso de que no se vean afectadas "palabras clave" necesarias para la interpretación semántica de las frases.

#### *Evaluación del procesamiento del lenguaje natural*

La tarea del componente de “comprensión” es extraer el contenido semántico de la expresión de palabras del usuario. La evaluación por lo tanto consiste en determinar la exactitud de los contenidos semánticos extraídos.

Un enfoque es comparar directamente los conceptos semánticos, que pueden ser extraídos de la frase pronunciada por el usuario. A menudo, los conceptos semánticos se pueden formalizar en términos de pares atributo-valor.

Al igual que el enfoque adoptado para el reconocimiento de habla, se calcula la exactitud de conceptos (keyword accuracy) teniendo en cuenta los conceptos semánticos que se han sustituido, eliminado o insertado. Por otra parte, cada frase se puede etiquetar si ha sido correctamente analizada semánticamente y a continuación calcular la exactitud de comprensión o understanding accuracy en relación al número total de frases.

### 2.2.2 Simulador de usuarios

Con el fin de lograr un rendimiento aceptable de los sistemas de diálogo, es necesario desarrollar herramientas de evaluación de teniendo en cuenta varios aspectos. Por ejemplo, es necesaria la recopilación, transcripción y etiquetado de corpus de grabaciones de frases de usuarios de gran tamaño. No obstante, este material es con frecuencia difícil de obtener.

Un simulador de usuarios es una herramienta para facilitar el desarrollo y evaluación de sistemas de diálogo. Básicamente, es un sistema de diálogo adicional que interactúa automáticamente con el sistema de diálogo a evaluar, representando en cierta medida el comportamiento de un usuario real.

Mediante la simulación, el sistema de diálogo recibe como entrada una grabación de las frases pronunciadas por los usuarios, lo que presenta la ventaja de considerar los fenómenos reales existentes en el reconocimiento de habla.

El simulador proporciona a los desarrolladores de sistemas de diálogo un medio sencillo de comprobar las características del sistema y obtener una serie de medidas de evaluación, explicadas en la sección anterior.

## 2.3 Tecnologías del habla

Uno de aspectos que quizás dificultan el avance de las tecnologías del habla es la ausencia de una interfaz de programación (API) estándar, reconocida realmente por toda la industria.

El uso de estándares permite que los motores de RAH de distintos fabricantes puedan trabajar con aplicaciones de otras empresas, de modo que los desarrolladores pueden centrarse en lo que debe hacer su aplicación.

Los modelos de interfaz más extendidos para el desarrollo de sistemas de RAH son los siguientes:

- *JSAPI (Java Speech API),* de Sun Microsytems;
- *SRAPI (Speech Recognition API),* de Novell,
- *ViaVoice* de IBM y
- *SAPI (Speech API),* de la propia Microsoft.
- *Voice XML (VXML)*, estándar para aplicaciones de telefonía de la W3C.

Nos vamos a centrar en la tecnología usada para el desarrollo del sistema de diálogo *Mayordomo*: SAPI para Windows. Mientras que la API de reconocimiento de habla para programar aplicaciones en Windows se llama Microsoft SAPI (Speech Application Programming Interface), la aplicación para el reconocimiento de habla integrada para los usuarios se llama Windows Speech Recognition. A continuación, se explican cada una de ellas.

### 2.3.1 Windows Speech Recognition

#### *2.3.1.1 Visión General*

Desde hace unos años, Microsoft ha mostrado un gran interés en hacer accesibles las tecnologías del habla para un público mayoritario. Este interés ha dado lugar a productos como *Speech Server*, el cual se usa para implementar sistemas de telefonía con funciones de habla habilitadas, y *Voice Command*, el cual permite a los usuarios controlar dispositivos *Windows Mobile®* usando funciones del habla.

*Windows Speech Recognition* es una aplicación para el reconocimiento de habla, integrada en Windows Vista y Windows 7 y basada en SAPI 5.3. Este reconocedor de habla permite a los usuarios controlar el ordenador haciendo uso de comandos específicos de voz, por ejemplo, introducir texto y controlar todas las aplicaciones de Windows sin necesidad de usar el teclado o el ratón. Incluso las aplicaciones en las que estos "comandos" no vienen integrados, pueden ser controladas haciendo uso de una plantilla que se sitúa por encima de los elementos de la interfaz, incluyendo números que posteriormente se pueden hablar para activar esta función.

*Windows Speech Recognition* tiene una precisión de reconocimiento bastante elevada y proporciona un conjunto de comandos que ayuda en el dictado. Es necesario realizar un entrenamiento dirigido mediante un tutorial, para ayudar a familiarizar al usuario con los

comandos de reconocimiento de habla y al sistema con la voz del usuario, mejorando así la precisión del reconocimiento del habla (ver Apéndice B).

En la actualidad, la aplicación es compatible con varios idiomas, incluyendo inglés (EE.UU. y Reino Unido), español, alemán, francés, japonés y chino (tradicional y simplificado)

Estas son las principales características de *Windows Speech Recognition*:

- Una curva de aprendizaje reducida, centrándose en que el sistema "funcione".
- Interfaces de usuario rediseñadas sencillas pero eficaces para poder dictar y editar texto, además de corregir errores.
- Un tutorial interactivo que para enseñar a utilizar la aplicación al usuario mientras el ordenador se adapta a su voz.
- Es una tecnología del habla más precisa ya que aprende mientras el usuario la utiliza.
- Clarificación e interpretación mejoradas. Cuando se emite un comando que puede interpretarse de muchas formas, el sistema solicita al usuario que clarifique lo que quiso dar a entender.

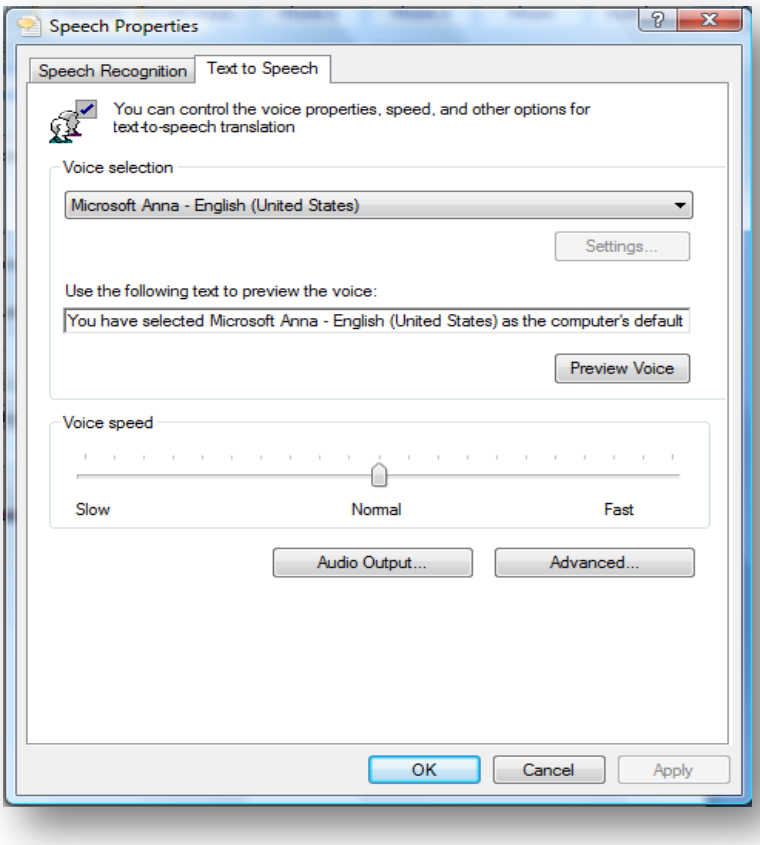



**Figura 1:** ***Propiedades del conversor de texto a habla: selección de la voz y velocidad.***

Es muy importante y totalmente necesario entrenar primero el reconocedor de habla (ver Apéndice B). Completar el tutorial interactivo de reconocimiento de voz permitirá al ordenador empezar a reconocer el habla del usuario con mayor precisión. Completar el tutorial también garantizará al usuario el aprendizaje de los comandos de voz necesarios para realizar tareas mediante la voz.

Se creará un perfil de reconocimiento de voz (ver Sección 5) propio a ese usuario. Un perfil de reconocimiento de voz es el conjunto de información que se crea cuando configura el reconocimiento de habla. El perfil puede incluir información específica sobre *cómo hablar* (incluidos los acentos y las pronunciaciones) y el sonido del *entorno que le rodea* (por ejemplo, el ruido de fondo de un ventilador, el ruido del aire acondicionado y otros sonidos que normalmente existen en las oficinas). En el sonido del entorno además influye el tipo de micrófono usado y la posición de éste con respecto los altavoces. Un micrófono de sobremesa normalmente funciona peor que otro tipo de micrófonos.

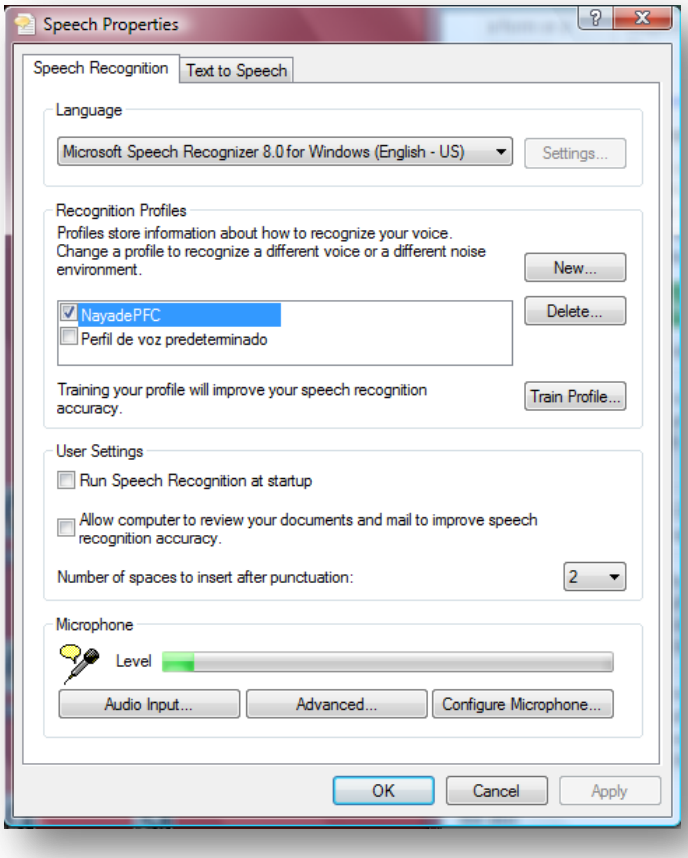



**Figura 2:** ***Propiedades del reconocedor del habla.***

Las condiciones cambian de un equipo a otro debido a todas estas características. Por estos motivos aunque fuera posible realizar una prueba de concepto que funcionase en un ordenador determinado, esta prueba seguramente no funcionaría en otro, por las diferentes condiciones acústicas. Y es que cuanto más ruido haya en el entorno, peor será el reconocimiento.

Además, Windows proporciona Windows Speech Recognition Profile Tool (WSR), que permite a un usuario de Windows hacer una copia de seguridad de los perfiles de usuario y poder restaurarlos posteriormente. Este procedimiento es útil cuando se transfiere un perfil de usuario entre un ordenador a otro o si se hace una instalación nueva del sistema operativo. Así evitamos tener que iniciar de nuevo el asistente del reconocimiento.

Es importante observar que toda la funcionalidad que Windows Speech Recognition aporta a los usuarios para manejar el sistema operativo, es la funcionalidad que directamente tienen las aplicaciones desarrolladas por los programadores, ya que ésta está integrada en el sistema operativo.

### 2.3.1.2 Otros aspectos técnicos

#### Síntesis de Voz

Para la síntesis de voz, Windows incluye la voz Microsoft Anna para el idioma inglés, la cual reemplaza la voz de Microsoft Sam usada en versiones anteriores. Anna está diseñada para sonar más natural y que se le entienda mejor.

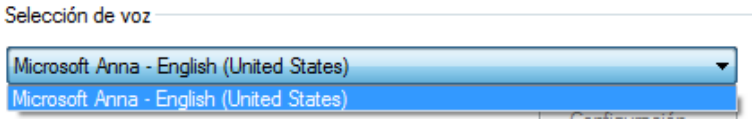



Figura 3: *Voces instaladas*

#### Reconocedor de habla

La aplicación utiliza Microsoft Speech Recognizer 8.0. Desde el *Panel de Control* se pueden consultar los reconocedores de voz instalados:

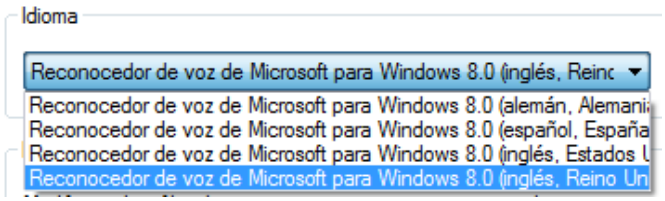



Figura 4: *Reconocedores de habla instalados*

#### Perfiles de reconocimiento

Como hemos comentado anteriormente, la primera vez que se ejecuta el software Windows Speech Recognition, se crea un perfil de reconocimiento del usuario (ver Apéndice B), en el cual se van almacenando una serie de datos. Estos perfiles se pueden crear, modificar o eliminar en el *Panel de Control*, dentro de las *Propiedades del Habla*.

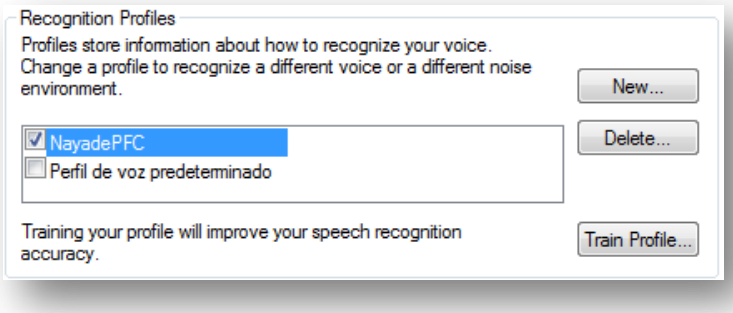



Figura 5: *Ejemplo de perfiles de reconocimiento*

El motor de reconocimiento de habla es el que gestiona estos perfiles. El motor implementa un componente de interfaz de usuario, User Training, al que se llama cuando se ejecuta la opción "*Train Profile*".

### 2.3.2 Herramienta para desarrolladores

Desde el punto de vista de los desarrolladores, se incluye un nuevo espacio de nombres para código manejado, *System.Speech* (Recognition y Synthesis), el cual permite habilitar fácilmente el reconocimiento de habla en aplicaciones basadas en *Windows Forms* y otras basadas en *Windows Presentation Foundation (WPF)*. Además, hay una nueva API de reconocimiento de habla actualizada y basada en COM para código nativo *Speech API (SAPI 5.3)* para dar acceso a las capacidades de habla integradas en la plataforma.

En este momento, la documentación es un poco deficiente. El SDK de Windows proporciona solamente ejemplos mínimos, y la mayoría de los que hay son para código nativo SAPI 5.3. Para encontrar información acerca de System.Speech, es recomendable visitar la página principal para desarrolladores (Microsoft Developer Network) en http://msdn.microsoft.com/en-us/library/gg145021.aspx (.NET Framework 4).

#### *2.3.2.1 Speech API*

##### *API para TTS*

La **síntesis de habla**, que en el contexto de los sistemas de diálogo se suele realizar mediante la técnica de *text-to-speech* (*TTS*), se usa para traducir a voz texto plano o en formato XML (la voz es la seleccionada por defecto en las opciones del habla en el Panel de Control).

El método para sintetizar el habla, puede funcionar tanto síncronamente (devuelve el flujo de ejecución sólo cuando el sistema ha terminado completamente de hablar) como asíncronamente.

##### *API para ASR*

El **reconocimiento de habla** (*ASR*) es más complicado que la síntesis, y la API refleja esa complejidad. Mientras que la síntesis tiene principalmente sólo una interfaz, el reconocimiento tiene al menos nueve. En cambio, las interfaces más importantes representan el motor de reconocimiento, el contexto en el que es usado, gramáticas y el resultado de la interpretación. El reconocimiento de habla tiene dos modos:

- Modo **dictado**: un modo de interpretación del habla de manera libre, que usa una gramática proporcionada por el reconocedor para un idioma concreto. Este es el reconocedor por defecto.
- Modo **gramática (command and control)**: busca la correspondencia entre las palabras pronunciadas por el usuario y una o más gramáticas libres de contexto (CFGs). Las gramáticas deben ser suministradas a la aplicación en forma de ficheros de *gramáticas precompiladas,* o bien, suministradas en tiempo de ejecución con el formato especificado en *W3C Speech Recognition Grammar Specification (SRGS)* o con la

especificación antigua *XML CFG*. Windows SDK incluye un compilador de gramáticas (*gc.exe*).

### *Reconocedor de habla*

Los motores de RAH generalmente son clasificados mediante dos características principales. La primera es el idioma del motor. El idioma lo provee el fabricante y no debe ser cambiado. Un ordenador puede tener varios tipos de motores instalados a la vez. En cambio, el uso del motor está limitado y depende del tipo de reconocimiento. La segunda característica es el tipo de reconocimiento, es decir, si la instancia del motor es creada como in-process (también conocida como InProc) o shared (compartido).

Por defecto, el motor de reconocimiento es seleccionado en las *Propiedades del Habla* del *Panel de Control*. Se usará el motor por defecto mientras no se especifique explícitamente otro motor. En muchas ocasiones, los requisitos de los usuarios se pueden solucionar con un único motor. En cambio, las aplicaciones no se ven restringidas al motor por defecto y pueden usar otros motores si es necesario. Por otro lado, un motor activo es el que es instanciado y está siendo usado como mínimo en un contexto de reconocimiento.

Más de una instancia de un motor puede ser usada. Cada aplicación puede tener su propia instancia de un reconocedor y en algunas situaciones, cada contexto de reconocimiento puede tener su propia instancia de motor. Las aplicaciones que usan reconocedores InProc deben tener su propia instancia. Si se necesita mayor granularidad para el reconocimiento, se recomienda usar diferentes contextos de reconocimiento mejor que usar múltiples reconocedores.

#### Tipos de reconocimiento

- Un motor de reconocimiento InProc está incluido dentro del mismo proceso que la aplicación. Un reconocedor InProc restringe el acceso a esa única aplicación. Por ejemplo, este reconocedor prohibiría que otras aplicaciones usaran el micrófono del sistema. Mientras otras aplicaciones pueden ejecutar su propia instancia de un reconocedor InProc, ningún recurso puede ser común a ellos. Un motor InProc puede ser usado, por ejemplo cuando se reconoce desde un fichero *wav*. De hecho, un motor Shared no puede usar como entrada ficheros *wav*.
- El segundo tipo de reconocedor en un reconocedor Shared (compartido). Se ejecuta como un proceso separado de la aplicación (*sapisvr.exe*). Como resultado, otras aplicaciones pueden usar los recursos del motor a la vez. Por ejemplo, el mismo micrófono del sistema será usado por todas las aplicaciones abiertas. A su vez, el reconocimiento resultante del motor podrá ser usado por cualquiera de las aplicaciones. De hecho, los motores compartidos llegan a informar a las aplicaciones cuando un resultado no es aplicable a ellas.

Para motores compartidos, una instancia del reconocedor se crea automáticamente cuando un contexto de reconocimiento es creado. En este caso, el tipo de reconocedor será el mismo que el tipo de contexto de reconocimiento. Esto es, un contexto de reconocimiento compartido creará un reconocedor compartido del mismo tipo que el motor activo. En cambio, todas las aplicaciones que usan un motor compartido deben crear instancias de este tipo de

motor. Una aplicación puede que no use un motor compartido y una segunda aplicación usar otro tipo de motor.

Los motores InProc se crean de manera semejante en que una instancia del motor InProc se crea cuando un contexto de reconocimiento es creado. Las instancias del motor InProc pueden ser creadas separadas. Cada aplicación puede tener sólo una instancia activa a la vez. En cambio, mientras que una aplicación puede tener sólo una instancia a la vez, diferentes aplicaciones, incluso estando abiertas a la vez, pueden cada una de ellas tener un motor activo diferente. Por ejemplo, una aplicación puede estar usando un motor InProc Inglés y otra aplicación puede estar reconociendo desde un motor Chino.

#### 2.3.2.2 System.Speech

System.Speech está orientado principalmente al uso de lenguajes de programación relacionados con *Microsoft .NET*, incluyendo C#, J#, Microsoft Visual Basic .NET, Microsoft JScript .NET, y C++.

La librería *System.Speech* viene incluida dentro de *.NET Framework 4.0* (y posteriores versiones). Esta librería es una colección de clases que permite el manejo del reconocimiento de habla (ASR) y la transformación de texto a habla (TTS). Está construida en su mayor parte sobre SAPI, siguiendo los mismos enfoques de programación de *SAPI 5.3*. Es por ello que los autores de este tipo de aplicaciones pueden usar *System.Speech* además de usar *SAPI*, o usar este espacio de nombres como alternativa. Aunque es opaco a los desarrolladores, *System.Speech.** se comunica con los motores de manera directa o indirecta haciendo llamadas a través de SAPI (*Sapi.dll*).

Este espacio de nombres puede ser usado para habilitar el habla en consola, *Windows Forms*, y aplicaciones de *Windows Presentation Foundation* (WPF). Para usar esta librería "manejada", una referencia a System.Speech.dll ha de ser añadida al proyecto. La especificación de ésta se puede consultar en la web del grupo MSDN (Microsoft Developers Network).

Aunque las clases de .NET Framework 4.0 están disponibles a bajo nivel para *Windows XP* y *Windows Server 2003*, la plataforma de soporte para las aplicaciones del habla se hace complicada de usar ya que estas tecnologías sólo están completamente integradas dentro de Windows. Por lo tanto, las siguientes limitaciones y advertencias se aplican:

- Aunque todas las versiones de Windows compatibles con .NET Framework incluyen un motor de síntesis del habla, sólo *Windows Vista* y *Windows XP Tablet PC Edition* integran un motor de reconocimiento del habla. Para otras versiones de Windows, será necesario incluir los ficheros binarios redistribuibles de Speech SDK.
- Cierta funcionalidad dentro del espacio de nombres de System.Speech.* depende de SAPI 5.3, pero Microsoft no tiene planes de redistribuir los binarios de SAPI 5.3 al bajo nivel de *Windows XP* o *Windows Server 2003*. Si una aplicación "manejada" que depende de esta funcionalidad avanzada se ejecuta en uno de estos antiguos sistemas operativos, saltará una excepción de PlatformNotSupportedException.

El espacio de nombres incluye algunas características no encontradas en SAPI 5.3, incluyendo un constructor de gramáticas, un constructor de *prompt* y una clase para documentos SRGS y gramáticas.

Los siguientes espacios de nombres componen la porción de habla para código "manejado" de la librería .NET Framework 4.0:

| Espacio de nombres | Descripción |
|---|---|
| **System.Speech.AudioFormat** | Contiene tipos que describen el flujo de audio usado para la entrada del reconocimiento de habla y la salida de la síntesis del habla. |
| **System.Speech.Recognition** | Contiene tipos para la implementación del reconocimiento de habla, incluyendo el acceso al servicio compartido por defecto de reconocimiento de habla, control de motor de reconocimiento, acceso a las gramáticas integradas y la capacidad de crear gramáticas. Y de recibir eventos del reconocedor. |
| **System.Speech.Recognition.SrgsGrammar** | Contiene tipos para crear y manipular gramáticas SRGS. |
| **System.Speech.Synthesis** | Contiene tipos para implementar la síntesis de habla, incluyendo acceso al motor sintetizador, características de la voz usada, soporte para documentos SSML, y otras cuestiones. |
| **System.Speech.Synthesis.TtsEngine** | Apoya la creación de motores basados en SSML para la síntesis de habla. |

**Tabla 1:** ***Espacios de nombres incluidos en System.Speech***

### *System.Speech.Recognition*

El espacio de nombres **Recognition** contiene tipos de *Windows Desktop Speech Technology* para la implementación del reconocimiento de habla. El software de *Windows Desktop Speech Technology* ofrece una infraestructura básica de reconocimiento de habla que digitaliza señales acústicas, y recupera palabras y elementos del habla desde la entrada de audio.

Las aplicaciones usan el espacio de nombres *System.Speech.Recognition* para acceder y extender esta tecnología básica de reconocimiento de habla, mediante la definición de algoritmos para identificar y actuar sobre determinadas frases o patrones de palabras, y mediante la gestión de comportamiento en ejecución de esta infraestructura del habla.

Las aplicaciones gestionan y usan gramáticas (conjunto de reglas que definen cómo un conjunto de palabras específico y frases han de ser entendidas) a través de la clase, de propósito general, **Grammar**, y a través de instancias creadas de **SrgsDocument**, que contienen documentos de gramáticas compatibles con la especificación de W3C Speech Recognition Grammar Specification (SRGS).

La API permite tres maneras de construir gramáticas:

- Usando las clases **GrammarBuilder** y **Choices**, una manera fácil de especificar documentos de gramáticas.
- .NET 3.0 proporciona un completo soporte con el espacio de nombres **System.Speech.Recognition.SrgsGrammar** para crear gramáticas compatibles SRGS.
- Además, para crear un modelo de gramática de dictado convencional, .NET 3.0 proporciona la clase **DictationGrammar**.

Instancias de los objetos *SpeechRecognizer* y *SpeechRecognitionEngine*, junto con instancias de los objetos *Grammar*, proporcionan el principal acceso a los motores de reconocimiento de *Windows Desktop Speech Technology*.

La clase *SpeechRecognizer* se usa para crear aplicaciones cliente haciendo uso de la actual tecnología de reconocimiento del sistema (el cual se configura a través del miembro AudioInput del *Panel de Control*) y del mecanismo de entrada de audio por defecto de audio del ordenador.

La clase *SpeechRecognitionEngine* permite más control en la configuración y tipo del motor de reconocimiento, el cual se ejecuta dentro del proceso principal de la aplicación (InProc). La clase *SpeechRecognitionEngine* proporciona además la posibilidad de seleccionar dinámicamente la entrada de audio, desde ficheros wav o cualquier otro dispositivo.

A grandes rasgos, la principal diferencia de *SpeechRecognizer* y *SpeechRecognitionEngine* que el primero levanta la aplicación *Windows Speech Recognition*, por lo que todo lo que se dice es escuchado por esta aplicación también, provocando errores y problemas cuando cree que ha escuchado un comando propio de esta aplicación, como “Open file”; mientras que usando la clase *SpeechRecognitionEngine*, sólo el motor de reconocimiento es activado, quedando todo el control a manos del desarrollador.

### Clases

| Clase | Descripción |
|---|---|
| **AudioLevelUpdatedEventArgs** | Devuelve datos del evento **AudioLevelUpdated**. |
| **AudioSignalProblemOccurredEventArgs** | Devuelve datos del evento **AudioSignalProblemOccurred**. |
| **AudioStateChangedEventArgs** | Devuelve datos del evento **AudioStateChanged**. |
| **Choices** | Representa una lista de items alternativos para construir un elemento de una gramática. |
| **DictationGrammar** | Representa una gramática usada para el dictado de texto libre. |
| **EmulateRecognizeCompletedEventArgs** | Devuelve datos del evento **EmulateRecognizeCompleted**. |
| **Grammar** | Proporciona soporte en tiempo de ejecución para obtener y gestionar información de gramáticas. |

| | |
|---|---|
| **GrammarBuilder** | Proporciona un mecanismo de fácil uso para construir objetos complicados **Grammar** desde entradas sencillas. |
| **LoadGrammarCompletedEventArgs** | Devuelve datos del evento **LoadGrammarCompleted**. |
| **RecognitionEventArgs** | Clase básica para los objetos de los argumentos del evento pasados a los manejadores de los eventos de reconocimiento del habla. |
| **RecognitionResult** | Representa el resultado del motor de reconocimiento sobre una entrada de audio como la información detallada sobre la mejor frase con coincidencia exacta de entre las candidatas, y una lista de todas ellas. |
| **RecognizeCompletedEventArgs** | Devuelve datos del evento **RecognizeCompleted**. |
| **RecognizedAudio** | Representa la entrada de audio para un motor de reconocimiento usada para generar el resultado de frases candidatas. |
| **RecognizedPhrase** | Representa la información detallada sobre una frase candidata encontrada por un motor de reconocimiento. |
| **RecognizedWordUnit** | Proporciona la unidad atómica de habla reconocida. |
| **RecognizerInfo** | Representa información sobre **SpeechRecognizer** o **SpeechRecognizerEngine**. |
| **RecognizerUpdateReachedEventArgs** | Devuelve datos del evento **RecognizerUpdateReached**. |
| **ReplacementText** | Contiene texto reemplazado por un motor de reconocimiento usando normalización del habla. |
| **SemanticResultKey** | Adjunta una cadena clave a los valores **SemanticResultValue** para definir los objetos **SemanticValue** creados en **Grammar** usando las instancias **GrammarBuilder**. |
| **SemanticResultValue** | Modifica valores de los objetos **SemanticValue** |

| | |
|---|---|
| | creados en **Grammar** usando las instancias **GrammarBuilder**. |
| **SemanticValue** | Representa la organización semántica de una frase reconocida. |
| **SpeechDetectedEventArgs** | Devuelve datos del evento **SpeechDetected**. |
| **SpeechHypothesizedEventArgs** | Infraestructura. Devuelve notificaciones del evento **SpeechHypothesized**. |
| **SpeechRecognitionEngine** | Proporciona acceso a ejecutar cualquier propiedad instalada de los servicios de reconocimiento del habla encontrada en un sistema Windows Desktop. |
| **SpeechRecognitionRejectedEventArgs** | Devuelve notificaciones del evento **SpeechRecognitionRejected**. |
| **SpeechRecognizedEventArgs** | Devuelve notificaciones del evento **SpeechRecognized**. |
| **SpeechRecognizer** | Proporciona acceso al servicio compartido de reconocimiento del hablar por defecto disponible en el Windows Desktop. |
| **SpeechUI** | Proporciona texto e información de estado al sistema operativo en la interfaz de habla de usuario para mostrar al usuario. |
| **StateChangedEventArgs** | Devuelve datos del evento **StateChanged**. |

Tabla 2: *Clases de System.Speech.Recognition*

## Enumeraciones

| Enumeración | Descripción |
|---|---|
| **AudioSignalProblem** | Enumera los tipos de problemas de la señal de audio. |
| **AudioState** | Enumera los valores del estado del audio del reconocedor. |
| **DisplayAttributes** | Enumera los formatos para mostrar las palabreas en una frase reconocida. |
| **RecognizeMode** | Enumera los valores del modo de reconocimiento. |
| **RecognizerState** | Enumera los valores del estado del reconocedor. |
| **SubsetMatchingMode** | Enumera los valores del modo de coincidencia entre subconjuntos. |

Tabla 3: *Enumeraciones de System.Speech.Recognition*

### *System.Speech.Synthesis*

#### Clases

| Clase | Descripción |
|---|---|
| **BookmarkReachedEventArgs** | Devuelve datos del evento **[E:System.Speech.Synthesis.SpeechSynthesizer.BookmarkReached]**. |
| **FilePrompt** | Representa una *salida de voz (prompt)* desde un fichero. |
| **InstalledVoice** | Representa un objeto **Voice** instalado. |
| **PhonemeReachedEventArgs** | Devuelve datos del evento **SpeechSynthesizer.PhonemeReached**. |
| **Prompt** | Ejecuta un *prompt* desde un texto o desde un **PromptBuilder**. |
| **PromptBuilder** | Crea un objeto vacío **Prompt** y provee métodos para añadirle contenido. |
| **PromptEventArgs** | Representa la clase básica para las clases **SpeechSynthesizer EventArgs.** |
| **PromptStyle** | Define un estilo de salida de habla que consiste en opciones para énfasis, velocidad, y volumen. |
| **SpeakCompletedEventArgs** | Devuelve notificaciones del evento **SpeechSynthesizer.SpeakCompleted.** |
| **SpeakProgressEventArgs** | Devuelve datos del evento **SpeechSynthesizer.SpeakProgress.** |
| **SpeakStartedEventArgs** | Devuelve notificaciones del evento **SpeechSynthesizer.SpeakStarted**. |
| **SpeechSynthesizer** | Apoya la producción de habla y salida DTMF. |
| **StateChangedEventArgs** | Devuelve datos del evento **SpeechSynthesizer.StateChange**. |
| **VisemeReachedEventArgs** | Devuelve datos del evento **VisemeReached**. |
| **VoiceChangeEventArgs** | Devuelve datos del evento **VoiceChange**. |
| **VoiceInfo** | Representa una voz de síntesis de habla (TTS). |

Tabla 4: *Clases de System.Speech.Speech*

#### Enumeraciones

| Enumeración | Descripción |
|---|---|
| **PromptBreak** | Enumera los valores para niveles de pausas entre palabras |
| **PromptEmphasis** | Enumera los valores para niveles de énfasis del habla en *prompts*. |

| | |
|---|---|
| **PromptRate** | Enumera los valores para niveles de velocidad de habla en *prompts.* |
| **PromptVolume** | Enumera los valores para niveles de volumen en *prompts.* |
| **SayAs** | Enumera los formatos de datos para el habla como hora, fecha, etc. |
| **SynthesisMediaType** | Enumera los tipos de media para síntesis. |
| **SynthesisTextFormat** | Enumera los tipos de entrada de texto para sintetizadores. |
| **SynthesizerEmphasis** | Enumera los niveles de énfasis del sintetizador. |
| **SynthesizerState** | Enumera los valores para el **State** del sintetizador. |
| **VoiceAge** | Define los valores para la edad de las voces sintetizadas. |
| **VoiceGender** | Define los para el género de las voces sintetizadas. |

Tabla 5: *Enumeraciones de System.Speech.Recognition*

## *Ejemplo de aplicación con System.Speech*

```
// Incluimos los espacios de nombres necesarios:
using namespace System::Speech::Synthesis;
using namespace System::Speech::Recognition;
using namespace System::Speech::Recognition::SrgsGrammar;

// Declaramos las variables para los motores:
SpeechSynthesizer^ speaker;
SpeechRecognizer^ recog;

this->speaker = gcnew System::Speech::Synthesis::SpeechSynthesizer();
this->recog = gcnew System::Speech::Recognition::SpeechRecognizer();

// Definimos el evento que salta cuando se reconoce:
this->recog->SpeechRecognized += gcnew
System::EventHandler<System::Speech::Recognition::SpeechRecognizedEven
tArgs ^>(this, &Form1::recog_SpeechRecognized);

// Habla (sintesis de voz por parte del motor TTS):
speaker->SpeakAsync("Please, identify yourself");

…
// Carga las gramaticas para el reconocimiento:
array <Speech::Recognition::Grammar^>^ vectorGram;
array <SrgsDocument^>^ vectorDoc;

for(int i = 0 ; i < n_gram; i++){
      vectorDoc[i] = gcnew SrgsDocument(electrodomesticos[i]);
      vectorGram[i] = gcnew Grammar(vectorDoc[i]);
      // Asocia cada gram. al motor de reconocimiento (SR):
      recog->LoadGrammar(vectorGram[i]);
}
// Activa el reconocimiento
recog->Enabled = true;
…

// Definicion del evento de reconocimiento, con sus argumentos (la
frase reconocida):
System::Void recog_SpeechRecognized(System::Object^  sender,
System::Speech::Recognition::SpeechRecognizedEventArgs^  e) {
```

```
        // Repite la frase reconocida:
        speaker->SpeakAsync(e->Result->Text);

}
```

# 3. Sistema de diálogo *Mayordomo*

## 3.1 Introducción

*Mayordomo* (Ábalos et al. 2010a, 2010b; Espejo et al. 2010a, 2010b) es una implementación de un sistema de Inteligencia Ambiental (AmI) para controlar distintos electrodomésticos de un hogar. La interacción se realiza a través de un sistema de diálogo multimodal, en fase de desarrollo, el cual permite la interacción con el hogar mediante habla espontánea, o bien, mediante una interfaz gráfica tradicional (basada en teclado y ratón).

Además de la funcionalidad básica de control de los electrodomésticos, el sistema *Mayordomo* incluye una serie de funcionalidades adicionales, por ejemplo, administración de usuarios y control parental:

- El control parental de determinados electrodomésticos permite restringir la interacción con los mismos para determinados usuarios.
- El administrador del sistema tiene privilegios para realizar acciones especiales, por ejemplo, instalar y desinstalar electrodomésticos o gestionar el control parental.
- Además, el sistema lleva a cabo un registro de todas las acciones que se realizan dentro del entorno, así como del autor de las mismas.

*Mayordomo* puede funcionar en cualquier tipo de vivienda, es decir, con cualquier distribución de habitaciones. Además, puede interactuar con cualquier tipo de electrodomésticos, siempre y cuando el fabricante de éstos proporcione los ficheros de configuración necesarios. Un fichero de configuración contiene las características o atributos de un electrodoméstico, así como las acciones que pueden realizarse con él (ver sección 3.6).

La instalación y desinstalación de electrodomésticos se puede realizar de forma dinámica, es decir, es posible añadir o eliminar electrodomésticos sin necesidad de reiniciar el sistema. Una vez instalado el electrodoméstico, el sistema puede interactuar con él exactamente igual que si hubiera estado disponible desde el momento de inicio.

## 3.2 Interacción mediante habla

Para implementar la interacción mediante habla el sistema usa el software Windows Speech Recognition, que incluye tanto el motor de reconocimiento automático de habla (RAH) como el conversor texto-habla (TTS, Text-To-Speech).

Windows incluye dos herramientas para programadores: SAPI 5.3 (Speech API) y System.Speech (espacio de nombres de .NET Framework). En concreto, (como se ha mencionado en la sección 2.3) para implementar el sistema hemos usado la herramienta System.Speech, ya que está orientada principalmente al uso de lenguajes de programación relacionados con Microsoft .NET. Esta herramienta proporciona una colección de clases que

permite usar RAH (clases de System.Speech.Recognition) y TTS (clases de System.Speech.Synthesis).

Actualmente, la interacción con el sistema ha de hacerse en el idioma Inglés, debido a una serie de problemas (Espejo et al. 2010b). Entre las ideas de trabajo futuro, se encuentra la adaptación del sistema de diálogo al idioma castellano.

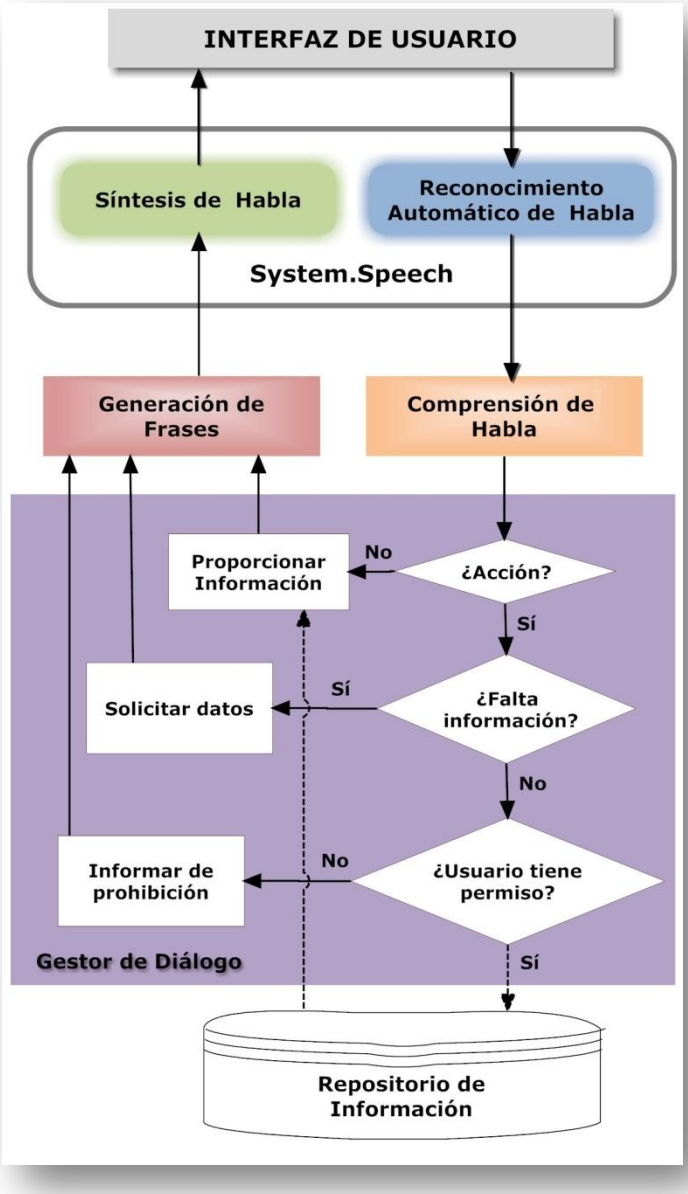



Figura 6: *Esquema de la interacción mediante el habla en* Mayordomo

### 3.2.1 Reconocimiento automático de habla (RAH)

El reconocimiento de habla es gestionado por el motor de reconocimiento de System.Speech.Recognition, creando una serie de eventos para cada parte del reconocimiento.

En este caso, SpeechRecognized es el evento que da como resultado la frase reconocida, que será la entrada al módulo de comprensión de habla.

Para realizar el reconocimiento, el motor necesita saber que frases ha de entender. Como se ha indicado anteriormente, cada electrodoméstico tiene asociado un fichero de configuración. Dicho fichero es el que permite que el usuario pueda "hablar" con el electrodoméstico, pues contiene la gramática específica que se usa durante la fase de RAH. Dicha gramática está escrita en formato SRGS (*Speech Recognition Grammar Specification),* que es una especificación más concreta que XML y ABNF.

A la hora de especificar las gramáticas para los diversos electrodomésticos de la vivienda, seguimos la siguiente estrategia (ver Figura 7). La regla inicial de las gramáticas contiene a su vez cuatro subreglas: *order*, *sentence*, *request* y *palabras clave*. La subregla *palabras clave* contiene una lista con todas las palabras clave relacionadas con el uso de un determinado electrodoméstico. Por ejemplo, para la TV son palabras clave aquéllas relacionadas con el lugar en que se encuentra el electrodoméstico (p. ej. living room, kitchen, etc.), la propiedad sobre la que se quiere actuar (p. ej. volume, channel, etc.), y la acción que se quiere realizar (p. e. switch on, turn off, etc.).

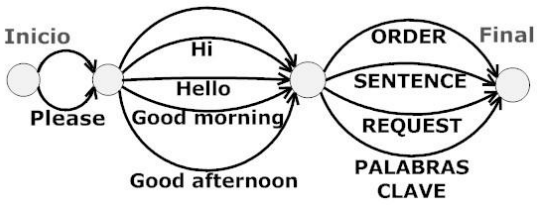



**Figura 7:** ***Regla inicial de las gramáticas***

Esta estrategia es adecuada para los usuarios que no proporcionan toda la información requerida para ejecutar una determinada *acción* sobre un electrodoméstico. Si faltan datos para poder realizar la acción, el gestor de diálogo del sistema solicita al usuario dichos datos. En ese momento el usuario puede proporcionar tan solo el dato que falta, es decir, no necesita pronunciar la orden completa. La ventaja es que la interacción resulta más cómoda para el usuario cuando las órdenes son complejas y largas. Por ejemplo, si a la hora de procesar la orden:

"*Set the temperature of the washing machine of the laundry room to thirty degrees*"

el sistema no entiende el dato correspondiente a la habitación en que se encuentra el electrodoméstico, entonces pregunta al usuario: "*Where?*", y éste puede responder: "*In the laundry room*".

### 3.2.2 Comprensión de habla

El proceso de comprensión de habla se basa en el concepto que denominamos *acción*. En nuestro dominio de aplicación, una acción está formada por cuatro tipos de datos: *habitación*, *electrodoméstico*, *atributo* y *valor* (ver Tabla 6). Usando estos cuatro elementos el

sistema puede ejecutar una determinada orden sobre un electrodoméstico, o bien, proporcionar la información solicitada por el usuario.

Para implementar el proceso de comprensión de habla se utiliza un método que busca en la frase reconocida los cuatro tipos de datos del concepto *acción***.** Para buscar el elemento *habitación*, el método trata de localizar en la frase cada uno de los nombres de las habitaciones existentes en la vivienda. El sistema puede detectar que en lugar de referirse a una habitación concreta, el usuario se está referido a todas las habitaciones de la vivienda.

Para buscar el elemento *electrodoméstico*, el sistema procede de forma análoga, es decir, trata de localizar en la frase reconocida cada uno de los nombres de los electrodomésticos existentes en la vivienda. Además, dado que el usuario puede referirse a los electrodomésticos de manera abreviada (por ejemplo, *"music"* en vez de *"piped music"*), el sistema trata de localizar en la frase reconocida fragmentos de los nombres de estos electrodomésticos.

| | |
|---|---|
| ***Habitación*** | *Habitación donde se encuentra el electrodoméstico, y por tanto, sobre la que se realiza la acción. Esta información es necesaria para distinguir, por ejemplo, qué luces han sido encendidas.* |
| ***Electrodoméstico*** | *Electrodoméstico concreto sobre el que se realiza la acción.* |
| ***Atributo*** | *Característica específica del electrodoméstico que se ve afectada por la acción.* |
| ***Valor*** | *Valor proporcionado por el electrodoméstico tras realizarse la acción sobre él.* |

**Tabla 6:** ***Descripción de los campos del concepto acción***

*Mayordomo* también procede de forma análoga para buscar el elemento *atributo*, es decir, trata de localizar en la frase reconocida cada uno de los nombres de los atributos correspondientes a los electrodomésticos existentes en la vivienda. Si encuentra un atributo, comprueba si ha encontrado previamente algún electrodoméstico. Si no lo encuentra, asume que el usuario ha omitido el nombre del electrodoméstico, en cuyo caso, busca a qué electrodoméstico está asociado. Si el atributo corresponde a un electrodoméstico único en la vivienda, la acción se asocia a dicho electrodoméstico. En cambio, si el atributo se corresponde con varios electrodomésticos, como en el caso del atributo *volumen*, que puede referirse tanto a la televisión como al hilo musical, el sistema pregunta al usuario a qué electrodoméstico se refiere.

Finalmente, el sistema usa el módulo *Reconocer Semántica* (ver Figura 8) para buscar el elemento *valor*, para lo cual trata de localizar en la frase reconocida cada uno de los nombres de los posibles valores de los atributos de los electrodomésticos existentes en la vivienda. Dado que algunos atributos son numéricos (están en el rango 0-10), el sistema trata de localizar números en ese rango dentro de la frase reconocida.

Por economía del lenguaje, hemos observado que los usuarios suelen pronunciar frases en las que se encuentran implícitos los atributos, y sólo se encuentra explicitado alguno de los valores de dichos atributos. Por ejemplo, en la frase: *"Switch on the lights"*, el atributo es *"State"*, pero éste no aparece en la frase. Para analizar estos casos, el sistema busca verbos para los cuales se sobreentienden *valores* y/o *atributos*.

Para determinar si el usuario está realizando una pregunta, o bien, comunicando el deseo de ejecutar una orden, se analiza la frase reconocida. Si en ésta se encuentra alguna palabra que comience con *"wh-"*, o alguna conjugación del verbo *"to be"* en presente, se asume que se trata de una pregunta. Más concretamente, si la palabra es *"what"* o *"which"*, se asume que el usuario está preguntando por el valor de algún atributo de un determinado electrodoméstico en una determinada habitación. Si la palabra es *"where"*, se asume que el usuario solicita información acerca del lugar donde algún electrodoméstico cumple determinadas condiciones.

| Información que falta | Pregunta realizada por el gestor de diálogo |
|---|---|
| **Habitación** | *Where?* |
| **Electrodoméstico** | *What appliance? / Which appliance…?* (si duda entre varios electrodomésticos, los nombra) |
| **Atributo** | *Which option?* |
| **Valor** | *Which value?* |

**Tabla 7:** ***Preguntas realizadas por el gestor de diálogo***

### 3.2.3 Gestión del diálogo

Una vez finalizado el análisis de la frase, el gestor del diálogo debe decidir la siguiente respuesta que debe generar el sistema. En concreto, debe determinar si ha de proporcionar información al usuario, o bien, realizar una acción concreta sobre un determinado electrodoméstico. Para ello, comprueba si en la frase falta algún dato de los cuatro tipos de datos constitutivos del concepto *acción* mencionados anteriormente. Si no falta ninguno, en caso de ser una consulta, el gestor del diálogo realiza una llamada al módulo *Proporcionar Información*. Siempre que se realiza una acción, ésta queda registrada en un fichero de traza, junto a la fecha y la hora. La Figura 8 presenta un esquema del proceso descrito.

En caso contrario, si se trata de una orden, se invoca al módulo *Realizar Acción*. Si falta algún dato, el gestor de diálogo decide que se genere la pregunta correspondiente para obtener del usuario dicho dato.

Como se ha mencionado anteriormente, el módulo *Realizar Acción* es invocado si no falta información ni se trata de una consulta. Este módulo hace efectivos los cambios respecto a los cuatro tipos de datos comentados anteriormente (habitación, electrodoméstico, atributo y valor) pudiéndose aplicar a una habitación concreta o incluso a todas las habitaciones de la vivienda a la vez. Por ejemplo, si un usuario solicita encender la luz de la cocina, el campo habitación habrá sido rellenado con "*kitchen*", el electrodoméstico con "*light*", el atributo con "*state*" y el valor con "*on*"; y en las estructuras de datos relativas a las luces de la cocina se modifica el atributo *"state"* a "*on*".

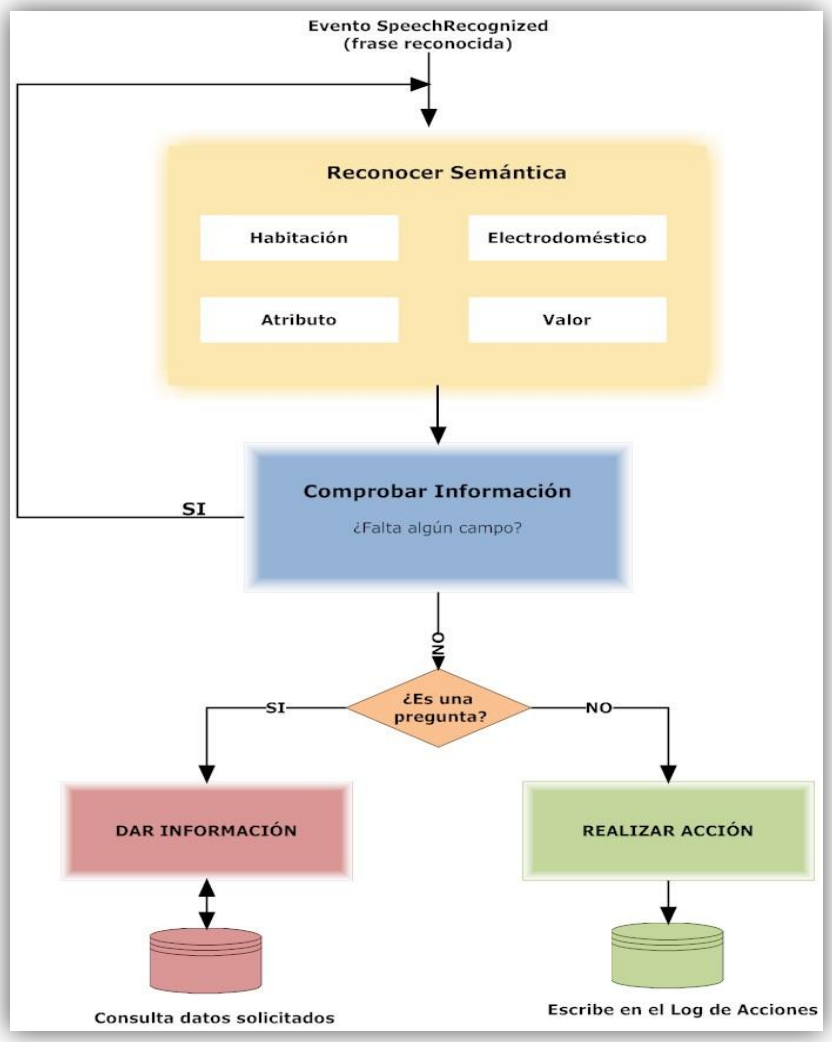



**Figura 8***: Esquema del proceso seguido por el gestor del diálogo*

Para que el gestor del diálogo conozca las características de la vivienda en la que el sistema *Mayordomo* ha de interactuar, se utiliza un archivo de configuración que contiene una descripción genérica de la vivienda, incluyendo el número de habitaciones de la misma, así como el nombre de cada habitación. Estos nombres podrán ser reconocidos y comprendidos por los módulos de RAH y PLN, respectivamente.

En este archivo de configuración, cada habitación tiene asociada a su vez una lista con los distintos electrodomésticos que se encuentran instalados en la misma. Para cada electrodoméstico existe información acerca del nombre, que como en el caso de las habitaciones, es el nombre por el que se reconoce en el sistema de diálogo.

El archivo de configuración específico de cada electrodoméstico contiene información acerca de la funcionalidad del electrodoméstico. Se detallan en él una serie de atributos o características, así como los valores posibles para cada uno de los atributos. Por ejemplo, en el caso de la televisión, el sistema usa un archivo que contiene todos los atributos relacionados con este electrodoméstico, como *volumen* y *canal*, así como los valores posibles que pueden tomar estos atributos.

Disponer de diferentes archivos de configuración para los distintos electrodomésticos facilita la implementación y escalabilidad del sistema. En particular, y pensando en una posible futura utilización comercial, nuestro diseño tiene en cuenta que cada fabricante de electrodomésticos puede desarrollar diferentes modelos de un mismo electrodoméstico. Así, por ejemplo, nuestro sistema podría interactuar con tipos diferentes de lavadoras en una misma habitación.

### 3.2.4 Generación de frases

Como se ha comentado en la sección anterior, el módulo de gestión del diálogo es el encargado de determinar la siguiente acción a realizar por el sistema. Existen dos posibilidades: respuesta oral para el usuario, o bien, acción sobre algún electrodoméstico. Por ejemplo, si el usuario ha formulado la pregunta: "*where are the lights on?*", el sistema puede responder: "*The lights are on in the kitchen and in the living room*". En cambio, si el usuario transmite una orden al sistema, el gestor del diálogo ejecuta la orden y seguidamente ordena la generación de un mensaje de confirmación. Por ejemplo, si la orden es: "*Turn up the volume of the T.V. in the living room*", el sistema puede responder: "*You have changed to five the volume of the T.V. in the living room*".

Para generar las respuestas, *Mayordomo* utiliza un conjunto de patrones que se instancian con unos valores u otros según el electrodoméstico, habitación, atributo y valor implicados. Por ejemplo, el patrón correspondiente al último ejemplo es el siguiente: "*You have changed to* (valor) *the* (atributo) *of the* (electrodoméstico) *in the* (habitación)".

### 3.2.5 Síntesis de habla

El sistema utiliza la síntesis de habla mediante conversión texto-habla (Text-To-Speech, TTS) para comunicarse de forma oral con el usuario, utilizando como entrada las frases en formato de texto creadas por la fase de generación de frases. Por ejemplo, esta técnica se usa al inicio de la interacción para proporcionar un mensaje de bienvenida al usuario y solicitar su autentificación.

La síntesis de habla también se usa para solicitar al usuario que complete la información que falta con objeto de poder ejecutar una determinada orden. Una vez realizada ésta, el sistema informa al usuario acerca del cambio de estado en el electrodoméstico en cuestión. Para implementar la síntesis de habla, usamos el software *System.Speech.Synthesis*.

## 3.3 Interacción gráfica

Con el propósito de facilitar la interacción con los electrodomésticos a un amplio rango de usuarios potenciales, *Mayordomo* proporciona una interfaz gráfica que incluye una serie de botones. Esta interfaz (ver Figura 9) permite visualizar las distintas habitaciones de la vivienda. La interfaz permite seleccionar cualquier habitación para mostrar los distintos electrodomésticos instalados en la misma. Para representar cada electrodoméstico se usa una imagen de éste que varía según su estado. Además, se muestra en la interfaz una lista con las

distintas características o atributos del electrodoméstico. Una vez seleccionado el atributo que se desea modificar o consultar, aparecen en otra lista los distintos valores posibles para este atributo o característica. Para cambiar el estado del electrodoméstico basta elegir un nuevo valor.

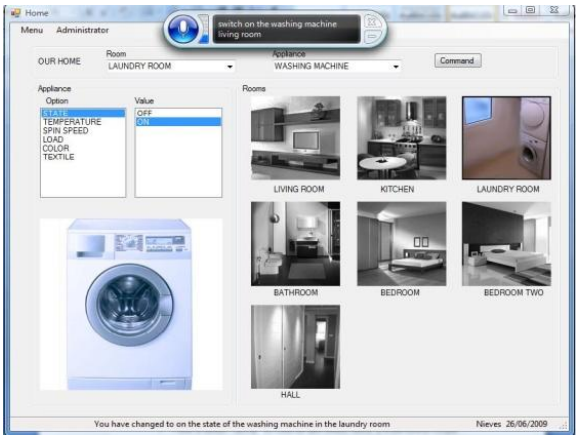



**Figura 9*: Interfaz gráfica de* Mayordomo**

La interfaz cuenta con una barra de estado y con un campo de texto en el que se muestra cada respuesta del sistema. Esta barra puede ser de gran utilidad en el caso de que se desee interactuar con el sistema en un lugar en que exista ruido, el cual puede impedir la correcta escucha de los mensajes orales generados mediante la técnica de conversión texto-habla (TTS).

La interfaz también proporciona un *prompt* de órdenes que permite dialogar con el sistema en formato de texto. Esta modalidad de interacción está pensada para posibles situaciones en las que el usuario interactúe en un entorno ruidoso que no aconseje usar la modalidad oral. El texto escrito en el *prompt* constituye la entrada del módulo de comprensión de habla descrito en la sección 2.1.2. En este caso, se consigue un mejor comportamiento del sistema dada la ausencia de errores de RAH.

## 3.4 Diseño actual

### 3.4.1 Electrodomésticos, atributos, valores y acciones.

Para cada electrodoméstico que forma parte del hogar con el que ha de interactuar el sistema, se definen las siguientes características (atributos y valores) y se definen las siguientes acciones, especificadas en los ficheros de configuración (.cfg) y gramáticas (.xml), respectivamente.

#### 3.4.1.1 Luz

| Nombre fichero | | *light.cfg* | |
|---|---|---|---|
| Nombre electrodoméstico | | *LIGHT* | |
| Atributos | | Valores | |
| ESTADO | *State* | ENCENDIDA<br>APAGADA | *On*<br>*Off* |
| INTENSIDAD | *Brightness* | [0-10] | *[0-10]* |
| Nombre de fichero de gramática | | *SRGSlight.xml* | |
| Acciones | | Atributo | |
| ENCENDER | *Switch on*<br>*Turn on* | *the light*<br>*the lights* | LUZ<br>LUCES |
| APAGAR | *Switch off*<br>*Turn off* | | |
| SUBIR | *Turn up* | *the brightness* | INTENSIDAD |
| BAJAR | *Turn down* | | |
| AJUSTAR A UN NÚMERO DETERMINADO | *Set to[0-10]* | | |

**Tabla 8:** ***Diseño para la luz.***

### 3.4.1.2 Lavadora

| Nombre de fichero de configuración | | *washing_machine.cfg* | |
|---|---|---|---|
| Nombre electrodoméstico | | *WASHING MACHINE* | |
| Atributos | | Valores | |
| ESTADO | *State* | ENCENDIDA<br>APAGADA | *on*<br>*off* |
| TEMPERATURA | *Temperature* | FRÍA<br>30º<br>40º<br>50º<br>60º<br>70º<br>95º | *cold*<br>*30*<br>*40*<br>*50*<br>*60*<br>*70*<br>*95* |
| VELOCIDAD DE CENTRIFUGADO | *spin speed* | RPM | *[1000-9000]* |
| CARGA | *Load* | MEDIA<br>COMPLETA | *half*<br>*full* |
| COLOR | *Color* | BLANCO<br>COLORES | *white*<br>*mixed* |
| TIPO TEJIDO | *textile* | ALGODÓN<br>LANA<br>DELICADO<br>SINTÉTICO<br>OTRO | *cotton*<br>*woolen*<br>*delicate*<br>*synthetic*<br>*other* |
| HORA DE INICIO DEL PROGRAMA | *time start* | HORA | *19:30 (p.ej.)* |

| Nombre de fichero de gramática | | *SRGSwashing.xml* | |
|---|---|---|---|
| Acciones | | Atributo | |
| ENCENDER | *Switch on*<br>*Start* | *the washing machine* | LAVADORA |
| | *Begin* | *the washing* | |
| APAGAR | *Switch off*<br>*Stop* | *the washing machine* | |
| | *End* | *the washing* | |
| CONFIGURAR OPCIONES DE LAVADO | *Set* | *temperature*<br>*spin speed*<br>*load*<br>*color*<br>*textile*<br>*time start* | TEMPERATURA<br>VEL.CENTRIFUGADO<br>CARGA<br>COLOR<br>TIPO TEJIDO<br>HORA INICIO |
| | | *options* | TODAS LAS OPCIONES DE LAVADO |

Tabla 9: *Diseño para la lavadora.*

### 3.4.1.3 Hilo musical

| Nombre fichero | | *piped_music.cfg* | |
|---|---|---|---|
| Nombre electrodoméstico | | *PIPED MUSIC* | |
| Atributos | | Valores | |
| ESTADO | *state* | ENCENDIDO<br>APAGADO | *on*<br>*off* |
| VOLUMEN | *volume* | [0-10] | *[0-10]* |
| ORIGEN | *source* | RADIO<br>DISPOSITIVO USB<br>DISCO DURO<br>REPRODUCTOR DE CD | *radio*<br>*U S B device*<br>*hard disk*<br>*C D player* |
| DISPOSITIVO | *device* | RADIO<br>DISPOSITIVO USB<br>DISCO DURO<br>REPRODUCTOR DE CD | *radio*<br>*U S B device*<br>*hard disk*<br>*C D player* |
| EMISORA DE RADIO | *radio channel* | NOMBRE DE LA EMISORA | *channel 1*<br>*channel 2* |
| Nombre de fichero de gramática | | *SRGSpiped_music.xml* | |
| Acciones | | Atributo | |
| ENCENDER | *Switch on*<br>*Turn on* | *the music*<br>*the piped music* | MÚSICA<br>HILO MUSICAL |
| APAGAR | *Switch off*<br>*Turn off* | | |
| SUBIR | *Turn up* | *the volume* | VOLUMEN |
| BAJAR | *Turn down* | | |
| SELECCIONAR | *Select* | *the device*<br>*the source*<br>*the hard disk*<br>*the radio*<br>*the U S B device*<br>*C D player* | DISPOSITIVO<br>ORIGEN<br>DISCO DURO<br>RADIO<br>DISPOSITIVO USB<br>REPRODUCTOR DE CD |
| AJUSTAR A UN NÚMERO DETERMINADO | *Set to[0-10]* | *the volume level* | NIVEL DE VOLUMEN |

Tabla 10: *Diseño para el hilo musical.*

#### 3.4.1.4 Televisión

| Nombre fichero | | *tv.cfg* | |
|---|---|---|---|
| Nombre electrodoméstico | | *TV* | |
| Atributos | | Valores | |
| ESTADO | *state* | ENCENDIDA<br>APAGADA | *On*<br>*Off* |
| VOLUMEN | *volume* | [0-10] | *[0-10]* |
| CANALES | *channel* | NOMBRE DE LOS CANALES | *B B C 1*<br>*B B C 2*<br>… |
| HORA | *time* | HORA O ADVERBIOS DE TIEMPO | *now*<br>*today*<br>*tomorrow*<br>*tonight* |
| Nombre de fichero de gramática | | *SRGStv.xml* | |
| Acciones | | Atributo | |
| ENCENDER | *Switch on*<br>*Turn on* | *the T V* | TV |
| APAGAR | *Switch off*<br>*Turn off* | | |
| SUBIR | *Turn up* | *the T V volume*<br>*the volume* | VOLUMEN |
| BAJAR | *Turn down* | | |
| AJUSTAR A UN NÚMERO DETERMINADO | *Set to[0-10]* | | |
| CAMBIAR DE CANAL | *Switch to [0-10]* | *the T V channel*<br>*the channel* | CANAL DE TV |

**Tabla 11:** ***Diseño para la televisión.***

## 3.5 Clases

### 3.5.1 Electrodomestico.h

En el archivo Electrodomestico.h se define la clase *Electrodomestico*, con sus métodos y sus atributos, y la estructura *Atributo*.

| Estructura | | Atributo |
|---|---|---|
| Atributos | | |
| String | nombre | Almacena el nombre del atributo. |
| Int | valor | Valor actual del atributo. |
| Int | tipo | Tipo de atributo: 0 = numérico; 1 = cadena de caracteres. |
| vector<string> | rango | Sólo si el tipo de atributo es 1 (cadena de caracteres), si no, estará vacío. Contiene en cada posición uno de los posibles valores del atributo. |

Tabla 12: *Estructura Atributo*

| Clase | | Electrodomestico |
|---|---|---|
| Atributos | | |
| String | nombreElectr | Almacena el nombre del electrodoméstico. Es el que se usa a la hora de reconocer a qué electrodoméstico nos referimos. |
| vector<Atributo> | listaAtributos | Contiene una lista de los atributos. |
| String | ficheroCFG | Contiene la ruta del fichero de configuración (.cfg). |
| String | ficheroXML | Contiene la ruta de la gramática del electrodoméstico (.xml). |
| Métodos | | |
| Electrodomestico() | | Constructor de la clase. |
| ~Electrodomestico() | | Destructor de la clase. |
| void leeFicheroConfig(const char *nombreFichero) | | Lee un fichero de configuración para un electrodoméstico. |
| Atributo getAtributo(int indice) | | Extrae del vector de atributos el atributo que está en esa posición |
| void setAtributo(Atributo unAtributo, int indice) | | Introduce en la posición indicada ese atributo. |
| int getNumDatos() | | Consulta el número total de atributos. |
| void setNumDatos(int num) | | Modifica el número total de atributos. |
| string getNombre() | | Consulta nombre del electrodoméstico. |
| void setNombre(string name) | | Modifica el nombre del electrodoméstico. |
| int buscarAtributo(string nombre) | | Busca un atributo, dentro del vector de |

| | |
|---|---|
| | atributos, con ese nombre. |
| string getFicheroCFG() | Consulta el nombre del fichero de configuración (.cfg) |
| string getFicheroXML() | Consulta el nombre del fichero de gramática (.xml) |
| void setFicheroCFG(string nombre) | Modifica el nombre del fichero de configuración (.cfg) |
| void setFicheroXML(string nombre) | Modifica el nombre del fichero de gramática (.xml) |

Tabla 13: *Clase Electrodoméstico*

### 3.5.2 Habitacion.h

En el archivo Habitacion.h se define la clase *Habitacion*, con sus métodos y sus atributos.

| Clase | | Habitacion |
|---|---|---|
| Atributos | | |
| String | nombreHab | Almacena el nombre la habitación. |
| vector<Electrodomestico> | listaElectrodomesticos | Contiene una lista de los electrodomésticos de dicha habitación. |
| Métodos | | |
| Habitacion() | | Constructor de la clase. |
| ~Habitacion() | | Destructor de la clase. |
| void setNombreElectrodomestico(int indice, string name) | | Modifica el nombre de un electrodoméstico del vector. |
| Electrodomestico getElectrodomestico(int indice) | | Extrae del vector de electrodomésticos el elemento que está en esa posición |
| void setElectrodomestico(Electrodomestico unElectr, int indice) | | Introduce en la posición indicada ese electrodoméstico. |
| int getNumDatos() | | Consulta el número total de electrodomésticos. |
| void setNumDatos(int num) | | Modifica el número total de electrodomésticos. |
| string getNombre() | | Consulta nombre de la habitación. |
| void setNombre(string name) | | Modifica el nombre de la habitación. |
| vector<int> buscarElectrodomestico(string nombre) | | Busca un electrodoméstico, dentro del vector, con ese nombre. |
| void borrarElectrodomestico(int indice) | | Elimina el electrodoméstico que se encuentra en la posición indicada del vector. |

Tabla 14: *Clase Habitación*

### 3.5.3 Hogar.h

En el archivo Hogar.h se define la clase *Hogar*, con sus métodos y sus atributos.

| Clase | | Hogar |
|---|---|---|
| Atributos | | |
| String | nombreHogar | Almacena el nombre del hogar. |
| vector<Habitacion> | listaHabitaciones | Contiene una lista de las habitaciones de la casa. |
| Métodos | | |
| Hogar() | | Constructor de la clase. |
| ~Hogar() | | Destructor de la clase. |
| void leeFicheroConfig(const char *nombreFichero) | | Lee un fichero de configuración para un hogar. |
| void escribeFicheroConfig(const char *nombreFichero) | | Escribe un fichero de configuración con los datos en memoria. |
| Habitacion getHabitacion(int indice) | | Extrae del vector de habitaciones la habitación que está en esa posición. |
| void setHabitacion(Habitacion, int indice) | | Introduce en la posición indicada esa habitación. |
| int getNumDatos() | | Consulta el número total de habitaciones. |
| void setNumDatos(int num) | | Modifica el número total de habitaciones. |
| string getNombre() | | Consulta nombre del hogar. |
| void setNombre(string name) | | Modifica el nombre del hogar. |
| vector<int> buscarHabitacion(string nombre) | | Busca una habitación dentro del vector de habitaciones, con ese nombre. |
| void setNombreHabitacion(int indice, string name) | | Modifica el nombre de una habitación del vector. |

**Tabla 15:** ***Clase Hogar***

## 3.6 Archivos de configuración

### 3.6.1 Archivo de configuración del hogar

**Formato genérico del fichero:** La extensión del fichero ha de ser *.cfg.* Las columnas están separadas por tabulaciones.

```
<nombre del hogar>
<número total de habitaciones>
<habitación 1>
<número de líneas>  <suma total de electr.>
<cantidad>  <nombre electr. 1>  <archivo de configuración 1>  <archivo de gramática 1>
<cantidad>  <nombre electr. 2>  <archivo de configuración 2>  <archivo de gramática 2>
…
<habitación 2>
<número de líneas>  <total de electr.>
…
```

Es muy importante tener en cuenta que el nombre del electrodoméstico se asigna aquí, a pesar de que en el archivo de configuración del propio electrodoméstico se permite asignar un nombre, que vendría dado por el fabricante. De esta manera se permite más flexibilidad para asignar nombres a electrodomésticos iguales.

**Ejemplo:**

Home.cfg

```
OUR HOME
8
LIVING ROOM
3	3
1	LIGHT	light.cfg	SRGSlight.xml
1	T V	tv.cfg	SRGStv.xml
1	PIPED MUSIC	piped_music.cfg	SRGSpiped_music.xml
KITCHEN
1	1
1	LIGHT	light.cfg	SRGSlight.xml
LAUNDRY ROOM
2	2
1	LIGHT	light.cfg	SRGSlight.xml
1	WASHING MACHINE	washing_machine.cfg	SRGSwashing.xml
BATHROOM
1	1
1	LIGHT	light.cfg	SRGSlight.xml
BEDROOM
3	3
1	LIGHT	light.cfg	SRGSlight.xml
1	PIPED MUSIC	piped_music.cfg	SRGSpiped_music.xml
```

```
1	T V	tv.cfg	SRGStv.xml
CORRIDOR
2	2
1	LIGHT	light.cfg	SRGSlight.xml
1	PIPED MUSIC	piped_music.cfg	SRGSpiped_music.xml
HALL
1	1
1	LIGHT	light.cfg	SRGSlight.xml
GARAGE
1	1
1	LIGHT	light.cfg	SRGSlight.xml
```

### 3.6.2 Archivos de configuración de los electrodomésticos

**Formato genérico del fichero:** La extensión del fichero ha de ser *.cfg.* Las columnas están separadas por tabulaciones.

```
<número total de atributos>
<nombre del electrodoméstico>
<atributo 1>	<tipo atributo>	<numero de valores>	<valor actual>	[valor[0]	valor[1] …]
<atributo 2>	<tipo atributo>	<numero de valores>	<valor actual>	[valor[0]	valor[1] …]
…
<atributo n>	<tipo atributo>	<numero de valores>	<valor actual>	[valor[0]	valor[1] …]
```

Aunque aquí permitimos asignar un nombre al electrodoméstico *(<nombre del electrodoméstico>*), no será el que se use, ya que consideramos que este nombre viene dado por el fabricante y es de control. El nombre real que se usará será el propio que se asigna en el archivo de configuración del hogar.

*<tipo atributo>* puede valer 0 o 1, dependiendo de si el atributo tiene valores numéricos o sus valores son cadenas de caracteres, respectivamente.

*<numero de valores>* indica el número total de valores que tiene ese atributo, para cadenas de caracteres, coincide con el número de elementos del vector que contiene los valores. Para valores numéricos enteros dentro del intervalo [0, 10], simplemente se indica el valor -1. Para otros valores numéricos, usar cadenas de caracteres.

*<valor actual>* indica, para valores numéricos, el valor entero actual que tiene ese atributo, mientras que para cadenas de caracteres, corresponde al índice del vector que contiene todos los valores.

**Ejemplos:**

***light.cfg***

```
2
LIGHT
STATE 1 2 0 OFF ON
BRIGHTNESS 0 -1 1
```

***tv.cfg***

```
4
T V
STATE 1 2 0 OFF ON
CHANNEL 1 3 0 B_B_C_1 B_B_C_2 FOX
VOLUME 0 -1 5
TIME 1 4 0 NOW TOMORROW TODAY TONIGHT
```

***piped_music.cfg***

```
5
PIPED MUSIC
STATE 1 2 0 OFF ON
VOLUME 0 -1 1
SOURCE 1 4 0 RADIO U_S_B_DEVICE HARD_DISK
C_D_PLAYER
DEVICE 1 4 0 RADIO U_S_B_DEVICE HARD_DISK C_D_PLAYER
RADIO_CHANNEL 1 2 0 CHANNEL_ONE CHANNEL_TWO
```

***washing_machine.cfg***

```
8
WASHING MACHINE
STATE 1 2 0 OFF ON
TEMPERATURE 1 7 1 COLD THIRTY FORTY FIFTY SIXTY
SEVENTY NINETY_FIVE
SPIN_SPEED 1 9 0 ONE_THOUSAND TWO_THOUSAND
THREE_THOUSAND FOUR_THOUSAND FIVE_THOUSAND
SIX_THOUSAND SEVEN_THOUSAND EIGHT_THOUSAND
NINE_THOUSAND
LOAD 1 2 0 HALF FULL
COLOR 1 2 0 WHITE MIXED
TEXTILE 1 5 0 COTTON WOOLEN DELICATE
SYNTHETIC OTHER
TIME 1 2 0 9_30 18_15
OPTIONS 1 1 0 NONE
```

# 4. Gestor del corpus

## 4.1 Introducción

Una vez presentado el sistema de diálogo *Mayordomo*, vamos a analizar el diseño (e implementación) realizado para desarrollar una serie de herramientas que permitan evaluar el sistema: un gestor de corpus de frases y un simulador de usuario.

Para poder realizar la evaluación, ya sea globalmente o de cualquiera de las partes o módulos en los que se divide el sistema, necesitamos un **corpus de frases** grabadas que serán la entrada de las herramientas desarrolladas.

El corpus es un conjunto de frases de audio, grabaciones de frases realizadas por usuarios, que nuestro sistema maneja para realizar las acciones en el dominio para el que está definido. Es decir, en nuestro corpus tenemos frases para encender electrodomésticos, apagarlos, etc. No sirven grabaciones de frases de otros dominios, por supuesto, ya que el sistema de diálogo no las entendería.

Hay una serie de pasos a seguir para conseguir un corpus de frases. El primero de todos es definir qué frases queremos. A continuación, buscar personas (locutores) que realicen estas grabaciones de frases y obtener así distintos ficheros de audio para la misma frase.

Estas frases grabadas son necesarias para que nuestro gestor del corpus obtenga una serie de medidas de evaluación para ver cómo funciona objetivamente el sistema. La evaluación es un requisito para saber cómo funciona realmente un sistema de diálogo (y cualquier tipo de software).

En las siguientes secciones se discute el diseño y desarrollo de cada uno de las partes del gestor, cuya finalidad es obtener las medidas de evaluación del sistema de diálogo.

### 4.1.1 Obtención del corpus

Para empezar, hay que diseñar el corpus de frases. En este caso, se han elegido una serie de frases con el fin de conectar cualquier electrodoméstico. Además, también necesitaremos frases que contengan únicamente los nombres de habitaciones, electrodomésticos, atributos, valores y acciones, como veremos más adelante. Cada uno de este conjunto de frases forma un escenario.

Nuestro diseño actual del corpus contiene dos escenarios de frases. Al escenario destinado a conectar electrodomésticos lo llamaremos a partir de ahora *escenario 1*, mientras que al escenario destinado a proporcionar nombres de habitaciones, electrodomésticos, atributos, valores y acciones lo llamaremos *escenario 2*. Cada uno de estos escenarios contiene 75 frases divididas en tres tandas de 25 frases.

Los escenarios son los siguientes:

| **ESCENARIO 1** |
| --- |
| Tanda 0:<br>Frase 1: Switch on the light<br>Frase 2: Please, switch on the light<br>Frase 3: Hi, switch on the light<br>Frase 4: Hello, switch on the light<br>Frase 5: Good morning, switch on the light<br>Frase 6: Good afternoon, switch on the light<br>Frase 7: Turn on the light<br>Frase 8: Please, turn on the light<br>Frase 9: Hi, turn on the light<br>Frase 10: Hello, turn on the light<br>Frase 11: Good morning, turn on the light<br>Frase 12: Good afternoon, turn on the light<br>Frase 13: Switch on the washing machine<br>Frase 14: Please, switch on the washing machine<br>Frase 15: Hi, switch on the washing machine<br>Frase 16: Hello, switch on the washing machine<br>Frase 17: Good morning, switch on the washing machine<br>Frase 18: Good afternoon, switch on the washing machine<br>Frase 19: Start the washing machine<br>Frase 20: Please, start the washing machine<br>Frase 21: Hi, start the washing machine<br>Frase 22: Hello, start the washing machine<br>Frase 23: Good morning, start the washing machine<br>Frase 24: Good afternoon, start the washing machine<br>Frase 25: Begin the washing machine<br><br>Tanda 1:<br>Frase 1: Please, begin the washing machine<br>Frase 2: Hi, begin the washing machine<br>Frase 3: Hello, begin the washing machine<br>Frase 4: Good morning, begin the washing machine<br>Frase 5: Good afternoon, begin the washing machine<br>Frase 6: Switch on the piped music<br>Frase 7: Please, switch on the piped music<br>Frase 8: Hi, switch on the piped music<br>Frase 9: Hello, switch on the piped music<br>Frase 10: Good morning, switch on the piped music<br>Frase 11: Good afternoon, switch on the piped music<br>Frase 12: Turn on the piped music<br>Frase 13: Please, turn on the piped music<br>Frase 14: Hi, turn on the piped music<br>Frase 15: Hello, turn on the piped music<br>Frase 16: Good morning, turn on the piped music<br>Frase 17: Good afternoon, turn on the piped music<br>Frase 18: Switch on the music<br>Frase 19: Please, switch on the music<br>Frase 20: Hi, switch on the music<br>Frase 21: Hello, switch on the music<br>Frase 22: Good morning, switch on the music |

Frase 23: Good afternoon, switch on the music
Frase 24: Turn on the music
Frase 25: Please, turn on the music

Tanda 2:
Frase 1: Hi, turn on the music
Frase 2: Hello, turn on the music
Frase 3: Good morning, turn on the music
Frase 4: Good afternoon, turn on the music
Frase 5: Switch on the T.V.
Frase 6: Please, switch on the T.V.
Frase 7: Hi, switch on the T.V.
Frase 8: Hello, switch on the T.V.
Frase 9: Good morning, switch on the T.V.
Frase 10: Good afternoon, switch on the T.V.
Frase 11: Turn on the T.V.
Frase 12: Please, turn on the T.V.
Frase 13: Hi, turn on the T.V.
Frase 14: Hello, turn on the T.V.
Frase 15: Good morning, turn on the T.V.
Frase 16: Good afternoon, turn on the T.V.
Frase 17: Switch the light on
Frase 18: Please, switch the light on
Frase 19: Hi, switch the light on
Frase 20: Hello, switch the light on
Frase 21: Good morning, switch the light on
Frase 22: Good afternoon, switch the light on
Frase 23: Turn the light on
Frase 24: Please, turn the light on
Frase 25: Hi, turn the light on

**ESCENARIO 2**

Tanda 0:
Frase 1: In the corridor
Frase 2: Of the laundry room
Frase 3: In the kitchen
Frase 4: In the Bedroom
Frase 5: Of the living room
Frase 6: In the bathroom
Frase 7: Of the hall
Frase 8: In the garage
Frase 9: In every room
Frase 10: In all rooms
Frase 11: Everywhere
Frase 12: Zero
Frase 13: One
Frase 14: Two
Frase 15: Three
Frase 16: Four

Frase 17: Five
Frase 18: Six
Frase 19: Seven
Frase 20: Eight
Frase 21: Nine
Frase 22: Ten
Frase 23: Brightness
Frase 24: Light
Frase 25: All the lights

Tanda 1:
Frase 1: Off
Frase 2: On
Frase 3: Down
Frase 4: Up
Frase 5: To
Frase 6: Music
Frase 7: The piped music
Frase 8: The volume
Frase 9: Volume level
Frase 10: Radio channel
Frase 11: Source
Frase 12: Device
Frase 13: Radio
Frase 14: C D player
Frase 15: Hard disk
Frase 16: U S B device
Frase 17: Switch on
Frase 18: Switch off
Frase 19: Turn up
Frase 20: Turn down
Frase 21: Turn on
Frase 22: Turn off
Frase 23: Select
Frase 24: Set to
Frase 25: Now

Tanda 2:
Frase 1: Today
Frase 2: Tomorrow
Frase 3: Tonight
Frase 4: T V
Frase 5: T V channel
Frase 6: T V volume
Frase 7: The washing
Frase 8: Washing machine
Frase 9: Temperature
Frase 10: Spin speed
Frase 11: Load
Frase 12: Color
Frase 13: Textile

Frase 14: Time start
Frase 15: Options
Frase 16: Cold
Frase 17: Thirty degrees
Frase 18: Forty degrees
Frase 19: Fifty degrees
Frase 20: Sixty degrees
Frase 21: Seventy degrees
Frase 22: Ninety five degrees
Frase 23: One thousand R P M
Frase 24: Two thousand R P M
Frase 25: Three thousand R P M

El procedimiento realizado para obtener las grabaciones de frases es el siguiente. Locutores voluntarios realizan llamadas telefónicas a un call-center. Éste les explica cómo grabar las 75 frases. En caso de que los locutores no estén contentos con las frases, el call-center permite la repetición de las grabaciones.

Como vemos, la obtención del corpus de frases depende de la existencia de locutores voluntarios y distintos que deseen realizar las grabaciones de frases.

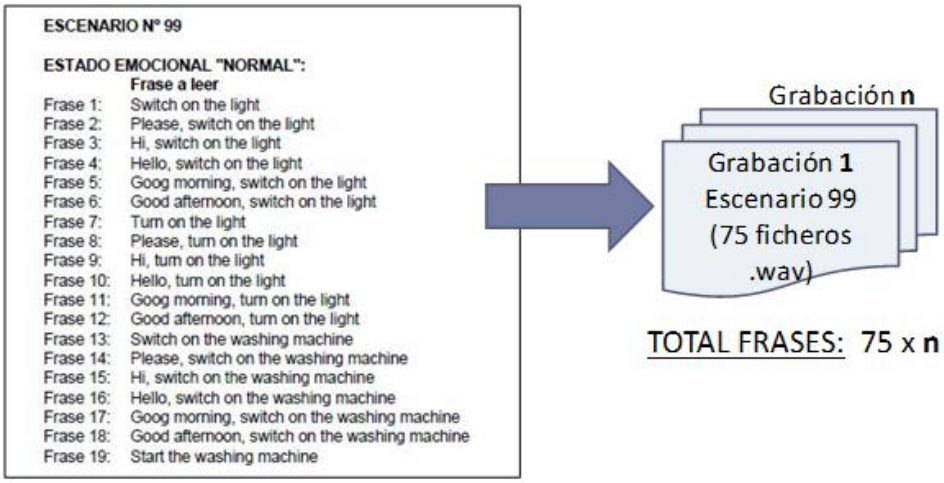



**Figura 10:** ***Obtención del corpus de grabaciones de frases***

Actualmente, nuestro corpus cuenta con 1500 grabaciones de frases, de las cuales 750 son del escenario 1 y las otras 750 corresponden al escenario 2, es decir, se han realizado 10 grabaciones completas de frases de cada uno de los escenarios. Para ello, se ha contado con la colaboración de quince usuarios voluntarios para la grabación y obtención de este corpus. Las frases han sido grabadas llamando por teléfono y se almacenan utilizando 16 bits por muestra a 8 kHz.

Cada frase pronunciada se ha almacenado en un fichero de audio con extensión .wav, cuyo nombre tiene un código identificativo de escenario, número de frase y tanda de frases. Esta información es útil para la etapa de la transcripción automática del corpus mediante la

herramienta diseñada para la evaluación del sistema de diálogo, como se explica en la siguiente sección.

### 4.1.2 Transcriptor Ortográfico Automático

El transcriptor ortográfico automático es un módulo, que dado un fichero de audio (.wav) del corpus, crea un fichero de texto (.txt) con el mismo nombre, cuyo contenido es la transcripción ortográfica de dicha frase de audio. El fichero de texto creado es de vital importancia para la evaluación del sistema de diálogo, ya que contiene la frase especificada en el escenario, sin posibles errores de RAH. El nombre de cada fichero .wav grabado contiene un código identificativo, como vemos en la figura siguiente:

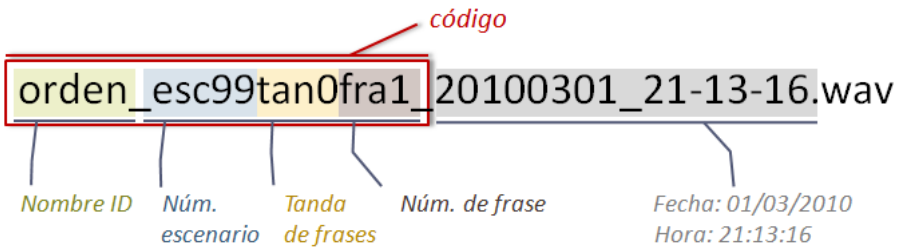



Figura 11: *Significado del nombre de un fichero del corpus de frases*

El transcriptor usa una tabla, almacenada en un fichero de texto llamado "Correspondencias.txt", que contiene las correspondencias entre código y frase especificada en el escenario:

| | | | | |
|---|---|---|---|---|
| orden_ | esc99 | tan0 | fra1 | Switch on the light |
| orden_ | esc99 | tan0 | fra2 | Please, switch on the light |
| ... | | | | ... |
| ... | | | | ... |

Tabla 16: *Tabla de correspondencias*

En el siguiente esquema se explica el funcionamiento del transcriptor automático, con un ejemplo de fichero de entrada. Como vemos, la salida del transcriptor es un fichero de texto cuyo contenido, en este caso, es la frase "Switch on the light".

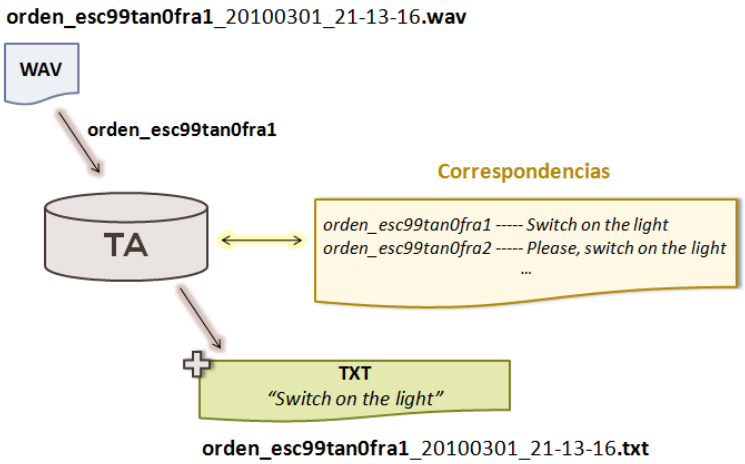



Figura 12: *Transcripción Automática Ortográfica de un fichero del corpus de frases*

Tras aplicar el transcriptor automático al corpus, obtenemos otros 1500 ficheros de texto con extensión ".txt" que también pasan a formar parte del corpus.

### 4.1.3 Transcriptor Semántico Automático

El transcriptor semántico automático es un módulo, que dado un fichero de texto correspondiente a la transcripción de una frase del corpus, crea un fichero de texto con extensión .RSCor, con el mismo nombre que el fichero de texto de entrada, y cuyo contenido es la representación semántica de referencia asociada a esa frase, y por lo tanto, a la frase de audio con el mismo nombre y extensión .wav.

La **representación semántica de referencia** creada es de gran importancia también para la evaluación del sistema de diálogo, ya que será usada por el simulador de usuario que será explicado en la siguiente sección.

En el siguiente esquema se explica el funcionamiento del transcriptor semántico, usando un ejemplo de fichero de entrada. Como vemos, la entrada del transcriptor es un fichero de texto cuyo contenido es la frase "Switch on the light". La salida del transcriptor es otro fichero con la representación semántica: "Appliance = Light Attribute = State Value = On".

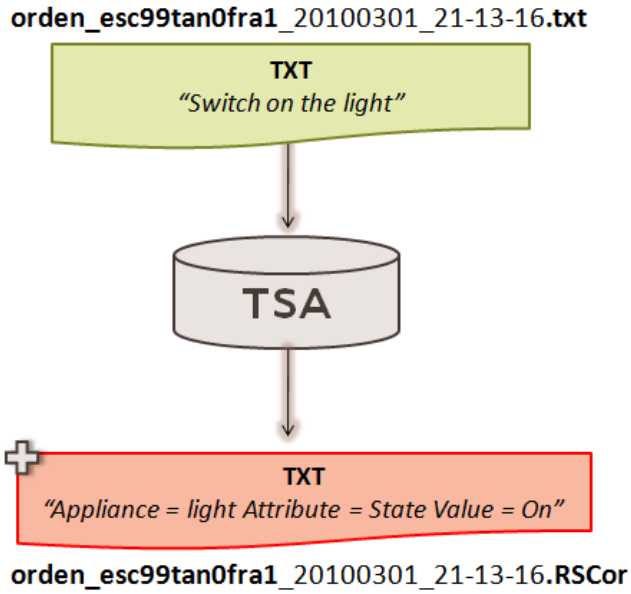



Figura 13: *Transcripción Automática Semántica de un fichero del corpus de frases*

Tras aplicar el transcriptor semántico automático del corpus, obtenemos otros 1500 ficheros de texto con extensión ".RSCor", que también pasan a formar parte del corpus.

Así, en total, el corpus contiene 1500 frases (grabaciones de pronunciaciones de frases), 1500 transcripciones ortográficas (archivos de texto) y otras 1500 representaciones semánticas.

### 4.1.4 Gestor del Corpus

El gestor del corpus ejecuta las tareas necesarias para obtener las medidas de evaluación del sistema de diálogo, tomando como entrada cada una de las frases que forman el corpus de grabaciones de frases.

Las medidas calculadas se centran en evaluar tanto la etapa de reconocimiento de habla (RAH) como la de procesamiento del lenguaje natural (PLN) y gestión del diálogo. Respecto el RAH, estas medidas son la exactitud de palabras (Word Accuracy, WA) y el porcentaje o tasa de frases reconocidas correctamente (Sentence Recognition, SR).

Respecto al PLN, se calcula el porcentaje o tasa de frases comprendidas correctamente (Sentence Understanding, SU), y la exactitud de conceptos (Keyword Accuracy, KWA). Respecto al gestor del diálogo, se calcula el número de frases que aunque no se han reconocido perfectamente han sido comprendidas correctamente (Implicit Recovery, IR).

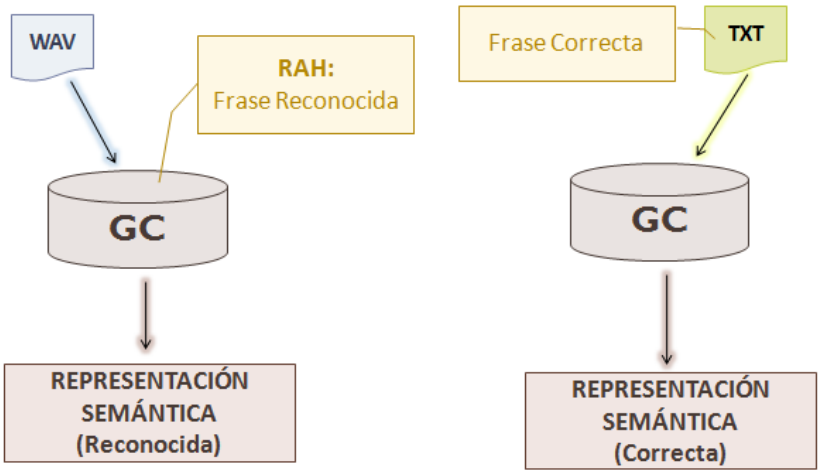



Figura 14: *Obtención de representaciones semánticas*

La primera tarea para calcular estas medidas es comparar cada frase reconocida con la frase de referencia, esto es, la frase pronunciada. Usando ambas frases, el gestor puede calcular la **exactitud de palabras** o **WA**:

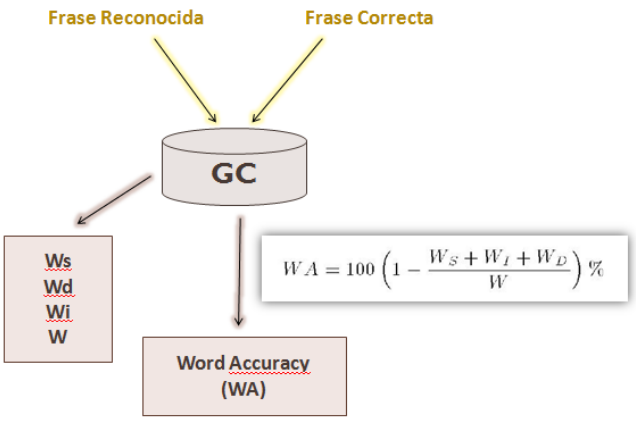



Figura 15: *Cálculo de la exactitud de palabras*

Si no se ha producido ningún error de reconocimiento, el gestor obtiene una representación que coincide con la de referencia (creada a partir de la transcripción ortográfica de la frase). El gestor utiliza un método que analiza ambas representaciones semánticas para extraer la información relevante. De esta manera, una vez obtenidas, calcula la **exactitud de conceptos**, **KWA**, de forma análoga a la usada por la exactitud de palabras:

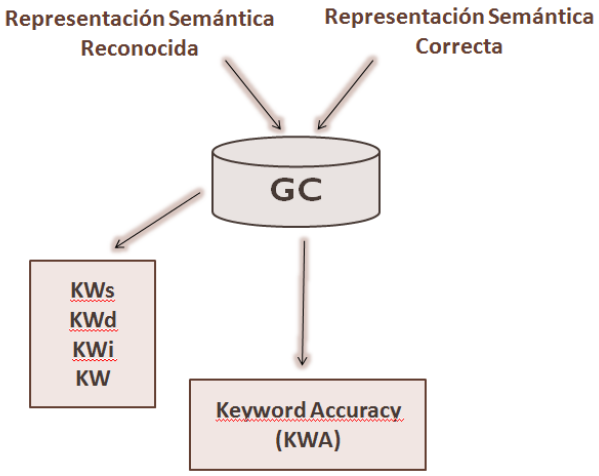



Figura 16: *Cálculo de la exactitud de conceptos*

El motor de reconocimiento, tras el reconocimiento de la frase, proporciona información útil para el gestor del corpus, por ejemplo, confianza de la frase reconocida e hipótesis de reconocimiento. Toda esta información la almacena el gestor en varios ficheros.

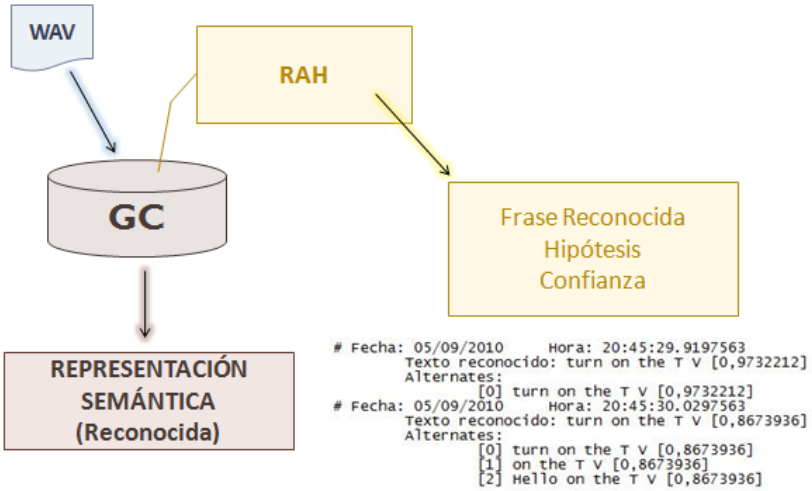



Figura 17: *Información obtenida en el reconocimiento*

Para calcular las demás medidas de evaluación, el gestor tiene en cuenta:

i) El número de frases reconocidas correctamente (WA = 100%) → Sentence Recognition (SR).

ii) El número de frases entendidas correctamente (KWA = 100%) → Sentence Understanding (SU).

iii) El número de frases que a pesar de no haberse reconocido correctamente (WA < 100 %), se han comprendido correctamente (KWA = 100%) → Implicit Recovery (IR).

iv) El número de frases que no se han reconocido correctamente.

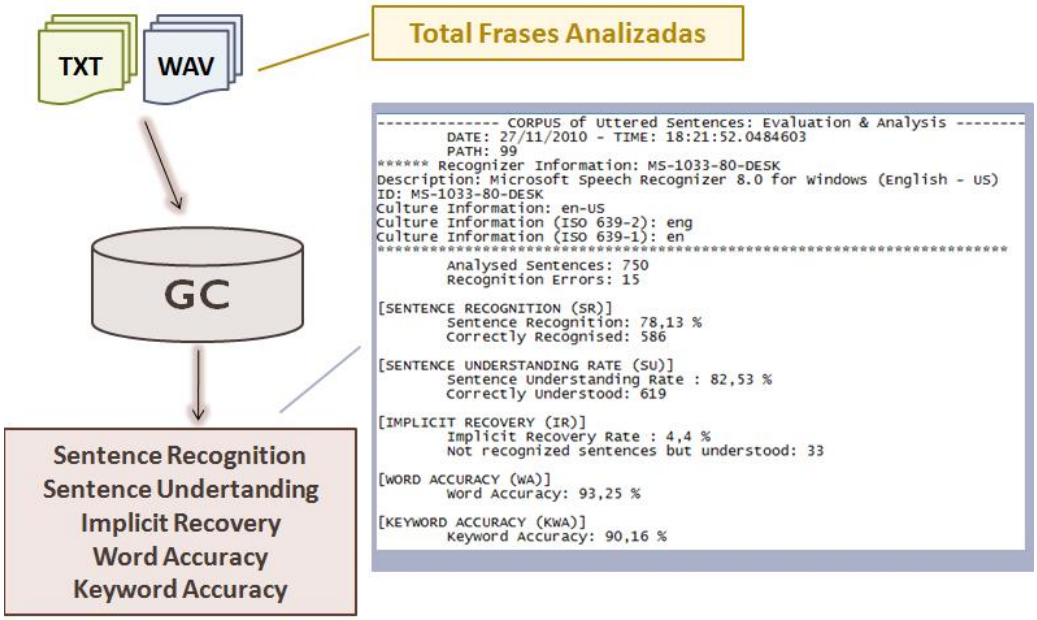



Figura 18: *Medidas estadísticas obtenidas por el gestor del corpus*

### 4.1.5 Simulación de usuarios

Otra herramienta para evaluar sistemas de diálogo es un simulador de usuarios. El simulador hace uso de las frases del corpus para crear diálogos, interactuando con el sistema de diálogo como si de un usuario se tratara, con unos objetivos marcados y respondiendo a las preguntas que realice el sistema de diálogo.

De esta manera, además de las medidas de evaluación anteriormente mencionadas, y que también tienen aplicación en esta herramienta, se puede considerar la medida **Task Completion (TC)** o completitud de tarea, teniendo en cuenta el número de diálogos terminados satisfactoriamente.

Los pasos a seguir para implementar la interacción entre el simulador y el sistema de diálogo son los siguientes:

- **Determinar el objetivo de la interacción con el sistema de diálogo.**

El usuario tendrá como objetivo realizar una **acción** (ver sección 3) sobre algún electrodoméstico del hogar, por ejemplo, encender una luz del pasillo. Por consiguiente, el simulador elegirá aleatoriamente uno de los objetivos previamente definidos e incluidos en el fichero “ListaObjetivos.txt”.

- **Elegir el fichero que será usado a modo de turno del usuario en el diálogo.**

En el caso del sistema *Mayordomo*, el usuario es el que tiene la iniciativa a la hora de empezar una conversación. Con lo cual, el simulador empezará el diálogo eligiendo una frase del corpus que se ajuste a sus objetivos. Para ello, busca entre todas las representaciones semánticas almacenadas en el gestor (ficheros .RSCor) y elige una de ellas aleatoriamente que cumpla *la mayor cantidad de objetivos posibles*. Éste será el fichero (con extensión .wav) que el simulador de usuario utilizará para interactuar con el sistema de diálogo en su primer turno.

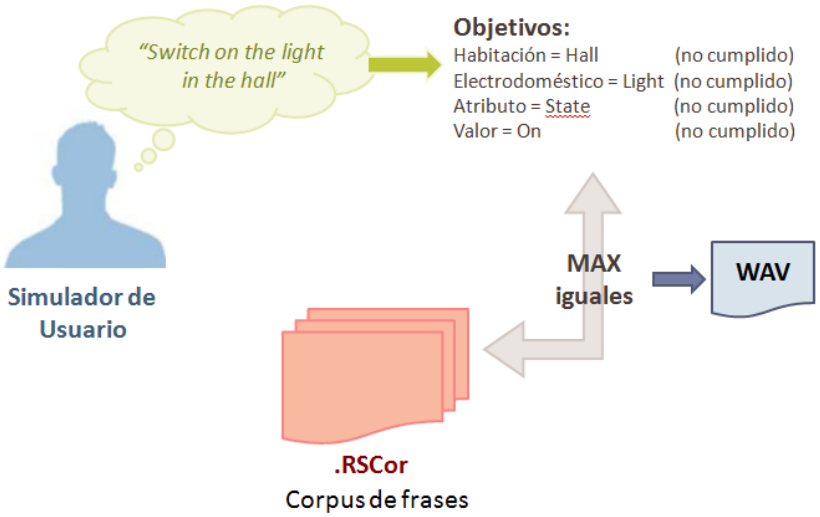



Figura 19: *Estrategia de selección de frases en la simulación de usuarios*

- **Reconocimiento de habla.**

Una vez que el fichero ha sido elegido, el motor realiza el reconocimiento de la frase grabada. De esta manera, se puede usar la frase reconocida, la frase correcta, y ambas representaciones semánticas (la obtenida del RAH y la de referencia).

- **Gestión del diálogo y generación de frases.**

Si el sistema de diálogo ha entendido correctamente la frase proporcionada por el simulador, esto es, no ha habido ningún error de reconocimiento, comprueba si falta alguna información (habitación, electrodoméstico, atributo o valor). Si falta alguno de estos campos, realiza las preguntas necesarias para obtenerla.

Además, en caso de duda, comprueba que la información obtenida coincide con el valor de los objetivos. Por ejemplo, si la habitación comprendida no está en los objetivos del usuario, pregunta si realmente el usuario quiso decir eso.

En caso de que la información haya sido entendida correctamente por el sistema de diálogo, el simulador marca los objetivos correspondientes como cumplidos.

Además, se calculan las mismas medidas de evaluación que se explicaron en la sección anterior para el gestor del corpus.

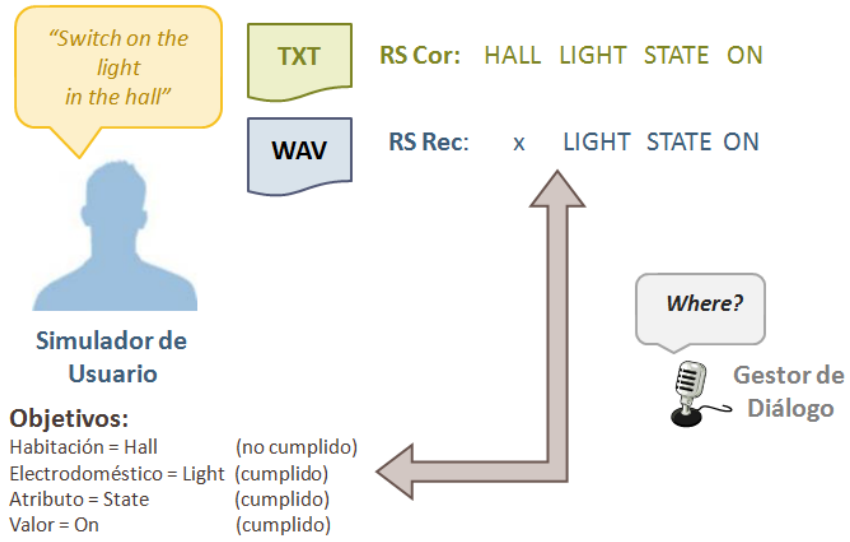



Figura 20: *Ejemplo turno de diálogo generado por el simulador de usuarios*

- **Siguiente turno de diálogo o fin de la interacción.**

El simulador, en su próximo turno, comprueba qué objetivos no tiene cumplidos y realiza los mismos pasos explicados anteriormente.

El diálogo acaba cuando todos los objetivos del simulador se han cumplido, dando lugar a una acción realizada y a un diálogo finalizado satisfactoriamente.

Si el diálogo se excede en un límite de turnos, se cancela y termina éste sin haber cumplido los objetivos. Esta situación se puede dar a causa de errores de reconocimiento en las frases del corpus elegidas, o bien, porque el sistema de diálogo no comprende correctamente alguno de los datos individuales (por ejemplo, habitación en que se desea realizar la acción).

A continuación se muestran algunos ejemplos de diálogos:

| *Ejemplo de diálogo que termina correctamente.* |
|---|
| #9<br>Goals:<br>0.- room=kitchen : False<br>1.- appliance=light : False<br>2.- attribute=state : False<br>3.- value=on : False<br><br>- User: Please switch on the light<br>- Mayordomo: Where?<br>- User: in the kitchen<br>- Mayordomo: You have changed to on the state of the light in the kitchen. |

*Ejemplo de diálogo que termina correctamente.*
*(*) Se muestran los objetivos del usuario, frases y representaciones semánticas.*

DIALOGUE #9:
GOAL[0] = 0.- room=kitchen : False
GOAL[1] = 1.- appliance=light : False
GOAL[2] = 2.- attribute=state : False
GOAL[3] = 3.- value=on : False
[0]

[FILE: CUSimUser\orden_esc99tan0fra2_20100223_13-46-55.RSCor ]
- User: Please switch on the light
RSRec={x light state on }
COR =Please switch on the light
RSCor={x light state on }
- Mayordomo: Where?

[1]
[FILE: CUSimUser\palabras_clave_esc98tan0fra3_20101108_20-11-43.RSCor ]
- User: in the kitchen
RSRec={kitchen light state on }
COR =In the kitchen
RSCor={kitchen x x x }
- Mayordomo: You have changed to on the state of the light in the kitchen.

*Ejemplo de diálogo que excede el límite de turnos de diálogo.*

#1
Goals:
0.- room=corridor : False
1.- appliance=piped music : False
2.- attribute=state : False
3.- value=on : False

- User: turn on the piped music
- Mayordomo: Where?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: turn the on hall
- Mayordomo: Sorry, could you repeat room?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?

*Ejemplo de diálogo en el que se excede el límite de turnos de diálogo (5).*
*(*) Se muestran los objetivos del usuario, frases y representaciones semánticas.*

DIALOGUE #1:
GOAL[0] = 0.- room=corridor : False
GOAL[1] = 1.- appliance=piped music : False
GOAL[2] = 2.- attribute=state : False
GOAL[3] = 3.- value=on : False

[0]
[FILE: CUSimUser\orden_esc99tan1fra16_20100223_19-59-10.RSCor ]
- User: turn on the piped music
RSRec={x piped_music state on }
COR =Good morning turn on the piped music
RSCor={x piped_music state on }
- Mayordomo: Where?

[1]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101110_13-19-26.RSCor ]
- User: in the hall
RSRec={hall piped_music state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[2]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101107_19-29-13.RSCor ]
- User: turn the on hall
RSRec={hall piped_music state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[3]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101110_13-19-26.RSCor ]
- User: in the hall
RSRec={hall piped_music state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[4]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101105_08-24-36.RSCor ]
- User: in the hall
RSRec={hall piped_music state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[5]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101110_13-34-47.RSCor ]

- User: in the hall
RSRec={hall piped_music state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[ERROR: dialogue limit has been exceeded.]

*Ejemplo de diálogo en el motor de reconocimiento rechaza la grabación y se produce un error de reconocimiento (5).*

#38
Goals:
0.- room=corridor : False
1.- appliance=light : False
2.- attribute=state : False
3.- value=on : False
-
- User: switch on the light
- Mayordomo: Where?
- User:
- Mayordomo: I haven't understood you. Could you repeat?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
- User: on the on hall
- Mayordomo: Sorry, could you repeat room?

*Ejemplo de diálogo en el motor de reconocimiento rechaza la grabación y se produce un error de reconocimiento (5).*
*(*) Se muestran los objetivos del usuario y las representaciones semánticas reconocidas y correctas*.

DIALOGUE #38:
GOAL[0] = 0.- room=corridor : False
GOAL[1] = 1.- appliance=light : False
GOAL[2] = 2.- attribute=state : False
GOAL[3] = 3.- value=on : False

[0]
[FILE: CUSimUser\orden_esc99tan0fra1_20100223_13-46-43.RSCor ]
- User: switch on the light
RSRec={x light state on }
COR =Switch on the light
RSCor={x light state on }
- Mayordomo: Where?

[1]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101114_01-35-01.RSCor ]
- User:
RSRec={x light state on }
COR =In the corridor
RSCor={corridor x x x }
[Recognition error]: CUSimUser\palabras_clave_esc98tan0fra1_20101114_01-35-01.RSCor
- Mayordomo: I haven't understood you. Could you repeat?

[2]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101110_13-19-26.RSCor ]
- User: in the hall
RSRec={hall light state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[3]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101107_22-19-05.RSCor ]
- User: in the hall
RSRec={hall light state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[4]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101116_13-14-42.RSCor ]
- User: in the hall
RSRec={hall light state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[5]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101114_20-28-23.RSCor ]
- User: on the on hall
RSRec={hall light state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?

[ERROR: dialogue limit has been exceeded.]

# 4.2 Análisis

## 4.2.1 Diagrama de clases

A continuación se muestra un diseño del diagrama de clases para el gestor del corpus y el simulador de usuarios.

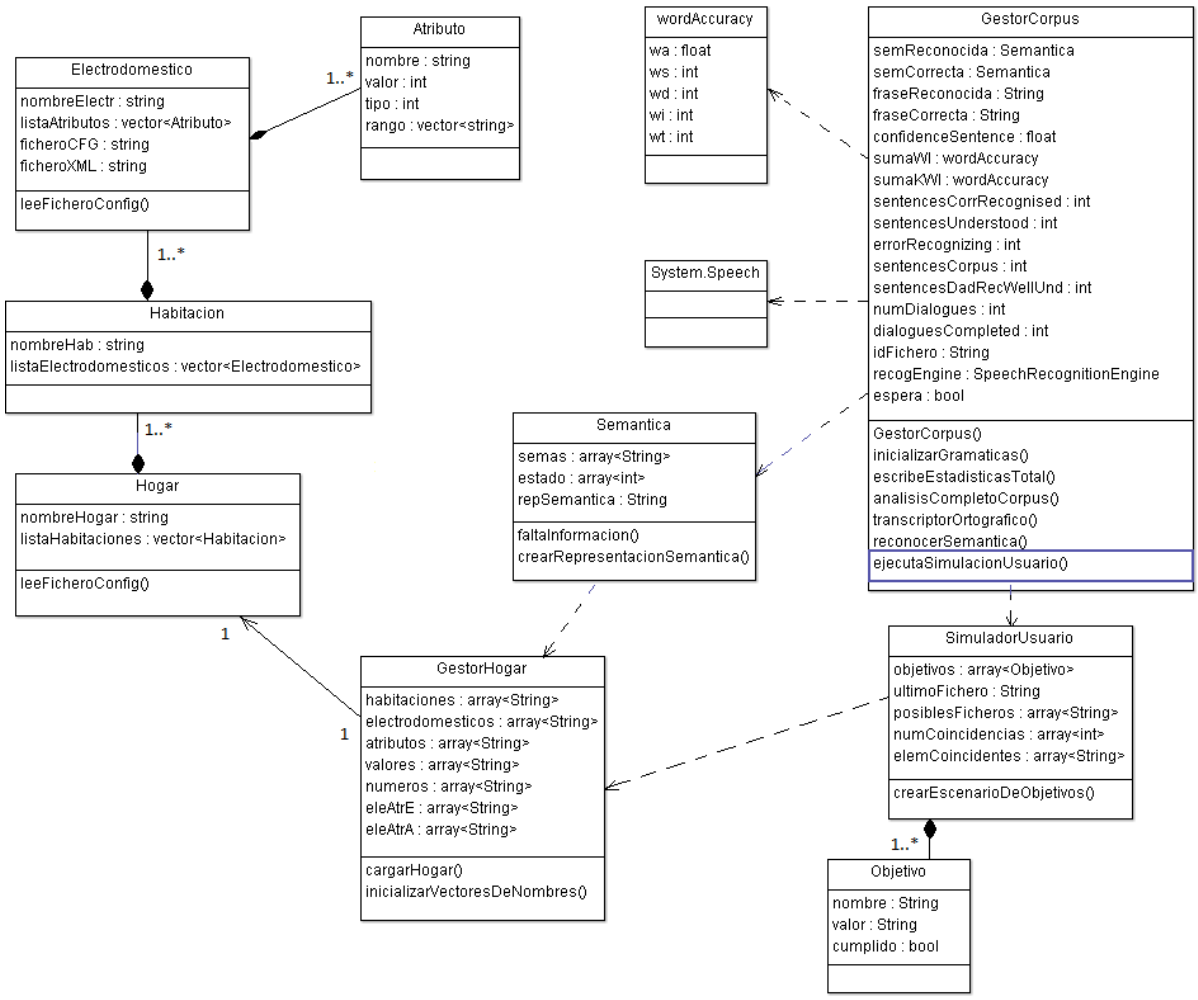



**Figura 21:** ***Diagrama de clases***

Por falta de espacio, sólo se muestran los métodos que se ha creído más relevante y/o utilizan instancias de las clases con las que están unidas mediante una relación.

# 4.3 Implementación

## 4.3.1 Librerías

En esta sección se indican las librerías usadas para la implementación, y qué función realizan en el sistema.

#### *System.Speech*

Necesaria para el funcionamiento del reconocedor de habla. Se han añadido las siguientes referencias a clases de dicha librería:

```
using namespace System::Speech::AudioFormat;
using namespace System::Speech::Synthesis;
using namespace System::Speech::Recognition;
using namespace System::Speech::Recognition::SrgsGrammar;
```

#### *Hogar*

Almacena toda la información de la configuración del hogar (ver clase Hogar en Sección 3.4).

```
#include "Hogar.h"

//Variable global
Hogar h;
```

### 4.3.2 Estructura wordAccuracy

Estructura de datos que contiene, dada una frase reconocida (salida del RAH), el número de palabras con respecto a la frase de referencia o frase correcta que han sido eliminadas, sustituidas e insertadas. También almacena, una vez calculada, la exactitud de palabra ("*word accuracy*") con respecto al total de palabras de la frase.

| Struct **wordAccuracy** | | |
|---|---|---|
| **Tipo** | **Nombre** | |
| float | wa | Exactitud de palabra (Word accuracy) |
| int | ws | Palabras sustituidas en la frase reconocida (Substituted words) |
| int | wd | Palabras eliminadas en la frase reconocida (Deleted words) |
| int | wi | Palabras insertadas en la frase reconocida (Inserted words) |
| int | wt | Número de palabras de la frase correcta o de referencia. |

Tabla 17: *Estructura wordAccuracy*

### 4.3.3 GestorHogar

Es la clase que contiene todos los nombres de valores, atributos, electrodomésticos, habitaciones y números del hogar.

**Miembros de la clase:**

| class **GestorHogar** | | |
|---|---|---|
| **Atributos (private)** | | |
| **Tipo** | **Nombre** | |
| array <String ^>^ | habitaciones | Contiene el nombre de todas las habitaciones del hogar. |
| array <String ^>^ | electrodomesticos | Contiene el nombre de todos los electrodomésticos del hogar. |
| array <String ^>^ | atributos | Contiene el nombre de todos los atributos del hogar. |
| array <String ^>^ | valores | Contiene el nombre de todos los valores del hogar. |
| array <String ^>^ | numeros | Contiene el nombre de todos los números del hogar. |
| array <String ^>^ | eleAtrE | Contienen pares electrodoméstico-atributo asociados entre sí |
| array <String ^>^ | eleAtrA | |

| **Métodos** | |
|---|---|
| GestorHogar() | Constructor de la clase. Llama al método que inicializa los vectores miembros de la clase. |
| cargarHogar() | Hace uso de la variable global h (clase Hogar). Lee los ficheros de configuración (electrodomésticos y hogar) y los almacena en esta variable. |
| String^ getHabitacion (int posicion) | Devuelve una cadena con el nombre de la habitación que hay en esa posición. |
| String^ getElectrodomestico (int posicion) | Devuelve una cadena con el nombre del electrodoméstico que hay en esa posición. |
| String^ getAtributo (int posicion) | Devuelve una cadena con el nombre del atributo que hay en esa posición. |
| String^ getValor (int posicion) | Devuelve una cadena con el nombre del valor que hay en esa posición. |
| String^ getEleAtrE(int posicion) | Devuelve una cadena con el nombre del electrodoméstico que hay en esa posición (asociado al atributo que hay en la misma posición en el vector eleAtrA). |
| String^ getEleAtrA (int posicion) | Devuelve una cadena con el nombre del atributo que hay en esa posición (asociado al electrodoméstico que hay en la misma posición en el vector eleAtrE). |
| String^ getNumero(int posicion) | Devuelve una cadena con el nombre del número que hay en esa posición. |
| void getArrayHabitaciones() | Muestra por pantalla (consola) el vector de nombres de habitaciones. |
| void getArrayElectrodomesticos() | Muestra por pantalla (consola) el vector de nombres de electrodomésticos. |
| void getArrayAtributos() | Muestra por pantalla (consola) el vector de nombres de atributos. |
| void getArrayValores() | Muestra por pantalla (consola) el vector de nombres de valores. |
| void getArrayEleAtr() | Muestra por pantalla (consola) el vector de nombres de pares asociados de electrodomésticos-atributos. |
| int numeroNumeros() | Devuelve la longitud del vector de números. |

| | |
|---|---|
| int numeroHabitaciones() | Devuelve la longitud del vector de habitaciones. |
| int numeroElectrodomesticos() | Devuelve la longitud del vector de electrodomésticos. |
| int numeroValores() | Devuelve la longitud del vector de valores. |
| int numeroAtributos() | Devuelve la longitud del vector de atributos. |
| int numeroEleAtrE() | Devuelve la longitud del vector de eleAtrE. |
| int numeroEleAtr A() | Devuelve la longitud del vector de eleAtrA. |
| void inicializarVectoresDeNombres() | (1)<br>NOTA: Versión 2.0 |

**Tabla 18:** ***Clase GestorHogar***

#### *(1) void inicializarVectoresDeNombres():*

Primero, la clase GestorHogar hace uso del método de clase "cargarHogar()" en el cual carga toda la información de los archivos de configuración en la variable global "h" (clase Hogar). Estos archivos de configuración, como se explicó en la clase Hogar y clases relacionadas, son "Home.cfg" (para el nombre de las habitaciones y electrodomésticos) y cada uno de los ficheros "*.cfg" de cada electrodoméstico (para atributos y valores).

A continuación, se recorren todos los elementos relacionados con esta variable para conseguir los nombres de todos los electrodomésticos, atributos, valores y habitaciones del hogar. También guarda las asociaciones existentes entre atributos y valores (como por ejemplo, "piped music" y "volume") y números (del cero al diez).

### 4.3.4 Semantica

Esta clase representa el significado o **representación semántica** contenida en una frase. Una representación semántica está formada por la información semántica o **semas** que aparecen en la frase procesada. Un sema es un par constituido por un significante y un significado: **{significante, significado}.**

Según sea la frase procesada, se obtendrá una representación semántica con un número distinto de semas, dependiendo de la información semántica existente en dicha frase.

En nuestro caso, los **significantes** son habitación, electrodoméstico, atributo y valor. Además, guardamos información de si estos significantes aparecen o no en la frase. El **significado** en nuestro caso es el **valor que toman los significantes**. Por ejemplo, para la frase: *"Switch on the light"* tenemos estos semas: {electrodomestico, light}, {atributo, state}, {valor, on}.

La **representación semántica de referencia** ha de obtenerse en función de la **frase** sin errores de RAH. De la misma manera, la **representación semántica obtenida** ha de obtenerse en función de la **frase reconocida**.

La representación semántica no es lo mismo que nuestro campo acción. El campo "**acción**" está formado por los cuatro campos: habitación, electrodoméstico, atributo y valor. Todos son importantes y necesarios ya que permiten la realización de una acción dentro del sistema de diálogo *Mayordomo*. Si falta uno de ellos, el sistema de diálogo preguntará por la información necesaria hasta completar todos los campos.

**Miembros de la clase:**

| class **Semantica** | | |
|---|---|---|
| **Atributos (private)** | | |
| **Tipo** | **Nombre** | |
| array <String ^>^ | semas | Vector con nombres de los semas (o significantes). |
| array <int>^ | estado | Vector de enteros, asociado al vector de semas o significantes.<br>0 = no ha sido encontrado en la frase.<br>1 = sí se ha encontrado. |
| String ^ | repSemantica | Cadena con la representación semántica (o significados). |
| **Métodos** | | |
| Semantica() | | Constructor de la clase. |
| ~Semantica() | | Destructor de la clase. |
| void limpiar() | | Misma función que el destructor de la clase: limpia el contenido de los atributos de la clase. |
| int tamSemanticaSemas() | | Devuelve el tamaño del vector semas. |
| int tamSemanticaEstatus() | | Devuelve el tamaño del vector estado. |
| array<String ^> ^ getArraySemas() | | Devuelve el vector semas. |
| array<String ^> ^ getArrayEstado() | | Devuelve el vector estado. |
| void addSemaEstado(String ^ sema, int estatus) | | Añade un nuevo sema con un estado. Si no existe, se incluye al final. Si existe, se sobreescribe el estado de dicho sema. |
| void setSema(int posicion, String ^sema) | | Sobreescribe el nombre del sema que se encuentra en la posición indicada. Ha de existir previamente. |
| void setEstado(int posicion, int estatus | | Sobreescribe el valor del estado que se encuentra en la posición indicada. Ha de existir previamente. |
| String ^ getSema(int posicion) | | Devuelve el sema que hay en la posición especificada. |
| int getEstado(int posicion) | | Devuelve el estado que hay en la posición especificada. |
| String ^ getRepSemantica() | | Devuelve una cadena con la representación semántica. |
| void setRepSemantica(String ^ cadena) | | Sobreescribe la representación semantic existente con la cadena especificada. |
| void addRepSemantica(String ^ cadena) | | Añade a la representación semántica existente la cadena deseada (al final). |
| void mostrarSemantica() | | Muestra por pantalla (consola) los vectores de semas y estado, junto con la representación semántica. |
| bool faltaInformacion(String ^sema) | | Devuelve true si el sema indicado tiene asociado el estado a 0. False en caso contrario (estado = 1). |
| String^ crearRepresentacionSemantica(String^ textoReconocido) | | (2)<br>Versión 2.0 |

Tabla 19: *Clase Semantica*

### *(2) String^ crearRepresentacionSemantica(String^ textoReconocido):*

Este método utiliza como entrada la salida del RAH y devuelve una cadena con la representación semántica, además de modificar el vector de semas y estado acorde con la información semántica encontrada en la frase.

Para crear una representación semántica, el método busca en el texto reconocido todos los significantes y significados posibles. En nuestro caso, para el dominio del hogar, los semas son: habitación, electrodoméstico, atributo y valor.

El método *faltaInformacion()* es utilizado para saber qué significante aún no ha sido encontrado. Así se evita buscar en el texto significantes que ya han sido encontrados en otras ejecuciones anteriores del método. (ver comprobarInformacion())

Primero, el significante **habitación** es buscado en el texto, comparando con la lista de habitaciones disponibles en el hogar. Si encuentra alguna habitación, almacena el significado (el nombre de la habitación) para la representación semántica y añade el sema correspondiente al vector (estado = 1). Además, busca si en vez de una habitación concreta el usuario se ha referido a todas las habitaciones, en cuyo caso el significado será "all". En caso de que ninguna de estas situaciones haya ocurrido, es porque ninguna habitación se ha encontrado en el texto, así que se marca como tal, añadiendo el sema con estado = 0 y con significado "x".

A continuación, hace lo mismo con el significante **electrodoméstico** buscando en la lista de electrodomésticos disponibles. En la frase reconocida además del nombre completo de un electrodoméstico, podemos encontrar partes o maneras abreviadas del mismo nombre (por ejemplo, "music" en vez de "piped music"). Por ello, el método busca también la existencia de partes del nombre de los electrodomésticos. Si encuentra algún significado (nombre del electrodoméstico), lo almacena para la representación semántica y añadirá el sema correspondiente al vector (estado = 1). Si no encuentra nada, se añade el sema con estado = 0 y el significado será "x".

Tras esto, el método actúa de la misma manera con el significante **atributo**, buscando en la lista de atributos de los electrodomésticos del hogar. Cuando encuentra un atributo de esta lista, comprueba si ha encontrado anteriormente un electrodoméstico. Si no se ha encontrado el significante electrodoméstico, entonces el método busca el par electrodoméstico-atributo correspondiente a el atributo reconocido (usando los vectores eleAtrE y eleAtrA). Si el atributo corresponde a un electrodoméstico único, se lo asocia y finaliza. Si corresponde a varios, el significado será "some". Si ha encontrado algún significado lo almacena para la representación semántica y añade el sema correspondiente al vector (estado = 1). Si no ha encontrado nada, se añade el sema con estado = 0 y con significado "x".

Para finalizar, para encontrar el **valor reconocido** el método busca en la frase reconocida cada uno de los nombres de lista de valores (asociados a los atributos de los electrodomésticos que tenemos en nuestro hogar). Además, como algunos atributos son numéricos, el método busca en el vector de números los números del cero al diez. A veces, un atributo se encuentra implícito en la frase de tal manera que sólo aparece alguno de los

valores que puede tomar dicho atributo (por ejemplo, en la frase "Switch on the light", cuyo atributo sería "state" que no aparece en la oración). Así, el método busca estos atributos implícitos de entre una lista de palabras (normalmente verbos o preposiciones) para poder asignar un valor a los significantes atributo y valor. Si encuentra el valor lo almacena para la representación semántica y añade el sema correspondiente al vector (estado = 1). Si no encuentra nada, se añade el sema con estado = 0 y con significado "x".

El método devuelve una cadena con el significado de los semas encontrados.

### 4.3.5 Objetivo

Esta clase contiene la descripción de un objetivo (nombre, valor y estado). Será usada en la clase SimuladorUsuario.

**Miembros de la clase:**

| class **Objetivo** | | |
|---|---|---|
| **Atributos (private)** | | |
| **Tipo** | **Nombre** | |
| String ^ | nombre | |
| String ^ | Valor | |
| Bool | cumplido | |
| **Métodos** | | |
| Objetivo() | Constructor de la clase. | |
| ~ Objetivo () | Destructor de la clase. | |
| String ^ getNombre() | Devuelve el nombre del objetivo. | |
| String ^ getValor() | Devuelve el valor del objetivo. | |
| bool getEstado() | Devuelve si el objetivo está cumplido (true) o no (false). | |
| void setNombre(String ^ name) | Sobreescribe el nombre del objetivo. | |
| void setValor(String ^ value) | Sobreescribe el valor del objetivo. | |
| void setEstado(bool estado) | Sobreescribe el estado del objetivo. | |

Tabla 20: *Clase Objetivo*

### 4.3.6 SimuladorUsuario

Esta clase representa la simulación de un usuario interaccionando con el sistema.

**Miembros de la clase:**

| class **SimuladorUsuario** | | |
|---|---|---|
| **Atributos (private)** | | |
| **Tipo** | **Nombre** | |
| array <Objetivo ^> ^ | Objetivos | Vector con objetos Objetivo. Lista de objetivos del simulador de usuario. |
| String ^ | ultimoFichero | Ultimo fichero usado en la simulación (sin extensión). |
| array <String ^> ^ | posiblesFicheros | Lista de posibles ficheros a usar en la simulación. |

| | | |
|---|---|---|
| array <int> ^ | numCoincidencias | Lista de número de coincidencias de los ficheros. |
| array <String ^> ^ | elemCoincidentes | Lista de los elementos que coinciden de cada fichero. |
| **Métodos** | | |
| SimuladorUsuario() | | Constructor de la clase. Inicializa la semilla para los números aleatorios usados. |
| ~ SimuladorUsuario () | | Destructor de la clase. |
| int getNumeroObjetivos() | | Devuelve el número de objetivos del vector. |
| int getNumeroObjetivosSinCumplir() | | Devuelve el número de objetivos sin cumplir (estado = 0) |
| String ^ getStringTodosObjetivosCumplidos() | | Devuelve una cadena con la lista de objetivos cumplidos (nombre=valor). |
| String ^ getStringValoresObjetivos() | | Devuelve una cadena con una lista de los valores de los objetivos. |
| void cumpleObjetivosTrasReconocimiento (String ^rs) | | (3) |
| void mostrarObjetivos() | | Muestra por pantalla (consola) la lista de objetivos del simulador (nombre=valor : estado). |
| array<String ^>^ mostrarObjetivosArray() | | Devuelve un vector de cadenas con el (nombre=valor : estado) de cada objetivo. |
| void seleccionarObjetivoAleatoriamente() | | (4) |
| void introducirObjetivoConsola(String ^ entradaConsola) | | Almacena los objetivos de la cadena (formato nombre=valor) sin cumplir (estado = false). |
| array<Objetivo^> ^transformaRSaObjetivos(String ^ repSem) | | Transforma una cadena con una representación semántica en un conjunto de objetivos. |
| void crearEscenariosDeObjetivos() | | (5) |
| void limpiarValoresTabla() | | Vacía los vectores usados para almacenar los posibles ficheros y coincidencias. |
| void limpiarValoresObjetivo() | | Elimina el contenido del vector de objetivos. |
| void crearListaFicherosCoincidencias(String ^ruta) | | (6) |
| void comparaObjetivos(array<Objetivo ^> ^objFichRSCOR, String ^nomFichero) | | (7) |
| String^ estrategiaBusquedaMaximosObjetivos() | | (8) |

**Tabla 21:** ***Clase SimuladorUsuario***

### *(3) void cumpleObjetivosTrasReconocimiento (String ^rs)*

Cambia el estado de los objetivos tras el reconocimiento de una frase y la posterior creación de la representación semántica. Necesita que se le pase como parámetro una cadena con la representación semántica (significado o valores encontrados en la frase reconocida).

El método marca como cumplido (estado = true) un objetivo, siempre y cuando este objetivo no había sido cumplido anteriormente, en el siguiente caso: el valor almacenado en la representación semántica es el mismo valor que tiene el objetivo.

### *(4) void seleccionarObjetivoAleatoriamente()*

Lee la lista de objetivos definidos previamente e incluidos en el fichero "ListaObjetivos.txt". Cada línea del fichero contiene una lista de objetivos para una simulación. Este método selecciona de manera aleatoria una línea con un conjunto de objetivos y los almacena en el vector de objetivos del simulador.

### *(5) void crearEscenariosDeObjetivos()*

Crea el fichero "ListaObjetivos.txt", en el cual, cada línea de fichero corresponde a un conjunto de objetivos para una simulación. El método, primero crea un objeto de la clase GestorHogar para tener acceso a todos los nombres de habitaciones, electrodomésticos, atributos y valores del hogar gracias al método de clase "inicializarVectoresDeNombres()". Una vez que obtiene los nombres, crea una combinación con todas las habitaciones, electrodomésticos, atributos y valores posibles.

NOTA: Ahora mismo sólo creamos combinaciones con objetivos cuyo atributo=STATE y valor=ON. El motivo es que nuestro corpus de frases grabadas sólo contiene frases para encender cualquier tipo de electrodoméstico.

### *(6) void crearListaFicherosCoincidencias(String ^ruta)*

Este método crea, a partir de los objetivos previamente definidos en el simulador, la tabla con los nombres de los ficheros cuya representación semántica coincide con dichos objetivos, además del número de coincidencias y los significantes en común. Es decir, rellena los vectores "posiblesFicheros" "numCoincidencias" y "elemCoincidentes". Para ello, ha de comparar los objetivos que tiene el simulador de usuario con las representaciones semánticas correctas de las frases del corpus (archivos *.RSCor), los cuales se encuentran en la ruta que se recibe como parámetro. Si no se le pasa ninguna ruta, tomará la ruta por defecto "CorpusUtterances\".

Básicamente, lee cada uno de los ficheros ".RSCor" de la carpeta, obteniendo la representación semántica. A continuación, transforma esta representación (con "transformaRSaObjetivos()" ) en un conjunto de objetivos, para facilitar la comparación con los objetivos del simulador. Tras esto, pasa comparar con el método "comparaObjetivos()" explicado a continuación.

### *(7) void comparaObjetivos(array<Objetivo ^> ^objFichRSCOR, String ^nomFichero)*

Este método compara los objetivos actuales del simulador de usuario con los objetivos que se le pasan como parámetro, correspondientes a la representación semántica del fichero actual "nomFichero" con extensión ".RSCor".

La comparación se realiza entre los objetivos no cumplidos del simulador de usuario. Se produce una coincidencia cuando el nombre y el valor del objetivo no cumplido del usuario es igual que el nombre y valor del objetivo de la semántica del fichero. Dada esta situación, el simulador almacena en el vector "posiblesFicheros" el nombre del fichero "nomFichero", junto

al número de objetivos no cumplidos que coinciden, en el vector “numCoincidencias”, más el nombre de esos objetivos, en el vector “elemCoincidentes”.

#### *(8) String ^ estrategiaBusquedaMaximosObjetivos()*

De todos los ficheros del vector “posiblesFicheros”, este método se centra en los ficheros con más coincidencias (información almacenada en “numCoincidencias”) y elige uno de ellos aleatoriamente. Este método devuelve el fichero seleccionado, con extensión “.RSCor”.

### 4.3.7 GestorCorpus

Esta clase realiza la evaluación del sistema, gestionando el corpus de frases, aplicando las herramientas anteriormente explicadas y obteniendo los resultados vistos al inicio de la sección.

**Miembros de la clase:**

| class **GestorCorpus** | | |
|---|---|---|
| **Atributos (private)** | | |
| **Tipo** | **Nombre** | |
| Semantica | semReconocida | Objeto que almacena la representación semántica reconocida. |
| Semantica | semCorrecta | Objeto que almacena la representación semántica de referencia. |
| String ^ | fraseReconocida | Cadena con la frase reconocida. |
| String ^ | fraseCorrecta | Cadena con la frase correcta. |
| Float | confidenceSentence | Nivel de confianza de la frase reconocida. |
| wordAccuracy * | sumaWI | Contiene la suma de wa, ws, wd, wi y wt obtenidas hasta el momento. |
| wordAccuracy * | sumaKWI | Contiene la suma de kwa, kws, kwd, kwi y kwt obtenidas hasta el momento. |
| Int | sentencesCorrRecognised | Número de frases correctamente reconocidas. |
| Int | sentencesUnderstood | Número de frases correctamente entendidas. |
| Int | errorRecognizing | Número de errores de reconocimiento (frases no reconocidas). |
| Int | sentencesCorpus | Total de frases analizadas. |
| int | sentencesBadRecWellUnd | Frases mal reconocidas pero correctamente entendidas. |

| | | |
|---|---|---|
| | | (Implicit Recovery) |
| Int | numDialogues | Número de diálogos. |
| Int | dialoguesCompleted | Dialogos finalizados correctamente (Task completion) |
| String ^ | idFichero | Fichero en uso. |
| SpeechRecognitionEngine^ | recogEngine | Motor de reconocimiento del habla. |
| Bool | Espera | Variable que indica que el motor está ocupado reconociendo (= true). |

| **Métodos** | |
|---|---|
| GestorCorpus() | Constructor de la clase. Crea instancias, define eventos para el motor y llama al método inicializarGramaticas() para que esté listo para el reconocimiento. |
| void setIdFichero(String ^id) | Sobreescribe el nombre del fichero en uso. |
| String ^ getFraseReconocida() | Devuelve la frase reconocida por el motor de reconocimiento. |
| String ^ getFraseCorrecta() | Devuelve la frase correcta (sacada de la transcripción ortográfica). |
| bool esperandoReconocedor() | Devuelve true si el motor está ocupado reconociendo. False si ha terminado. |
| void limpiaValores() | Elimina el contenido de fraseReconocida y fraseCorrecta. |
| int getNumErrorReconocimiento() | Devuelve el número de errores de reconocimiento. |
| void incrNumErrorReconocimiento() | Incrementa en uno el número de errores de reconocimiento. |
| void incrNumDialogosCompletados() | Incrementa en uno el número de diálogos completados correctamente. |
| void incrNumRecuperacionesImplicitas() | Incrementa en uno el número de recuperaciones implícitas (número de frases mal reconocidas pero entendidas bien). |
| int getNumDialogosCompletados() | Devuelve el número de diálogos completados correctamente. |
| int getNumRecuperacionesImplicitas() | Devuelve el número de recuperaciones implícitas, o frases mal reconocidas pero entendidas bien. |
| void setNumDialogues(int numDial) | Sobreescribe el número de diálogos a simular. |
| int getNumDialogos() | Devuelve el número de diálogos a simular. |
| void incrNumFrasesEntendidas() | Incrementa en uno el número de frases entendidas correctamente. |
| void incrNumFrasesCorrectas() | Incrementa en uno el número de frases correctamente reconocidas. |
| void incrNumFrasesCorpus() | Incrementa en uno el número de frases analizadas del corpus de frases. |
| void setNumFrasesEntendidas(int num) | Sobreescribe el número de frases correctamente entendidas. |
| void setNumFrasesCorrectas(int num) | Sobreescribe el número de frases |

| | |
|---|---|
| | correctamente reconocidas. |
| void setNumFrasesCorpus(int num) | Sobreescribe el número de frases analizadas del corpus. |
| int getNumFrasesEntendidas() | Devuelve el número de frases correctamente entendidas. |
| int getNumFrasesCorrectas() | Devuelve el número de frases correctamente reconocidas. |
| int getNumFrasesCorpus() | Devuelve el número de frases analizadas del corpus. |
| void actualizarSumaWA(float wa, int ws, int wd, int wi, int wt) | Actualiza la suma de wa, ws, wd, wi y wt (utilizada para los totales). |
| void limpiarSumaWA() | Pone los valores a cero. |
| void limpiarSumaKWA() | Pone los valores a cero. |
| void actualizarSumaKWA(float wa, int ws, int wd, int wi, int wt) | Actualiza la suma de kwa, kws, kwd, kwi y wt (utilizada para los totales). |
| void inicializarGramaticas() | (9) |
| void trazaCambiosReconocedor(String ^ nombreFichero, RecognitionResult ^resultado) | Escribe en el fichero "nombreFichero" el resultado del reconocimiento, con fecha y hora, alternativas y nivel de confianza. Usado en los eventos. |
| void trazaGestorCorpus(wordAccuracy wordInfo, int tipo) | Escribe en el fichero "TrazaGestorCorpus.txt" tanto la frase reconocida como la correcta y el cálculo de WA (si tipo = 0) o representación semántica reconocida y de referencia y el cálculo de KWA (tipo = 1). |
| void obtieneFraseCorrecta(String ^ nombreFichero) | (10) |
| bool obtieneRSCorrecta() | Si existe la frase correcta, calcula y guarda la representación semántica de referencia en el atributo "semCorrecta" llamando al método de la clase Semantica "crearRepresentacionSemantica()".<br>NOTA: Reemplaza los tabuladores por espacios. |
| void obtieneFraseReconocida(String ^ nombreFichero) | (11) |
| bool obtieneRSReconocida() | Si existe la frase correcta, calcula y guarda la representación semántica de referencia en el atributo "semReconocida" llamando al método de la clase Semantica "crearRepresentacionSemantica()".<br>NOTA: Reemplaza los tabuladores por espacios. |
| void reconoceFrase(String ^ nombreFichero) | (12) |
| void recogEngine_AudioStateChanged (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla |
| void recogEngine_SpeechRecognized (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: reconoce habla. |

| | |
|---|---|
| void recogEngine_RecognizeCompleted (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: reconocimiento de habla completado. (13) |
| void recogEngine_ SpeechHypothesized (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: hipótesis del reconocimiento.<br>NOTA: Guarda información en "SH.txt" |
| void recogEngine_AudioSignalProblem Occurred (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: problemas con la señal de audio. |
| void recogEngine_SpeechDetected (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: detectado el inicio del reconocimiento del habla. |
| Void recogEngine_ SpeechRecognition Rejected (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e) | Evento del reconocedor de habla: reconocimiento de habla rechazado.<br>NOTA: Guarda información en "SRR.txt" |
| array<String ^>^ formatearFrase(String ^ frase) | Devuelve en un vector la cadena "frase" formateada: sin signos de puntuación y en mayúscula. |
| float accuracy(int ws, int wd, int wi, int wtotalFraseCorrecta) | Calcula y devuelve la precision de palabra (wa), con los parámetros ws, wd, wi y wt. |
| void analizaWordAccuracy(wordAccuracy & wordInfo) | (14) |
| void corrigeSemanticas() | (15) |
| void analizaKeywordAccuracy (wordAccuracy & keywordInfo) | (16) |
| void analizaKeywordAccuracySU(wordAccuracy & keywordInfo) | (17) |
| void calculaErroresKWA(wordAccuracy & keywordInfo, String ^ rsCor, String ^ rsRec) | Calcula kwd, kws, kwi y kwt. |
| void calculaErroresWA(wordAccuracy & wordInfo, String ^ fCor, String ^ fRec) | Calcula wd, ws, wi y wt. |
| void escribeEvaluacion(array<String ^>^ co, array<String ^>^ re, array<String ^>^ ev, wordAccuracy wordInfo) | Escribe en el fichero "TrazaEvaluacionFallos.txt". |
| void escribeEstadisticas(wordAccuracy wInfo, wordAccuracy kwInfo) | Escribe en el fichero "Evaluacion.txt" los datos obtenidos para el Word accuracy y el Keyword accuracy. |
| void escribeEstadisticasTotal(String ^ ruta) | Calcula medidas para la evaluación haciendo uso de los atributos de la clase. Las almacena en el fichero "EvaluacionTotal.txt". |
| String ^comprobarInformacionObjetivos(bool faltaInfo, String ^valoresObjetivos) | (18) |

| | |
|---|---|
| bool utterUserSentence(String ^ficheroRSCOR) | (19) |
| String^ mezclaRepresentacionesSemanticas(String ^rsNueva,String ^rsOld) | Combina dos representaciones semánticas. |
| void analisisCompletoCorpus(String ^ruta) | (20) |
| void transcriptorOrtografico() | Ejecuta el método “transcribir()” para todos los ficheros de la ruta “CorpusUtterances\”. |
| void transcribir(String^ nombreFicheroWav, String^ ruta) | (21) |
| void reconoceSemantica() | Ejecuta el método “buscaRS()” para todos los ficheros de la ruta “CorpusUtterances\”. |
| void buscaRS(String^ nombreFicheroTXT) | (22) |
| void escribeEstadisticasSimulador(wordAccuracy wInfo, wordAccuracy kwInfo) | Escribe en el fichero “EvaluacionSinUser.txt” información sobre WA y KWA. |
| void escribeEstadisticasSimuladorDialogo(wordAccuracy *wInfo, wordAccuracy *kwInfo) | Escribe los totales de KWA y WA de cada diálogo simulado. Las almacena en el fichero “EvaluacionSimUser.txt”. |
| void escribeEstadisticasSimuladorTotal(String ^ ruta) | Calcula medidas para la evaluación haciendo uso de los atributos de la clase. Las almacena en el fichero “EvaluacionTotal-UserSimulator.txt”. |
| void ejecutaSimulacionUsuario(int num_dialogos) | (23) |

**Tabla 22:** ***Clase GestorCorpus***

### *(9) void inicializarGramaticas()*

Este método crea una instancia a una clase GestorHogar, para cargar en la variable global “h” toda la información necesaria. En este caso, el método necesita el nombre de los ficheros de gramáticas XML de cada electrodoméstico, para cargarlas en el motor (recogEngine->LoadGrammar()) y usarlas en el reconocimiento. Hace uso de las clases de System.Speech: Grammar y SrgsDocument.

### *(10) void obtieneFraseCorrecta(String ^ nombreFichero)*

En este método, tras comprobar que existe el fichero pasado como parámetro y comprobar que tiene extensión “.txt”, se lee el contenido que corresponde a la representación semántica de referencia (asociada al fichero de audio con ese mismo nombre y extensión “.wav”). NOTA: Elimina los signos de puntuación (comas y puntos) para un posterior formateo.

### *(11) void obtieneFraseReconocida(String ^ nombreFichero)*

En este método, tras comprobar que existe el fichero pasado como parámetro y comprobar que tiene extensión “.wav”, se llama al método “reconoceFrase()”. En caso que el

reconocedor encuentre problemas (InvalidOperationException) y lance una excepxión, ésta será capturada aquí.

*(12) void reconoceFrase(String ^ nombreFichero)*

El método selecciona el fichero de audio ".wav" pasado como parámetro como entrada del reconocedor (System.Speech: recogEngine->SetInputToWaveFile(nombreFichero)). Además, pone el atributo "espera" a true para que no se vuelva a llamar al reconocedor mientras éste está trabajando: recogEngine->RecognizeAsync(RecognizeMode::Multiple).

*(13) void recogEngine_RecognizeCompleted (System::Object^ sender, System::Speech::Recognition::AudioStateChangedEventArgs ^ e)*

Si el resultado del reconocimiento (e->Result) es nulo, significa que ha habido un fallo en el reconocimiento (porque la confianza a la hora de elegir entre las hipótesis está por debajo del umbral), y se escribe el nombre del fichero en el archivo "FalloReconocimiento.txt".

En caso contrario, ha habido éxito en el reconocimiento, con lo cual el texto del resultado (e->Result->Text) será lo que se guarde como el atributo **fraseReconocida**. Además, el resultado nos informa de la confianza de la frase resultado (e->Result->Confidence), que se guarda en el atributo **confidenceSentence**. Además, este evento escribe en el fichero "SRC.txt" toda la información con respecto al resultado del reconocimiento, gracias al método "trazaCambiosReconocedor()". Para finalizar, se cambia el valor del atributo "espera" a false para permitir más llamadas al reconocedor.

*(14) void analizaWordAccuracy(wordAccuracy & wordInfo)*

Obtiene la exactitud de palabras (o Word Accuracy) y todos los datos necesarios para su cálculo. Primero, dada una estructura wordAccuracy sin información (que será el parámetro devuelto), el método llama a calculaErroresWA() con dicha estructura, la cual se rellenará con las palabras sustituidas (ws), borradas (wd) e insertadas (wi) comparando la fraseCorrecta y la fraseReconocida. Tras conseguir esta información, llama al método accuracy() para obtener la exactitud de palabras.

*(15) void corrigeSemanticas()*

Modifica los dos atributos de la clase Semantica que contiene la clase: semReconocida y semCorrecta. Es el paso previo al cálculo de la exactitud de conceptos (o keyword accuracy).

Trocea las cadenas de ambas representaciones semánticas, y tomando como referencia siempre la representación semántica de referencia, si en ambas semánticas falta el mismo sema (estado = 0), no se incluirá ningún significado o valor (en este caso, "x") en las cadenas que contienen las representaciones semánticas de referencia y reconocida.

*(16) void analizaKeywordAccuracy (wordAccuracy & keywordInfo)*

Obtiene la exactitud de conceptos (Keyword Accuracy) y todos los datos necesarios para su cálculo. Primero, llama al método corrigeSemanticas() para modificar y adecuar ambas

representaciones semánticas al método para calcular los errores para el KWA. Dada una estructura wordAccuracy sin información (que será el parámetro devuelto), el método llama a calculaErroresKWA() con dicha estructura, la cual se rellenará con las palabras sustituidas (kws), borradas (kwd) e insertadas (kwi) comparando la representación semántica de referencia y la reconocida. Tras conseguir esta información, llama al método accuracy() para obtener la exactitud de conceptos.

### *(17) analizaKeywordAccuracySU (wordAccuracy & keywordInfo)*

Igual que el método anterior, pero sin llamar al método de corregirSemanticas().

### *(18) String ^ comprobarInformacionObjetivos(bool & faltaInfo, String ^valoresObjetivos)*

Este método comprueba si falta por cumplir algún objetivo del simulador de usuario, cuyos valores se pasan como parámetro en la cadena "valoresObjetivos", comparando la representación semántica reconocida (semReconocida).

Falta información en alguno de los siguientes casos:

i) En la representación semántica reconocida hay información de un significante reconocido, que coincide con uno de los objetivos, pero tienen significados diferentes. Ej.: para habitación, el objetivo es bedroom y ha reconocido bathroom.
ii) En la representación semántica, no aparece algún significante.

Para el primer caso, el método pide al simulador de usuario que vuelva a repetir el significante. Para el segundo caso, preguntará por cada significante que falta.

Devuelve faltaInfo = true si falta información que obtener para cumplir algún objetivo, y en el parámetro "pregunta" las preguntas necesarias que hacer al simulador. Devuelve false en caso contrario.

### *(19) bool utterUserSentence(String ^ficheroRSCOR)*

Este método, toma como parámetro de entrada un fichero con extensión ".RSCor" (que es el elegido por el simulador de usuario para cumplir sus objetivos). A continuación, llama a los métodos "obtieneFraseReconocida()" y "obtieneFraseCorrecta()" con el mismo fichero con extensión ".wav" y ".txt" respectivamente. Tras esperar a que el reconocedor de habla termine de reconocer, llama a los métodos "obtieneRSReconocida()" y "obtieneRSCorrecta()".

Si ha encontrado representación semántica reconocida, llama al método "mezclaRepresentacioneSemanticas()" para combinar la anterior representación semántica con la obtenida ahora.

Devuelve true si no se produce ningún error en el reconocimiento de la frase, ni en el cálculo de la representación semántica reconocida. False si hay problemas.

### *(20) void analisisCompletoCorpus(String ^ruta)*

Dada una ruta de una carpeta que contiene todos los ficheros que forman el corpus de frases, analiza todos los ficheros, obtiene tanto la frase reconocida y la representación semántica reconocida como la frase correcta y la representación semántica de referencia. Tras esto, calcula una serie de medidas estadísticas para la evaluación del sistema.

Este método hace llamadas a los siguientes métodos: primero, almacena el fichero actual que está siendo analizado con el método setIdFichero(), tras esto, ejecuta el método obtieneFraseReconocida() para el fichero con extensión ".wav" y obtieneFraseCorrecta() con el fichero ".txt", mientras, esperandoReconocedor() hasta que el motor acaba su trabajo. A continuación, llama a los métodos obtieneRSReconocida() y obtieneRSCorrecta()para obtener ambas representaciones semánticas.

Si el reconocimiento ha tenido éxito (tenemos una frase reconocida y su correspondiente representación semántica), pasamos a calcular la exactitud de palabras (Word accuracy) y de conceptos (Keyword accuracy) con los métodos analizaWordAccuracy() y analizaKeywordAccuracy(). Después, actualiza la suma de KWA y de WA con los métodos actualizarSumaWA() y actualizarSumaKWA().

Si la exactitud de palabras es perfecta (WA = 100%), llama al método incrNumFrasesCorrectas() aumentando así el número de frases perfectamente reconocidas. Si la exactitud de conceptos es perfecta (KWA = 100%), ejecuta el método incrNumFrasesEntendidas(), aumentando el número de frases perfectamente entendidas. En el caso que no haya una exactitud de palabras perfecta (WA < 100%) pero sí una exactitud de conceptos perfecta (KWA = 100%), se produce una recuperación implícita, ya que a pesar de que el sistema no ha reconocido correctamente, ha conseguido interpretar completamente el significado de la frase. De esta manera, el método llama a incrNumRecuperacionesImplicitas().

Si el reconocimiento ha sido fallido, aumenta el número de errores de reconocimiento llamando al método incrNumErroReconocimiento().

Tras analizar todos los ficheros de la carpeta, llama al método escribeEstadisticasTotal() para hacer los cálculos y escribir en el fichero "EvaluacionTotal.txt" las medidas de evaluación.

### *(21) void transcribir(String^ nombreFicheroWav, String^ ruta)*

Este método hace uso del fichero "Correspondencias.txt", que contiene una tabla en la que en la primera columna aparecen los códigos incluidos en los nombres de los ficheros del corpus y en la segunda columna, la transcripción correspondiente a ese código. Para el fichero dado como parámetro, busca su transcripción ortográfica correspondiente a partir del nombre, y tras esto, crea un nuevo fichero, cuyo nombre es el mismo que el que se pasó como parámetro pero con extensión ".txt" y cuyo contendido es la transcripción encontrada.

### *(22) void buscaRS(String^ nombreFicheroTXT)*

Este método lee la transcripción ortográfica creada anteriormente (".txt") y la formatea para luego llamar al método crearRepresentacionSemantica(). De esta manera,

obtiene la representación semántica de referencia y la almacena en un fichero con el mismo nombre y extensión ".RSCor".

### *(23) void ejecutaSimulacionUsuario(int num_dialogos)*

Este método ejecuta el simulador de usuario para crear "num_dialogos" diálogos entre él y el sistema. En cada diálogo, el simulador llama a "seleccionarObjetivosAleatoriamente()" y luego al método "crearListaFicherosCoincidencias()", para luego, ejecutar "estrategiaBusquedaMaximosObjetivos()" y seleccionar el fichero (aleatoriamente) que cumple más objetivos. A continuación, el gestor llama a "utterUserSentence()" con el fichero seleccionado, obteniendo así frase reconocida, frase correcta, representación semántica reconocida y representación semántica de referencia.

Si el reconocimiento ha tenido éxito (tenemos una frase reconocida y su correspondiente representación semántica), pasamos a calcular la exactitud de palabras (Word accuracy) y de conceptos (Keyword accuracy) con los métodos analizaWordAccuracy() y analizaKeywordAccuracy(). Después, actualiza la suma de KWA y de WA con los métodos actualizarSumaWA() y actualizarSumaKWA().

Si la exactitud de palabras es perfecta (WA = 100%), llama al método incrNumFrasesCorrectas() aumentando así el número de frases perfectamente reconocidas. Si la exactitud de conceptos es perfecta (KWA = 100%), ejecuta el método incrNumFrasesEntendidas(), aumentando el número de frases perfectamente entendidas. En el caso que no haya una exactitud de palabras perfecta (WA < 100%) pero sí una exactitud de conceptos perfecta (KWA = 100%), se produce una recuperación implícita, ya que a pesar de que el sistema no ha reconocido correctamente, ha conseguido interpretar completamente el significado de la frase. De esta manera, el método llama a incrNumRecuperacionesImplicitas().

El siguiente paso es comprobar qué información falta para que el simulador de usuario cumpla todos los objetivos definidos. Llama al método "comprobarInformacionObjetivos()" para ver que significantes faltan. Además, llama a "cumpleObjetivosTrasReconocimiento()" para cumplir los ya reconocidos.

Si no falta información, el gestor confirmará al simulador de usuario que ha completado la acción correspondiente a sus objetivos definidos e incrementa el número de diálogos completados con "incrNumDialogosCompletados()".

Si el reconocimiento ha sido fallido, aumenta el número de errores de reconocimiento llamando al método "incrNumErroReconocimiento()".

Por último, hay que indicar que el diálogo tiene un número límite de iteraciones, necesario en caso de que el gestor no entienda lo que quiere el simulador durante demasiadas iteraciones, cortando así el diálogo para pasar al siguiente.

El resultado de los diálogos se almacena en el fichero "Dialogues.txt". Tras simular todos los diálogos, llama al método "escribeEstadisticasSimuladorTotal()" para hacer los cálculos y escribir en el fichero "EvaluacionTotal-UserSimulator.txt" las medidas de evaluación.

## 4.4 Ficheros de resultados

### 4.4.1 Gestor del Corpus

*i) Correspondencias.txt*

Dónde se usa: en transcribir()

Motivo: contiene las correspondencias entre código y frase correcta.

| | |
|---|---|
| orden_esc99tan0fra1 | Switch on the light |
| orden_esc99tan0fra2 | Please, switch on the light |
| orden_esc99tan0fra3 | Hi, switch on the light |
| orden_esc99tan0fra4 | Hello, switch on the light |
| orden_esc99tan0fra5 | Good morning, switch on the light |
| orden_esc99tan0fra6 | Good afternoon, switch on the light |
| orden_esc99tan0fra7 | Turn on the light |
| orden_esc99tan0fra8 | Please, turn on the light |
| orden_esc99tan0fra9 | Hi, turn on the light |
| orden_esc99tan0fra10 | Hello, turn on the light |
| orden_esc99tan0fra11 | Good morning, turn on the light |
| orden_esc99tan0fra12 | Good afternoon, turn on the light |
| orden_esc99tan0fra13 | Switch on the washing machine |
| palabras_clave_esc98tan1fra17 | Switch on |
| palabras_clave_esc98tan1fra18 | Switch off |
| palabras_clave_esc98tan1fra19 | Turn up |
| palabras_clave_esc98tan1fra20 | Turn down |
| palabras_clave_esc98tan1fra21 | Turn on |
| palabras_clave_esc98tan1fra22 | Turn off |

*ii) TrazaGestorCorpus.txt*

Dónde se usa: en trazaGestorCorpus()

Motivo: para cada fichero del corpus, escribe la frase reconocida y frase correcta (en caso de que la WA no sea 100%) o la representación semántica reconocida y de referencia (en caso de que KWA no sea 100%).

```
[\orden_esc99tan0fra10_20100223_11-38-34]
        Recognized sentence: turn on nine
        Correct sentence: Hello turn on the light
WA=40 WS=1   WD=2   WI=0    WT=5
[\orden_esc99tan0fra10_20100223_11-38-34]
        Recognized SR: x state on
        Correct SR: light state on
KWA=66,66666 KWS=0  KWD=1 KWI=0  KWT=3
[\orden_esc99tan0fra10_20100223_13-48-23]
        Recognized sentence: Hello the on tonight
        Correct sentence: Hello turn on the light
WA=40 WS=2   WD=1   WI=0    WT=5
```

```
[\orden_esc99tan0fra10_20100223_13-48-23]
        Recognized SR: x state on
        Correct SR: light state on
KWA=66,66666 KWS=0 KWD=1 KWI=0 KWT=3
[\orden_esc99tan0fra11_20100223_11-38-46]
        Recognized sentence: Good morning the on tonight
        Correct sentence: Good morning turn on the light
WA=50 WS=2 WD=1 WI=0 WT=6
[\orden_esc99tan0fra11_20100223_11-38-46]
        Recognized SR: x state on
        Correct SR: light state on
KWA=66,66666 KWS=0 KWD=1 KWI=0 KWT=3
[\orden_esc99tan0fra11_20100223_13-48-34]
        Recognized sentence: Good morning the light
        Correct sentence: Good morning turn on the light
WA=33,33333 WS=2 WD=2 WI=0 WT=6
[\orden_esc99tan0fra11_20100223_13-48-34]
        Recognized SR: light x x
        Correct SR: light state on
KWA=33,33333 KWS=0 KWD=2 KWI=0 KWT=3
```

*iii) SRC.txt*

Dónde se usa: en el evento RecognitionCompleted()

Motivo: guarda la información obtenida en el reconocimiento (nombre fichero, frase reconocida y alternativas posibles).

```
# DATE: 27/11/2010      TIME: 13:37:58.1132047
File: CorpusUtterances\orden_esc99tan0fra3_20100223_11-11-49
        Recognized sentence: Hi switch on the light [0,985428]
        Alternates:
                [0] Hi switch on the light [0,985428]
# DATE: 27/11/2010      TIME: 13:38:00.9992098
File: CorpusUtterances\palabras_clave_esc98tan0fra7_20101116_13-15-45
        Recognized sentence: on [0,9559071]
        Alternates:
                [0] on [0,9559071]
                [1] on [0,9559071]
# DATE: 27/11/2010      TIME: 13:38:03.9632150
File: CorpusUtterances\palabras_clave_esc98tan1fra17_20101110_13-41-58
        Recognized sentence: switch on [0,9546555]
        Alternates:
                [0] switch on [0,9546555]
                [1] switch on [0,9546555]
                [2] switch on [0,9546555]
                [3] switch on [0,9546555]
                [4] switch on [0,9546555]
```

*iv) SH.txt*

Dónde se usa: en el evento SpeechHypothesized()

Motivo: guarda la información obtenida en el reconocimiento (nombre fichero, frase reconocida y alternativas posibles).

```
# DATE: 27/11/2010      TIME: 13:44:09.3314567
File:  CorpusUtterances\orden_esc99tan1fra7_20100223_19-57-03
        Recognized sentence: Please [0,7138512]
        Alternates:
                [0] Please [0,7138512]
# DATE: 27/11/2010      TIME: 13:44:09.3314567
File:  CorpusUtterances\orden_esc99tan1fra7_20100223_19-57-03
        Recognized sentence: Please switch [0,9809454]
        Alternates:
                [0] Please switch [0,9809454]
# DATE: 27/11/2010      TIME: 13:44:09.3314567
File:  CorpusUtterances\orden_esc99tan1fra7_20100223_19-57-03
        Recognized sentence: Please switch on the [0,9135746]
        Alternates:
                [0] Please switch on the [0,9135746]
# DATE: 27/11/2010      TIME: 13:44:09.3314567
File:  CorpusUtterances\orden_esc99tan1fra7_20100223_19-57-03
        Recognized sentence: Please switch on the piped [0,8884315]
        Alternates:
                [0] Please switch on the piped [0,8884315]
# DATE: 27/11/2010      TIME: 13:44:09.3314567
File:  CorpusUtterances\orden_esc99tan1fra7_20100223_19-57-03
        Recognized sentence: Please switch on the piped music [0,9769396]
        Alternates:
                [0] Please switch on the piped music [0,9769396]
# DATE: 27/11/2010      TIME: 13:44:09.8930577
File:  CorpusUtterances\palabras_clave_esc98tan0fra1_20101107_19-29-13
        Recognized sentence: in [0,002000033]
        Alternates:
                [0] in [0,002000033]
# DATE: 27/11/2010      TIME: 13:44:09.8930577
File:  CorpusUtterances\palabras_clave_esc98tan0fra1_20101107_19-29-13
        Recognized sentence: in the [0,7642781]
        Alternates:
                [0] in the [0,7642781]
```

*v) SRR.txt*

Dónde se usa: en el evento SpeechRecognitionRejected()

Motivo: guarda la información obtenida en el intento de reconocimiento (nombre fichero, frase reconocida y alternativas posibles con nivel de confianza).

```
# DATE: 27/11/2010      TIME: 13:38:09.9692256
File:  CorpusUtterances\palabras_clave_esc98tan0fra2_20101105_08-24-47
        Recognized sentence:  [0]
        Alternates:
                [0]  [0]
# DATE: 27/11/2010      TIME: 13:38:13.4324316
File:  CorpusUtterances\palabras_clave_esc98tan0fra1_20101108_20-11-23
        Recognized sentence: in the hall [0,4191467]
        Alternates:
                [0] in the hall [0,4191467]
                [1] in the hall [0,4191467]
                [2] in the hall [0,4191467]
# DATE: 27/11/2010      TIME: 13:38:16.9580378
File:  CorpusUtterances\palabras_clave_esc98tan0fra7_20101116_16-41-45
        Recognized sentence: Hello [0,07527696]
        Alternates:
                [0] Hello [0,07527696]
                [1] Hello [0,07527696]
                [2] Hello [0,07527696]
# DATE: 27/11/2010      TIME: 13:38:17.5352388
File:  CorpusUtterances\palabras_clave_esc98tan0fra7_20101107_22-20-07
        Recognized sentence:  [0]
        Alternates:
                [0]  [0]
# DATE: 27/11/2010      TIME: 13:38:18.0188397
File:  CorpusUtterances\palabras_clave_esc98tan0fra7_20101107_22-20-07
        Recognized sentence:  [0]
        Alternates:
                [0]  [0]
# DATE: 27/11/2010      TIME: 13:38:24.2744507
File:  CorpusUtterances\palabras_clave_esc98tan0fra1_20101114_01-35-01
        Recognized sentence: the washing [0,06160799]
        Alternates:
                [0] the washing [0,06160799]
                [1] the [0,008517146]
```

*vi) EvaluacionTotal.txt*

Dónde se usa: en el método que calcula las estadísticas para el gestor del corpus.

Motivo: Calcula todas las medidas a partir de los datos recopilados durante el análisis de todos los ficheros del corpus.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 18:21:52.0484603
        PATH: 99


****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
```

```
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********************************************************************
        Analysed Sentences: 750
        Recognition Errors: 15

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 78,13 %
        Correctly Recognised: 586

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 82,53 %
        Correctly Understood: 619

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate : 4,4 %
        Not recognized sentences but understood: 33

[WORD ACCURACY (WA)]
        Word Accuracy: 93,25 %

                WA      WS      WD      WI      WT
+(total):       68369,98        162     106     8       4087
AVG(sentence):91,16     0,216   0,141   0,01    5,449
%(sentence):    0,13    0,13    0,13    0,12    0,13
AVG(word):      16,73   0,039   0,025   0,001   1
%(word):        0,02    0,02    0,02    0,01    0,02

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 90,16 %

                KWA     KWS     KWD     KWI     KWT
+(total):       66266,63        25      192     0       2205
AVG(sentence):88,36     0,033   0,256   0       2,94
%(sentence):    0,13    0,13    0,13    1,134088E-38    0,13
AVG(word):      30,05   0,011   0,087   0       1
%(word):        0,05    0,04    0,05    1,398687E-38    0,05

**********************************************************************
```

### 4.4.2 Simulador de Usuarios

*i) ListaObjetivos.txt*

Dónde se usa: en los métodos del simulador de usuario para crear crearEscenariosDeObjetivos() y seleccionarObjetivoAleatoriamente().

Motivo:

Este fichero representa los objetivos de los usuarios para interactuar con el sistema de diálogo. Estos objetivos deben ser cuidadosamente diseñados para conseguir una muestra representativa de los posibles objetivos de usuario.

```
<Room=living_room Appliance=light Attribute=state Value=on>
<Room=living_room Appliance=t_v Attribute=state Value=on>
<Room=living_room Appliance=piped_music Attribute=state Value=on>
<Room=living_room Appliance=washing_machine Attribute=state Value=on>
<Room=kitchen Appliance=light Attribute=state Value=on>
<Room=kitchen Appliance=t_v Attribute=state Value=on>
<Room=kitchen Appliance=piped_music Attribute=state Value=on>
<Room=kitchen Appliance=washing_machine Attribute=state Value=on>
<Room=laundry_room Appliance=light Attribute=state Value=on>
<Room=laundry_room Appliance=t_v Attribute=state Value=on>
<Room=laundry_room Appliance=piped_music Attribute=state Value=on>
<Room=laundry_room Appliance=washing_machine Attribute=state Value=on>
<Room=bathroom Appliance=light Attribute=state Value=on>
<Room=bathroom Appliance=t_v Attribute=state Value=on>
<Room=bathroom Appliance=piped_music Attribute=state Value=on>
<Room=bathroom Appliance=washing_machine Attribute=state Value=on>
<Room=bedroom Appliance=light Attribute=state Value=on>
<Room=bedroom Appliance=t_v Attribute=state Value=on>
<Room=bedroom Appliance=piped_music Attribute=state Value=on>
<Room=bedroom Appliance=washing_machine Attribute=state Value=on>
<Room=corridor Appliance=light Attribute=state Value=on>
<Room=corridor Appliance=t_v Attribute=state Value=on>
<Room=corridor Appliance=piped_music Attribute=state Value=on>
<Room=corridor Appliance=washing_machine Attribute=state Value=on>
<Room=hall Appliance=light Attribute=state Value=on>
<Room=hall Appliance=t_v Attribute=state Value=on>
<Room=hall Appliance=piped_music Attribute=state Value=on>
<Room=hall Appliance=washing_machine Attribute=state Value=on>
<Room=garage Appliance=light Attribute=state Value=on>
<Room=garage Appliance=t_v Attribute=state Value=on>
<Room=garage Appliance=piped_music Attribute=state Value=on>
<Room=garage Appliance=washing_machine Attribute=state Value=on>
```

### *ii) EvaluacionTotal-UserSimulator.txt*

Dónde se usa: en el método que calcula las estadísticas para el gestor del corpus.

Motivo: Calcula todas las medidas a partir de los datos recopilados durante la generación aleatoria de diálogos usando los ficheros del corpus.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 20:12:42.3518359
        PATH: EstadisticasSimuladorTotal.txt
```

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********

Analysed Sentences: 469
Recognition Errors: 64

[SENTENCE RECOGNITION (SR)]
Sentence Recognition: 27,08 %
Correctly Recognised: 127

[SENTENCE UNDERSTANDING RATE (SU)]
Sentence Understanding Rate: 17,7 %
Correctly Understood: 83

[IMPLICIT RECOVERY (IR)]
Implicit Recovery Rate: 1,28 %
Not recognized sentences but understood: 6

[TASK COMPLETION (TC)]
Total of dialogues: 100
Finished dialogues: 30
Task completion: 30 %

[WORD ACCURACY (WA)]
Word Accuracy: 27,59 %

| | WA | WS | WD | WI | WT |
|---|---|---|---|---|---|
| +(total): | 11175 | 270 | 32 | 871 | 1620 |
| AVG(sentence): | 23,83 | 0,575 | 0,068 | 1,857 | 3,454 |
| %(sentence): | 0,21 | 0,21 | 0,21 | 0,21 | 0,21 |
| AVG(word): | 6,9 | 0,166 | 0,019 | 0,537 | 1 |
| %(word): | 0,06 | 0,06 | 0,06 | 0,06 | 0,06 |

[KEYWORD ACCURACY (KWA)]
Keyword Accuracy: 27,59 %

| | KWA | KWS | KWD | KWI | KWT |
|---|---|---|---|---|---|
| +(total): | 11175 | 270 | 32 | 871 | 1620 |
| AVG(sentence): | 23,83 | 0,575 | 0,068 | 1,857 | 3,454 |
| %(sentence): | 0,21 | 0,21 | 0,21 | 0,21 | 0,21 |
| AVG(word): | 6,9 | 0,166 | 0,019 | 0,537 | 1 |
| %(word): | 0,06 | 0,06 | 0,06 | 0,06 | 0,06 |

**********

### *iii) Dialogues.txt*

Dónde se usa: en el método que simula al usuario.

Motivo: generar un fichero que muestre los diálogos generados aleatoriamente, con los objetivos al inicio.

```
#0
Goals:
        0.- room=kitchen : False
        1.- appliance=piped music : False
        2.- attribute=state : False
        3.- value=on : False
-
- User: Hello switch on the music
- Mayordomo: Where?
- User: in the kitchen
- Mayordomo: You have changed to on the state of the piped music in the kitchen.

#1
Goals:
        0.- room=corridor : False
        1.- appliance=piped music : False
        2.- attribute=state : False
        3.- value=on : False
-
- User: Good morning switch on the music
- Mayordomo: Where?
- User: in the hall
- Mayordomo: Sorry, could you repeat room?
```

### *iv) LOG_simulador_usuario.txt*

Dónde se usa: en el método que simula al usuario.

Motivo: generar un fichero que muestre los diálogos generados aleatoriamente, con toda la información utilizada (frases y representaciones semánticas y los objetivos al inicio).

```
[Date: 28/11/2010 ]
[Time: 00:07:00.4889153 ]
[Number of dialogues: 1000]

DIALOGUE #0:
GOAL[0] = 0.- room=corridor : False
GOAL[1] = 1.- appliance=washing machine : False
GOAL[2] = 2.- attribute=state : False
GOAL[3] = 3.- value=on : False
[0]
[FILE: CUSimUser\orden_esc99tan0fra24_20101114_23-25-06.RSCor ]
- User: Good afternoon turn the washing machine
RSRec={x washing_machine x x }
```

COR =Good afternoon start the washing machine
RSCor={x washing_machine state on }
- Mayordomo: Where? Which option? Which value?
[1]
[FILE: CUSimUser\palabras_clave_esc98tan1fra2_20101110_13-39-26.RSCor ]
- User: on
RSRec={x washing_machine state on }
COR =On
RSCor={x x state on }
- Mayordomo: Where?
[2]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101108_20-11-23.RSCor ]
- User: in the hall
RSRec={hall washing_machine state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?
[3]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101116_13-14-42.RSCor ]
- User: in the hall
RSRec={hall washing_machine state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?
[4]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101110_13-19-26.RSCor ]
- User: in the hall
RSRec={hall washing_machine state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?
[5]
[FILE: CUSimUser\palabras_clave_esc98tan0fra1_20101107_19-29-13.RSCor ]
- User: turn the on hall
RSRec={hall washing_machine state on }
COR =In the corridor
RSCor={corridor x x x }
- Mayordomo: Sorry, could you repeat room?
[ERROR: dialogue limit has been exceeded.]

# 5. Experimentos realizados

## 5.1 Perfiles de Reconocimiento de Voz

Con este experimento, se pretende demostrar la importancia que tiene entrenar los modelos acústicos del motor para que el reconocimiento de habla sea más preciso. Para realizar este experimento, se ha usado un corpus de 525 frases.

Se crea un nuevo perfil de reconocimiento de voz, llamado "*Entrenado*" (entrada: micrófono de sobremesa). Este perfil de voz, ha sido creado y entrenado como se explica en el Apéndice B. A continuación, se crea otro perfil de reconocimiento de voz, llamado "*Sin entrenamiento*", pero esta vez, como su nombre indica, no se realiza ningún tipo de entrenamiento sobre él. Tras varios análisis realizados sobre el corpus, obtenemos los siguientes resultados:

| | **PERFIL DE VOZ** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Recognition Errors | Sin entrenamiento | 501 | 274 | 274 | 274 | 274 | 274 |
| | Entrenado | 329 | 21 | 18 | 15 | 10 | 11 |
| Correctly Recognised (SR) | Sin entrenamiento | 1 | 111 | 111 | 111 | 111 | 111 |
| | Entrenado | 71 | 376 | 405 | 406 | 407 | 416 |
| Correctly Understood (SU) | Sin entrenamiento | 1 | 139 | 139 | 139 | 139 | 139 |
| | Entrenado | 93 | 415 | 427 | 429 | 431 | 436 |
| Word Accuracy (%) | Sin entrenamiento | 12,78 | 71,09 | 71,09 | 71,09 | 71,09 | 71,09 |
| | Entrenado | 57,04 | 88,28 | 90,75 | 91,09 | 90,79 | 91,96 |
| Keyword Accuracy (%) | Sin entrenamiento | 37,5 | 76,59 | 76,59 | 76,59 | 76,59 | 76,59 |
| | Entrenado | 69,9 | 91,12 | 92,41 | 92,5 | 92,18 | 92,9 |

Tabla 23: *Datos experimentales obtenidos para dos perfiles de voz*

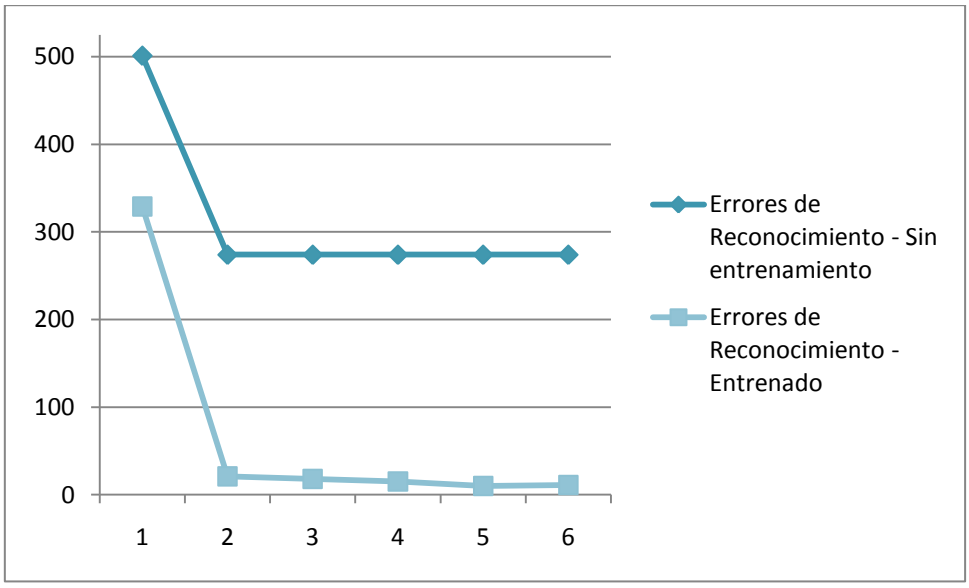



Figura 22: *Errores de reconocimiento*

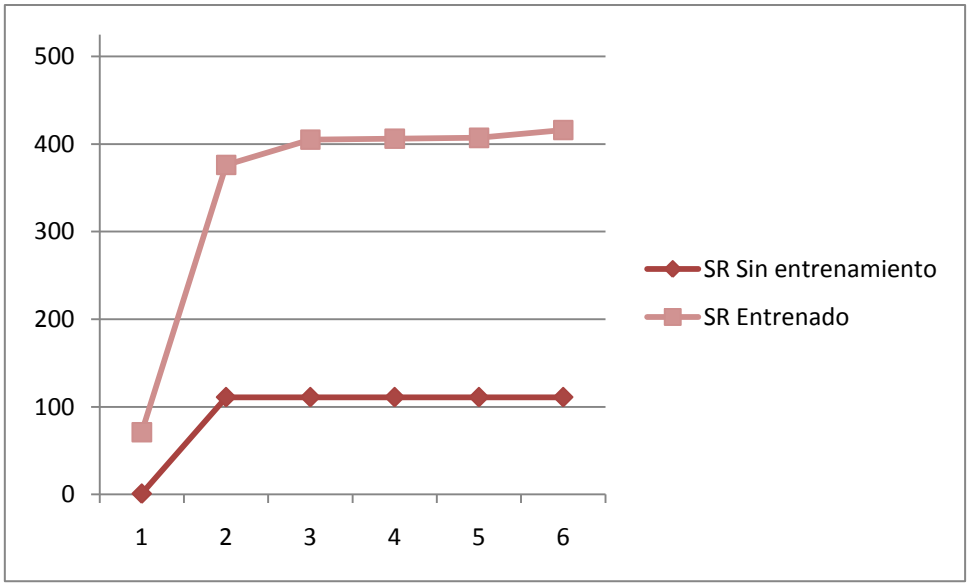



Figura 23: *Reconocimiento de frases (SR)*

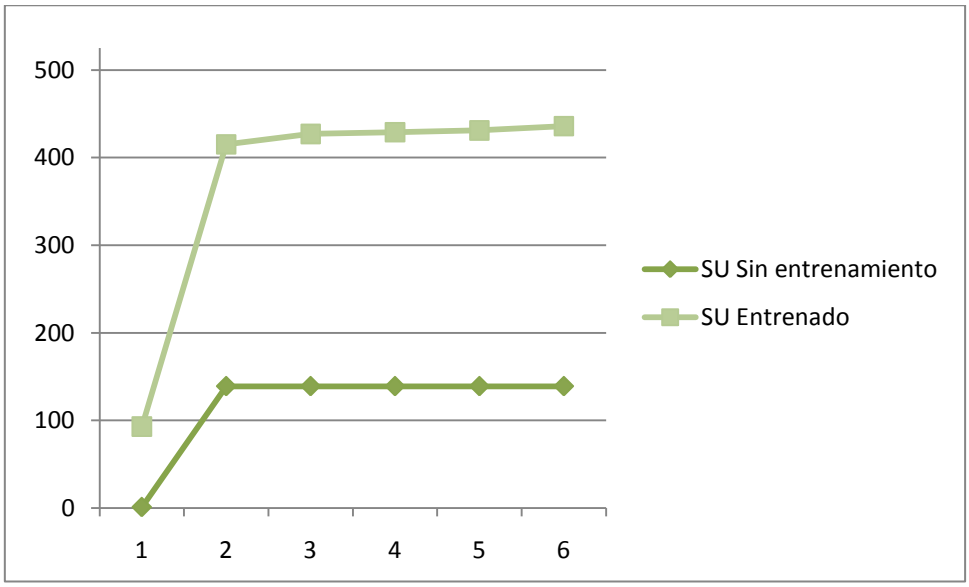



Figura 24: *Comprensión de frases (SU)*

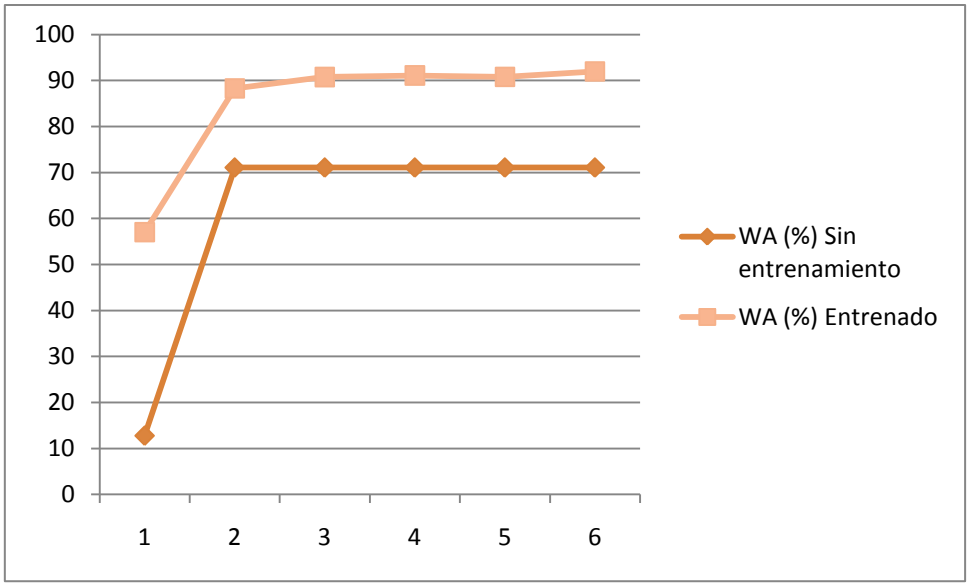



Figura 25: *Exactitud de palabras (WA)*

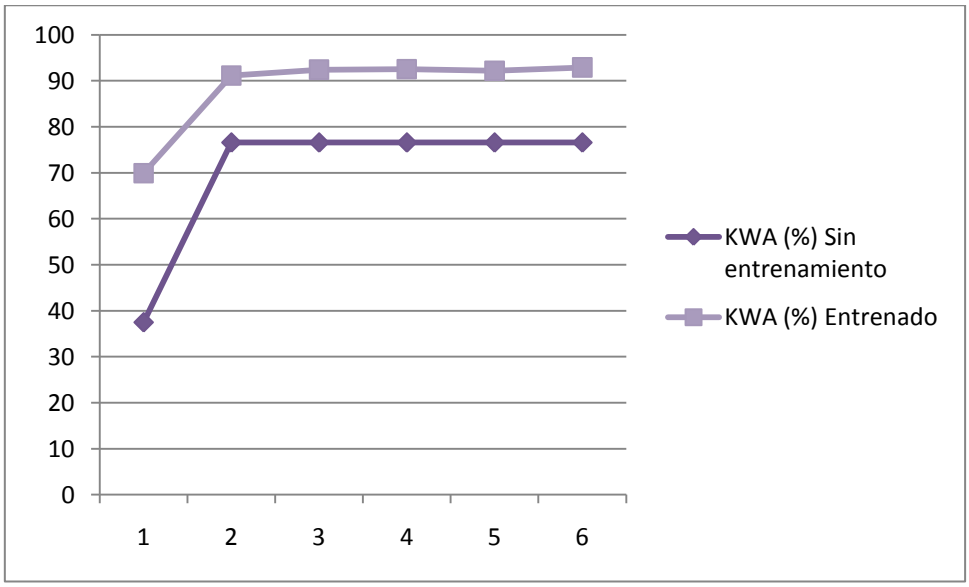



Figura 26: *Exactitud de conceptos (KWA)*

Para el perfil "*Sin entrenamiento*", obtenemos el siguiente fichero de evaluación:

```
        Sentences Analysed: 525
        Recognition Errors: 274

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 21,14 %
        Correctly Recognised: 111

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 26,48 %
        Correctly Understood: 139
```

```
[WORD ACCURACY (WA)]
        Word Accuracy: 71,09 %

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 76,59 %
```

Para el perfil "*Entrenado*", obtenemos el siguiente fichero de evaluación:

```
        Sentences Analysed: 525
        Recognition Errors: 11

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 79,24 %
        Correctly Recognised: 416

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 83,05 %
        Correctly Understood: 436

[WORD ACCURACY (WA)]
        Word Accuracy: 91,96 %

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 92,9 %
```

Como conclusión obtenemos que no entrenar un perfil de voz significa que la cantidad de frases rechazadas por el reconocedor es muy grande (**Recognition Errors**), mayor del 50% del número de frases. Aún así, de las pocas frases aceptadas, la exactitud de palabras (Word Accuracy) no es muy baja pues está en torno a un 70%. Una vez entrenado el perfil, no hay apenas frases rechazadas y la exactitud de palabras es de un 90%.

El entrenamiento del perfil de voz del reconocedor con *Windows Speech Recognition* es totalmente necesario si queremos que nuestro sistema de diálogo responda bien y eficazmente.

## 5.2 Análisis del corpus de frases

En esta sección, vamos a utilizar el gestor de corpus para evaluar cada uno de los escenarios que forman el corpus, primero de manera individual, y a continuación, de manera conjunta.

### 5.2.1 Escenario 1

Como se puede observar, tanto la exactitud de palabras como de conceptos es muy elevada (más de un 90%) para este conjunto de frases grabadas. Las tasas de reconocimiento y comprensión son bastante altas en ambos casos: la primera de un 78% y la segunda de un 82%. Usando las 750 grabaciones de frases del corpus que corresponden a este escenario, sólo se han producido 15 errores de reconocimiento.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 18:21:52.0484603
        PATH: \Escenario1

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********************************************************************
        Analysed Sentences: 750
        Recognition Errors: 15

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 78,13 %
        Correctly Recognised: 586

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 82,53 %
        Correctly Understood: 619

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate : 4,4 %
        Not recognized sentences but understood: 33

[WORD ACCURACY (WA)]
        Word Accuracy: 93,25 %

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 90,16 %
```

### 5.2.2 Escenario 2

Siguiendo el análisis realizado, tanto la exactitud de palabras como de conceptos son bastante bajas (inferiores en ambos casos a un 30%) para este conjunto de frases grabadas. Esto significa que el sistema no es capaz de reconocer correctamente la mayoría de las frases.

Esto se debe a que este escenario está formado por palabras aisladas, y la mayoría de las grabaciones tienen una duración inferior a dos segundos.

Las tasas de reconocimiento y comprensión son bastante bajas también en ambos casos: la primera de un 17% y la segunda de un 18%. De 750 grabaciones de frases, se han producido 292 errores de reconocimiento.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 18:53:55.2744625
        PATH: \Escenario2

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********************************************************************
        Analysed Sentences: 750
        Recognition Errors: 292

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 16,67 %
        Correctly Recognised: 125

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 18 %
        Correctly Understood: 135

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate : 1,33 %
        Not recognized sentences but understood: 10

[WORD ACCURACY (WA)]
        Word Accuracy: 26,81 %

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 28,52 %
```

### 5.2.3 Corpus de frases completo

El resultado del análisis del corpus completo es el siguiente. La exactitud de palabras y de conceptos son bastante altas (81% y 74% respectivamente). Esto significa que el sistema reconoce mejor de lo que entiende, y esto se debe a que este escenario está formado por palabras aisladas cuya exactitud, como pudimos ver en el apartado anterior, es bastante baja.

Las tasas de reconocimiento y comprensión son regulares también en ambos casos: la primera de un 47% y la segunda de un 50%. Considerando todas las grabaciones de frases del corpus, es decir, las 1500 frases, se han producido 305 errores de reconocimiento.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 19:33:34.9005693
        PATH: CorpusUtterances\

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********************************************************************
        Analysed Sentences: 1500
        Recognition Errors: 305

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 47,53 %
        Correctly Recognised: 713

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate : 50,2 %
        Correctly Understood: 753

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate : 2,67 %
        Not recognized sentences but understood: 40

[WORD ACCURACY (WA)]
        Word Accuracy: 81,47 %

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 74 %
```

## 5.3 Simulador de usuarios

El resultado del análisis realizado por el simulador de usuarios es el siguiente. Para 1000 diálogos generados, la exactitud de palabras y de conceptos están en torno a un 25%.

Las tasas de reconocimiento y comprensión son regulares también en ambos casos: la primera de un 28% y la segunda de un 17%. Considerando las 4751 grabaciones de frases utilizadas para generar 1000 diálogos, se han producido 263 errores de reconocimiento.

El número de diálogos completados es 274, lo que significa que, aproximadamente, en tres de cada cuatro casos, el diálogo no termina satisfactoriamente.

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 28/11/2010 - TIME: 00:57:33.2523795
        PATH: EstadisticasSimuladorTotal.txt

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********************************************************************
        Analysed Sentences: 4751
        Recognition Errors: 263

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 28,23 %
        Correctly Recognised: 1341

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate: 17,7 %
        Correctly Understood: 841

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate: 0,91 %
        Not recognized sentences but understood: 43

[TASK COMPLETION (TC)]
        Total of dialogues: 1000
        Finished dialogues: 274
        Task completion: 27,4 %

[WORD ACCURACY (WA)]
        Word Accuracy: 24,86 %

                WA      WS      WD      WI      WT
+(total):       111500  3111    321     10048   17940
AVG(sentence):  23,47   0,654   0,067   2,114   3,776
%(sentence):    0,02    0,02    0,02    0,02    0,02
AVG(word):      6,22    0,173   0,017   0,56    1
%(word):        0,01    0,01    0,01    0,01    0,01

[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 24,86 %

                KWA     KWS     KWD     KWI     KWT
+(total):       111500  3111    321     10048   17940
AVG(sentence):  23,47   0,654   0,067   2,114   3,776
```

| | | | | | |
|---|---|---|---|---|---|
| %(sentence): | 0,02 | 0,02 | 0,02 | 0,02 | 0,02 |
| AVG(word): | 6,22 | 0,173 | 0,017 | 0,56 | 1 |
| %(word): | 0,01 | 0,01 | 0,01 | 0,01 | 0,01 |

En la siguiente tabla, se confirman las mismas conclusiones que con la anterior simulación:

---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
DATE: 27/11/2010 - TIME: 22:47:36.4974306
PATH: EstadisticasSimuladorTotal.txt

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

**********
Analysed Sentences: 4944
Recognition Errors: 389

[SENTENCE RECOGNITION (SR)]
Sentence Recognition: 26,74 %
Correctly Recognised: 1322

[SENTENCE UNDERSTANDING RATE (SU)]
Sentence Understanding Rate: 17,03 %
Correctly Understood: 842

[IMPLICIT RECOVERY (IR)]
Implicit Recovery Rate: 0,91 %
Not recognized sentences but understood: 45

[TASK COMPLETION (TC)]
Total of dialogues: 1000
Finished dialogues: 238
Task completion: 23,8 %

[WORD ACCURACY (WA)]
Word Accuracy: 24,44 %

| | WA | WS | WD | WI | WT |
|---|---|---|---|---|---|
| +(total): | 111150 | 3291 | 319 | 10136 | 18192 |
| AVG(sentence): | 22,48 | 0,665 | 0,064 | 2,05 | 3,679 |
| %(sentence): | 0,02 | 0,02 | 0,02 | 0,02 | 0,02 |
| AVG(word): | 6,11 | 0,18 | 0,017 | 0,557 | 1 |
| %(word): | 0,01 | 0,01 | 0,01 | 0,01 | 0,01 |

```
[KEYWORD ACCURACY (KWA)]
        Keyword Accuracy: 24,44 %

                KWA     KWS     KWD     KWI     KWT
+(total):       111150  3291    319     10136   18192
AVG(sentence):  22,48   0,665   0,064   2,05    3,679
%(sentence):    0,02    0,02    0,02    0,02    0,02
AVG(word):      6,11    0,18    0,017   0,557   1
%(word):        0,01    0,01    0,01    0,01    0,01

************************************************************
```

Para la simulación en la que se generan 500 diálogos obtenemos las mismas conclusiones:

```
---------- CORPUS of Uttered Sentences: Evaluation & Analysis ----------
        DATE: 27/11/2010 - TIME: 23:36:57.4107852
        PATH: EstadisticasSimuladorTotal.txt

****** Recognizer Information: MS-1033-80-DESK
Description: Microsoft Speech Recognizer 8.0 for Windows (English - US)
ID: MS-1033-80-DESK
Culture Information: en-US
Culture Information (ISO 639-2): eng
Culture Information (ISO 639-1): en

************************************************************************
        Analysed Sentences: 2415
        Recognition Errors: 129

[SENTENCE RECOGNITION (SR)]
        Sentence Recognition: 27,33 %
        Correctly Recognised: 660

[SENTENCE UNDERSTANDING RATE (SU)]
        Sentence Understanding Rate: 17,64 %
        Correctly Understood: 426

[IMPLICIT RECOVERY (IR)]
        Implicit Recovery Rate: 1,04 %
        Not recognized sentences but understood: 25

[TASK COMPLETION (TC)]
        Total of dialogues: 500
        Finished dialogues: 132
        Task completion: 26,4 %

[WORD ACCURACY (WA)]
        Word Accuracy: 24,23 %
```

| | WA | WS | WD | WI | WT |
|---|---|---|---|---|---|
| +(total): | 55400 | 1602 | 149 | 5177 | 9144 |
| AVG(sentence): | 22,94 | 0,663 | 0,061 | 2,143 | 3,786 |
| %(sentence): | 0,04 | 0,04 | 0,04 | 0,04 | 0,04 |
| AVG(word): | 6,06 | 0,175 | 0,016 | 0,566 | 1 |
| %(word): | 0,01 | 0,01 | 0,01 | 0,01 | 0,01 |

[KEYWORD ACCURACY (KWA)]

Keyword Accuracy: 24,23 %

| | KWA | KWS | KWD | KWI | KWT |
|---|---|---|---|---|---|
| +(total): | 55400 | 1602 | 149 | 5177 | 9144 |
| AVG(sentence): | 22,94 | 0,663 | 0,061 | 2,143 | 3,786 |
| %(sentence): | 0,04 | 0,04 | 0,04 | 0,04 | 0,04 |
| AVG(word): | 6,06 | 0,175 | 0,016 | 0,566 | 1 |
| %(word): | 0,01 | 0,01 | 0,01 | 0,01 | 0,01 |

**********

# 6. Conclusiones

El estudio realizado en este Trabajo Fin de Máster está relacionado con una línea de investigación muy importante en el campo de los sistemas de diálogo hablado: la evaluación de estos sistemas y sus componentes. Este Trabajo me ha permitido aprender mucho sobre este área de investigación, de la que me resulta realmente interesante poder obtener resultados cuantitativos con los que alcanzar conclusiones sobre qué poder mejorar para conseguir que la interacción de usuarios con el sistema evaluado sea lo más satisfactoria posible.

Me gustaría poder seguir trabajando en el campo de los sistemas de diálogo, especialmente en lo que se refiere a la evaluación, motivo por el cual decidí hacer este Trabajo Fin de Máster. Durante el desarrollo de este Trabajo me han surgido muchos problemas que no se han mencionado en esta Memoria, y que con esfuerzo, conseguí resolver. Este trabajo puede ser la antesala de estudios más ambiciosos que quiero llevar a cabo en este área. Creo que las herramientas desarrolladas en este trabajo pueden ser muy útiles para mi futura investigación.

# 7. Trabajo futuro

Las herramientas implementadas en este Trabajo Fin de Máster para evaluación de sistemas de diálogo hablado están abiertas a una serie de mejoras para completar su funcionalidad y añadir mejoras a la experiencia del usuario. Entre éstas, podemos mencionar las siguientes:

Con respecto al simulador de usuarios, una línea de trabajo futuro consiste en realizar experimentos con las técnicas de elección de las frases que mejor se adaptan a los objetivos a lograr por el simulador de usuarios. Por ahora, sólo buscamos cumplir el máximo número de objetivos posibles y elegir un fichero de los posibles aleatoriamente, pero queremos probar con otras estrategias. Por ejemplo, una estrategia alternativa consiste en intentar logar el menor número de objetivos posibles en cada iteración. La ventaja de esto es que así se pueden cubrir un mayor espectro de posibles interacciones. Con esto, se consigue reflejar en cierta manera el comportamiento impredecible de un usuario real. Otro aspecto a tener en cuenta está relacionado con la elección del fichero que contiene la frase del usuario. En lugar de elegido aleatoriamente, se puede llevar la cuenta de las veces que ha sido utilizado cada fichero en todas las simulaciones para escoger el que menos veces haya salido elegido. De esta manera se usaría más equitativamente el corpus de frases.

Con respecto al corpus, es necesario seguir recopilando muchas más frases para continuar con la evaluación. Asimismo, es necesario diseñar más conjuntos de frases o escenarios que den cabida a generar diálogos con objetivos diferentes, como apagar la televisión o incluso, preguntar qué luces están encendidas en el hogar. Para esto necesitaremos más locutores voluntarios que deseen ayudar a la ampliación de nuestro corpus.

Con respecto a los datos obtenidos en la evaluación, tenemos un amplio campo que explotar. Se nos ocurren muchos experimentos que hacer con los datos objetivos acerca del funcionamiento del sistema de diálogo. Uno de ellos es probar su comportamiento con otros motores de reconocimiento distintos de Windows Speech Recognition, como HTK o Sphinx, comprobar qué diferencia hay entre las frases reconocidas, y analizar qué reconocedor se puede adaptar mejor a nuestro sistema de diálogo.

# 8. Apéndices

## A) Glosario de términos

| Término | *Significado* |
|---|---|
| **API** | *Application Programming Interface,* Interfaz de Programación de Aplicaciones |
| **ASR** | *Automatic Speech Recognition*, Reconocedor Automático de Habla. |
| **C&C** | *Command And Control,* Comando y Control |
| **CFG** | *Context-Free Grammar,* Gramáticas Libres de Contexto |
| **IR** | *Implicit Recovery,* Recuperación Implícita |
| **JSAPI** | *Java Speech Application Programming Interface ,* Interfaz de Programación de Aplicaciones de Habla de Java |
| **KWA** | *KeyWord Accuracy¸* exactitud de conceptos |
| **MSDN** | *Microsoft Developers Network*, Red de Desarrolladores de Microsoft |
| **PLN** | Procesador de Lenguaje Natural |
| **RAH** | Reconocimiento Automático de Habla |
| **SAPI** | *Speech Application Programming Interface*, Interfaz de Programación de Aplicaciones de Habla |
| **SDK** | *Software Development Kit,* Kit de Desarrollo de Software |
| **SR** | *Sentence Recognition,* tasa de reconocimiento de frases |
| **SRAPI** | *Speech Recognition Speech Application Programming Interface,* Interfaz de Programación de Aplicaciones del Reconocimiento de Habla |
| **SRGS** | *Speech Recognition Grammar Specification,* Especificación de Gramáticas para el Reconocimiento del Habla. |
| **SU** | *Sentence Understanding*, tasa de comprensión de frases |
| **TC** | *Task Completion¸* completitud de tareas |
| **TTS** | *Text to Speech*, síntesis de habla |
| **VXML** | *Voice eXtensible Markup Lenguage*, Lenguaje de Marcas Extensible para Voz |
| **WA** | *Word Accuracy,* exactitud de palabras |
| **W3C** | *World Wide Web Consortium,* Consorcio World Wide Web |
| **WPF** | *Windows Presentation Foundation* |
| **WSR** | *Windows Speech Recognition,* Reconocedor de Habla de Windows |
| **XML** | *eXtensible Markup Language* |

# B) Windows Speech Recognition (Windows 7)

## 1) Configurar el reconocimiento de habla

Para configurar el equipo para el Reconocimiento de voz de Windows o Windows Speech Recognition debe realizar tres tareas: configurar el micrófono, aprender a hablar al equipo y entrenarlo para que comprenda su voz.

- *Configurar el micrófono:*

1. Para abrir Opciones de reconocimiento de voz, haga clic en el botón Inicio, en Panel de control, en Facilidad de acceso y, a continuación, en Opciones de reconocimiento de voz.
2. Haga clic en Configurar micrófono.
3. Siga las instrucciones del asistente.

La calidad del reconocimiento de voz está directamente relacionada con la calidad del micrófono que use. Los dos tipos de micrófono más comunes para el reconocimiento de voz son los auriculares con micrófono de diadema y los micrófonos de escritorio. Los auriculares con micrófono son apropiados para trabajar con el reconocimiento de voz porque recogen menos sonidos superfluos.

- *Aprender a hablar al equipo:*

Windows incluye un tutorial de aprendizaje que le enseñará los comandos que debe usar para trabajar con el reconocimiento de voz. Tardará unos 30 minutos en completar el tutorial, por lo que antes de iniciarlo debe asegurarse de tener suficiente tiempo libre para hacerlo. Siga estos pasos para ejecutar el tutorial de aprendizaje para el reconocimiento de voz:

1. Para abrir Opciones de reconocimiento de voz, haga clic en el botón Inicio, en Panel de control, en Facilidad de acceso y, a continuación, en Opciones de reconocimiento de voz.
2. Haga clic en Seguir tutoriales de voz.
3. Siga las instrucciones del tutorial re reconocimiento de voz.

- *Entrenar a su PC para que reconozca su voz:*

1. Para abrir Opciones de reconocimiento de voz, haga clic en el botón Inicio, en Panel de control, en Facilidad de acceso y, a continuación, en Opciones de reconocimiento de voz.
2. Haga clic en Entrenar el equipo para que me comprenda mejor.

3. Siga las instrucciones del asistente.

Recuerde que el software de reconocimiento viene desactivado por defecto. Una vez activado desde el *Panel de Control,* Windows Speech Recognition puede estar en uno de los tres estados siguientes:

- Activo y escuchando.
- Activo y esperando el comando "*Start listening*".
- Desactivado.

Se pueden cambiar las opciones de habla del software de reconocimiento haciendo doble clic con el botón derecho sobre el icono del micrófono que aparece en la bandeja del sistema.

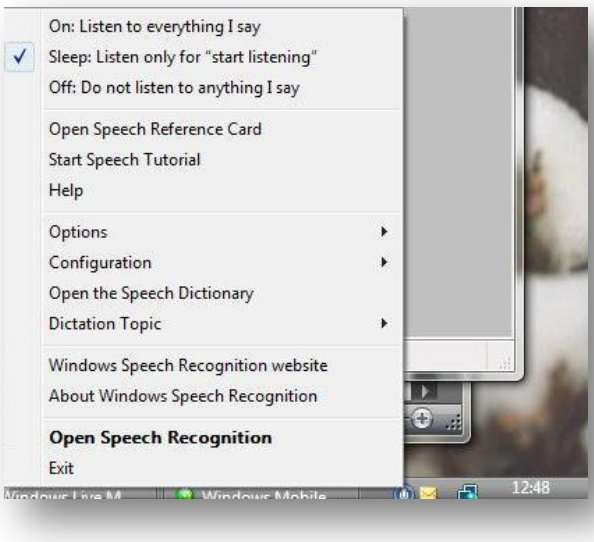



**Figura 27:** ***Opciones de estados del reconocedor***

Las opciones que aparecen en el menú son las siguientes:

- On: El reconocedor está a la escucha de todo lo que se dice, ejecutando todos los comandos que reconozca.
- Sleep: El reconocedor escucha pero no responderá a la voz hasta que se diga "*Start listening.*"
- Off: El reconocedor no escuchará nada de lo que se diga.

## 2) Configurar el idioma con Windows Speech Recognition

A continuación, se explica cómo configurar el idioma del reconocedor a inglés americano (para la versión **ULTIMATE** de cualquier sistema operativo Windows, en este caso, Windows 7 Ultimate).

1) Previamente, instalar los paquetes de idiomas deseados (en nuestro caso el inglés), desde Windows Update.
2) Tras esto, ir a: Panel de Control > Accesibilidad > Reconocimiento de Voz.

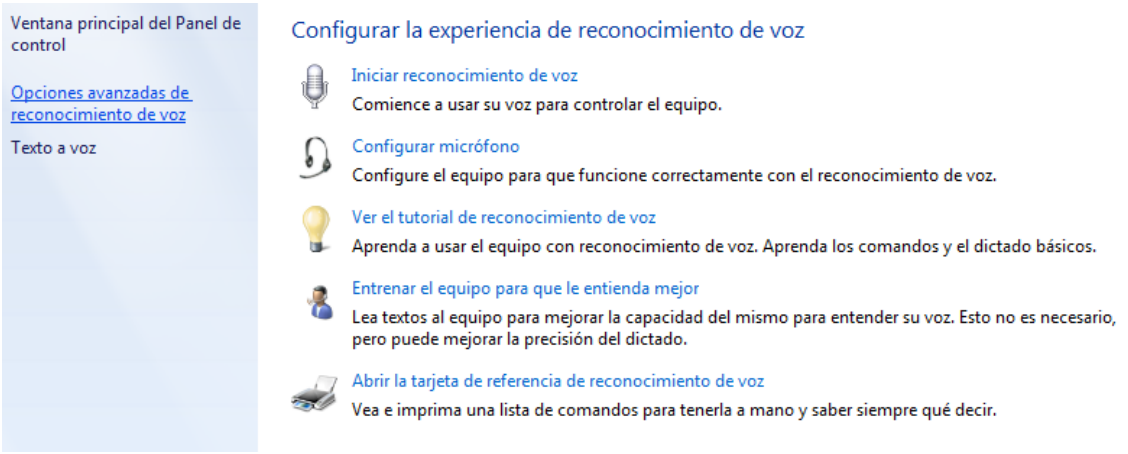
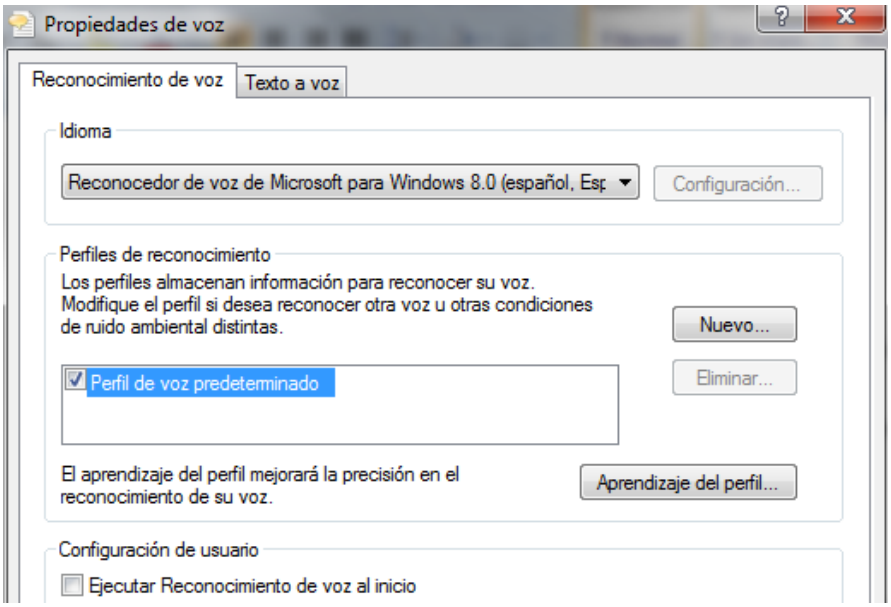



Figura 28: *Reconocimiento de voz en el Panel de Control*

3) En el menú de la izquierda, pulsar sobre "Opciones avanzadas de reconocimiento de voz". Aparece la siguiente ventana con dos pestañas:



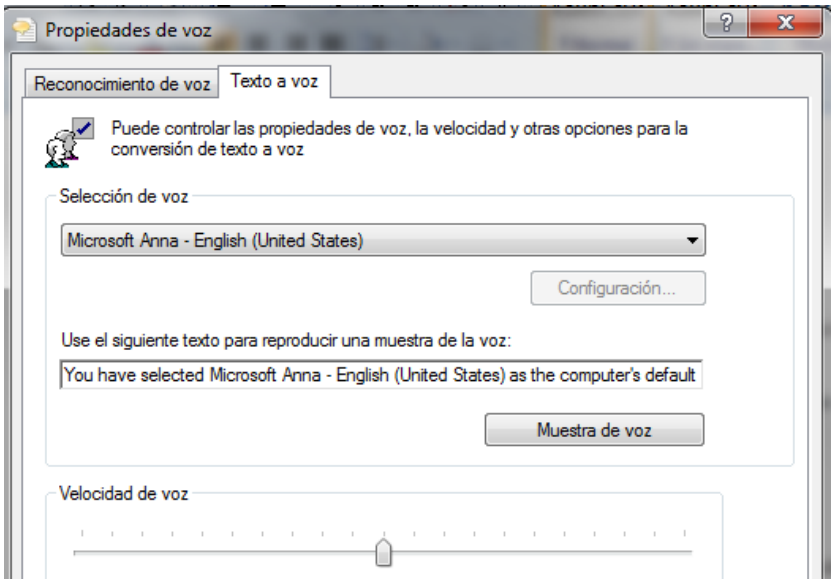



Figura 29: *Propiedades de voz en WSR*

4) Seleccionar en Reconocimiento de voz, el correspondiente al inglés de Estados Unidos.
5) Cambiar el idioma de Windows a inglés. Aparece el siguiente mensaje de advertencia, pulse Aceptar:

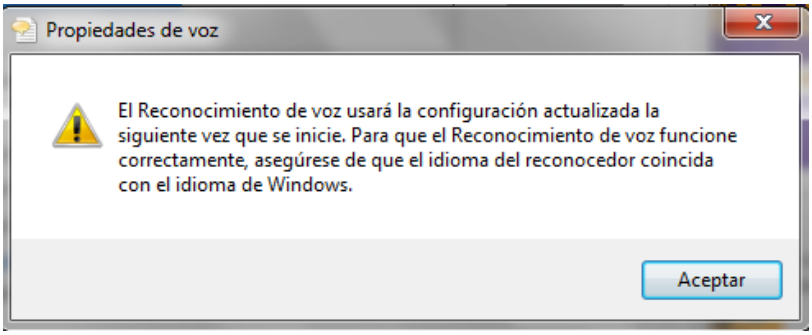



Figura 30: *Mensaje de advertencia*

Ir a Panel de Control > Reloj, Idioma y Región > Cambiar el idioma para mostrar:

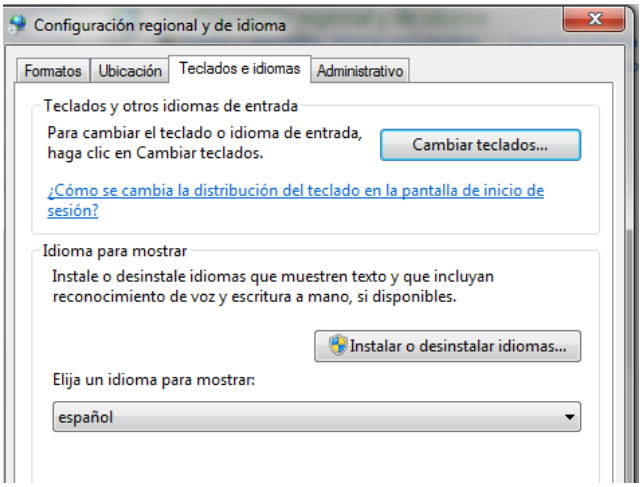



Figura 31: *Configuración de idioma*

Pulse Aceptar y guarde todo ya que se cerrará sesión.

6) Reinicie y el nuevo idioma está listo para usar.

## 3) Crear un perfil de voz

Es recomendable definir un perfil de voz en el primer momento en que se va a usar Windows Speech Recognition. Como vimos en la sección 3.1.1., un perfil almacena mucha información del entorno que influye en el reconocimiento.

Desde el Panel de Control -> Accesibilidad -> Reconocimiento de voz -> Opciones avanzadas de reconocimiento de voz. En la pestaña de Reconocimiento de Voz, en el recuadro de Perfiles de reconocimiento, pulse el botón Nuevo.

El primer paso es darle un nombre al nuevo perfil. Es recomendable poner un nombre que incluya el nombre de la persona que va a usar ese perfil y del entorno (oficina, hogar) o el tipo de micrófono que va a usar.

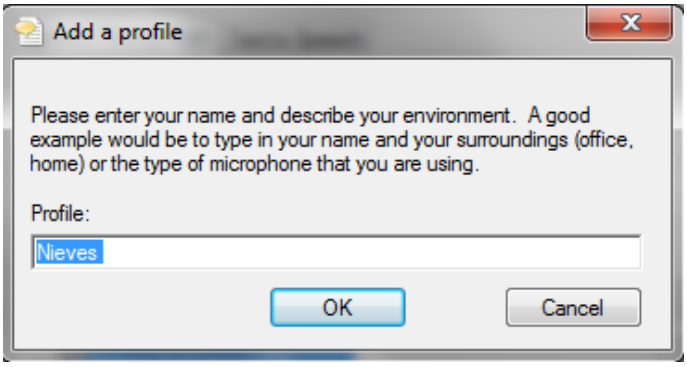



**Figura 32:** ***Añadir un perfil de voz***

El siguiente paso es elegir el tipo de micrófono que se usará:

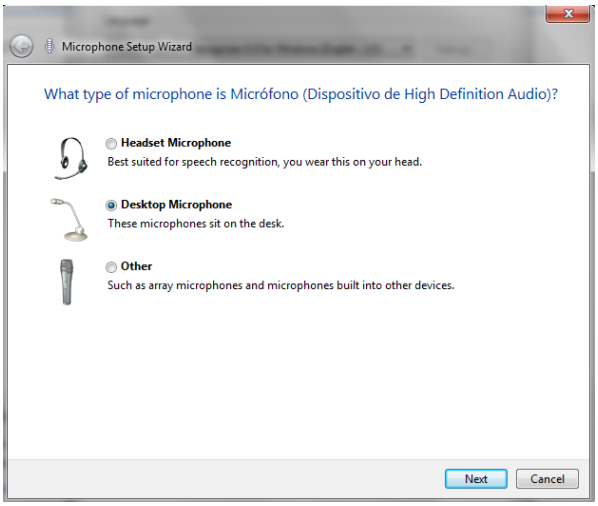



**Figura 33:** ***Elección del tipo de micrófono***

Tras crear un perfil de usuario, el siguiente paso necesario es configurar y entrenar dicho perfil. Este entrenamiento es requisito indispensable para que el reconocedor se 'adapte' a la voz del usuario y así reducir la cantidad de fallos de reconocimiento a posteriori.

Desde el Panel de Control -> Accesibilidad -> Reconocimiento de voz ->Entrenar el equipo para que le entienda mejor.

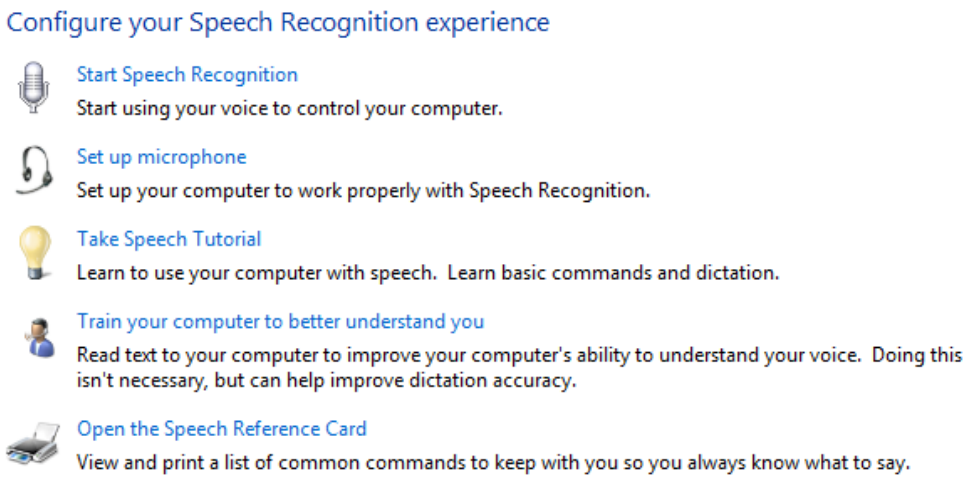



**Figura 34:** ***Acciones disponibles en WSR***

Se puede acceder también al entrenamiento del perfil de voz desde la pestaña donde creamos el perfil de usuario. Desde ahí, hay que seleccionar el perfil de voz y a continuación pulsar el botón Train Profile:

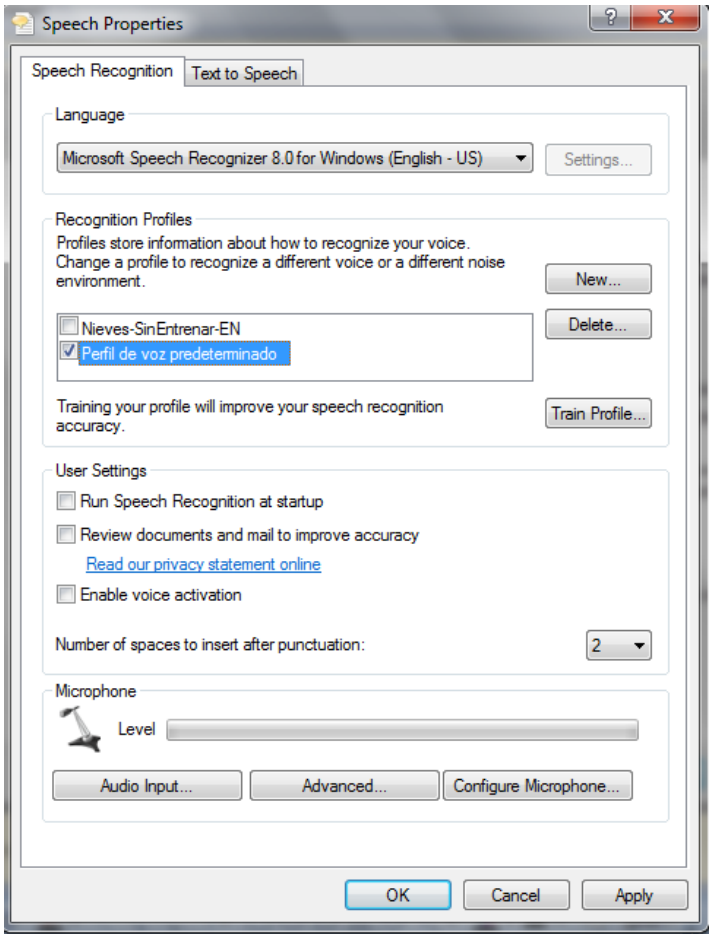



**Figura 35:** ***Propiedades del habla***

Antes de iniciar el entrenamiento se preguntará al usuario si quiere compartir los datos de su perfil con Microsoft.

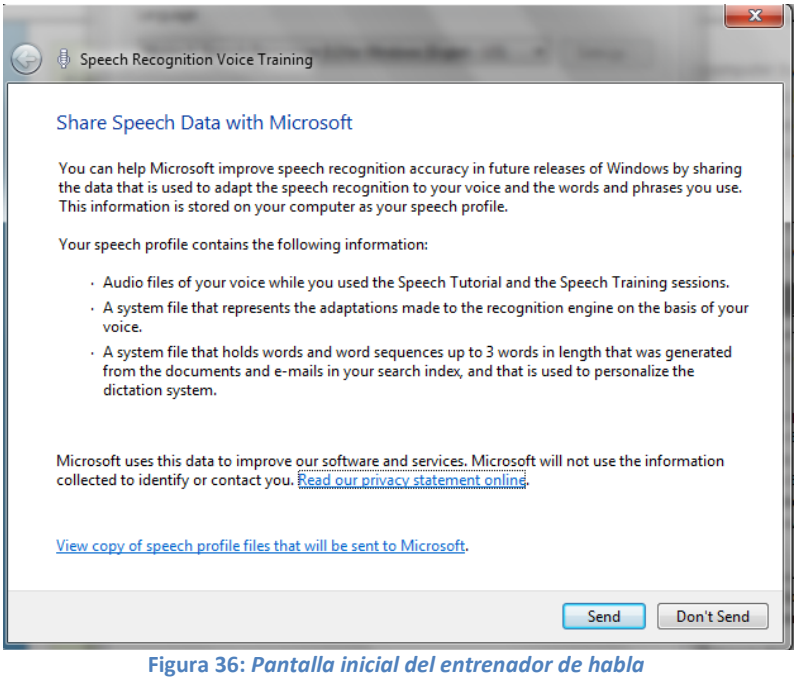



**Figura 36:** ***Pantalla inicial del entrenador de habla***

Como primer paso del entrenamiento, se explica el procedimiento. Se hará leer en voz alta al usuario una serie de frases de tal manera que el reconocedor aprenda como habla.

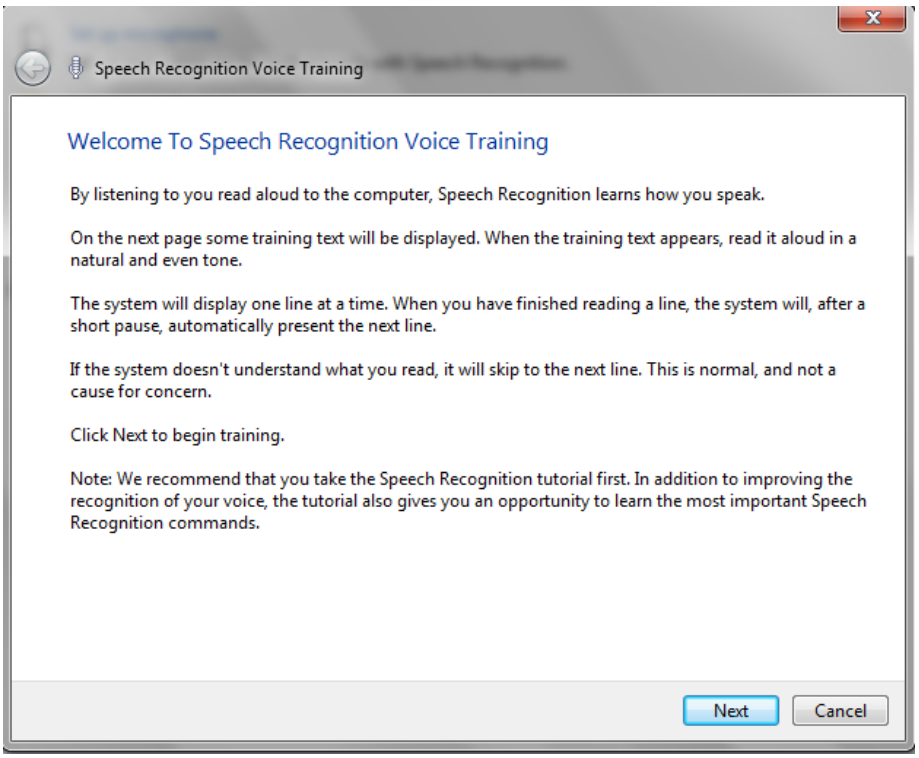



**Figura 37:** ***Información del procedimiento***

Una vez leídas las frases que aparecen en la pantalla, se puede dejar el entrenamiento o se puede seguir para obtener mejores resultados de comprensión del habla del usuario.

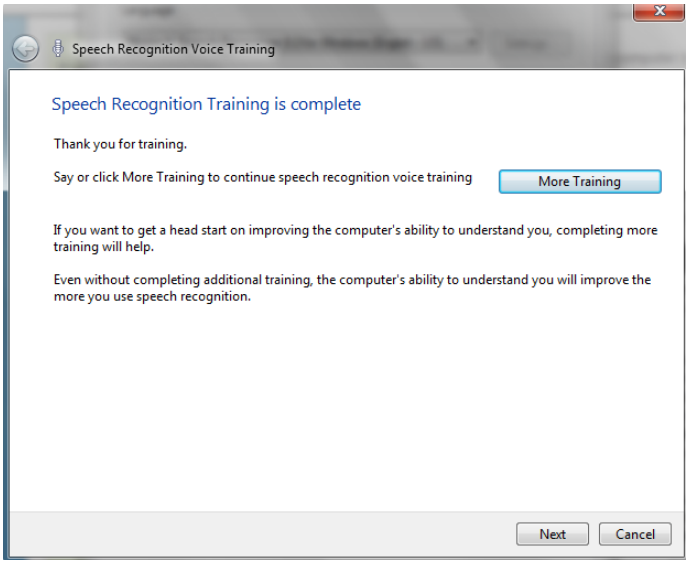



**Figura 38:** ***Fin del entrenamiento del habla***

# 10. Bibliografía

*R. López-Cózar, Masahiro Araki.* **Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment,** John Wiley & Sons, UK, 2005.

*R. López-Cózar, A. de la Torre, J. C. Segura, A. J. Rubio.* **Assessment of Dialogue Systems by Means of a New Simulation Technique**. *Speech Communication*, 40 (3), pág. 387-407, 2003

*J. Llisterri, M. J. Machuca.* "Introducción a los sistemas de diálogo" **Los Sistemas de Diálogo**. 2006.

*S. Möller*. **Assessment and Evaluation of Speech-Based Interactive Systems: From Manual Annotation to Automatic Usability Evaluation** - Speech Technology, 2010 - Springer (2010)

*M. Boros, W. Eckert, F. Gallwitz, G. Gorz.* **Towards understanding spontaneous speech: Word accuracy vs. concept accuracy.** International Conference on Spoken Language, 1996. ICSLP 96. Proceedings.(1996)

*N. Ábalos, G. Espejo, R. López-Cózar, Z. Callejas*. **Sistema de Diálogo Multimodal para una Aplicación de Inteligencia Ambiental en una Vivienda.** Revista para el Procesamiento del Lenguaje Natural, num. 44. (2010)

*N. Ábalos, G. Espejo, R. López-Cózar, Z. Callejas, D.Griol*. **A Multimodal Dialogue System for an Ambient Intelligent Application in Home Environments.** TSD 2010, LNAI 6231, pp. 491–498, 2010. Springer-Verlag Berlin Heidelberg (2010)

*G. Espejo, N. Ábalos, R. López-Cózar, Z. Callejas, D. Griol*. **Mayordomo: Ambient Intelligence in a Home Environment**. Proceedings of HCIAmI (2010)

*G. Espejo, N. Ábalos, R. López-Cózar, Z. Callejas, D. Griol*. **Sistema de diálogo Mayordomo: Una aplicación multimodal de Inteligencia Ambiental para el hogar**. Actas de Interacción 2010.

**Documentación para Microsoft Speech API (5.3):**

http://msdn.microsoft.com/en-us/library/ms723627(VS.85).aspx

Documentación para **System.Speech**:

- Recognition: http://msdn.microsoft.com/es-es/library/system.speech.recognition.aspx
- SRGS: http://msdn.microsoft.com/es-es/library/system.speech.recognition.srgsgrammar.aspx
- Synthesis: http://msdn.microsoft.com/es-es/library/system.speech.synthesis.aspx
- Versión actualizada para .NET Framework 4: http://msdn.microsoft.com/en-us/library/gg145021.aspx

Software de reconocimiento de voz para Windows, **Windows Vista Speech Recognition**:

http://msdn.microsoft.com/en-us/magazine/cc163663.aspx

http://articles.techrepublic.com.com/5100-10878_11-6165615.html

Gramáticas:

- XML: http://www.w3.org/TR/REC-xml
- SRGS: http://www.w3.org/TR/2004/REC-speech-grammar-20040316/

# 11. Índices de tablas y figuras

## 11.1 Índice de tablas

Tabla 1: *Espacios de nombres incluidos en System.Speech* ..... 18
Tabla 2: *Clases de System.Speech.Recognition* ..... 21
Tabla 3: *Enumeraciones de System.Speech.Recognition* ..... 21
Tabla 4: *Clases de System.Speech.Speech* ..... 22
Tabla 5: *Enumeraciones de System.Speech.Recognition* ..... 23
Tabla 6: *Descripción de los campos del concepto acción* ..... 28
Tabla 7: *Preguntas realizadas por el gestor de diálogo* ..... 29
Tabla 8: *Diseño para la luz.* ..... 33
Tabla 9: *Diseño para la lavadora.* ..... 34
Tabla 10: *Diseño para el hilo musical.* ..... 35
Tabla 11: *Diseño para la televisión.* ..... 36
Tabla 12: *Estructura Atributo* ..... 37
Tabla 13: *Clase Electrodoméstico* ..... 38
Tabla 14: *Clase Habitación* ..... 38
Tabla 15: *Clase Hogar* ..... 39
Tabla 16: *Tabla de correspondencias* ..... 48
Tabla 17: *Estructura wordAccuracy* ..... 60
Tabla 18: *Clase GestorHogar* ..... 62
Tabla 19: *Clase Semantica* ..... 63
Tabla 20: *Clase Objetivo* ..... 65
Tabla 21: *Clase SimuladorUsuario* ..... 66
Tabla 22: *Clase GestorCorpus* ..... 72
Tabla 23: *Datos experimentales obtenidos para dos perfiles de voz* ..... 87

# 11.2 Índice de figuras

Figura 1: *Propiedades del conversor de texto a habla: selección de la voz y velocidad.* 12
Figura 2: *Propiedades del reconocedor del habla.* ..... 13
Figura 3: *Voces instaladas* ..... 14
Figura 4: *Reconocedores de habla instalados* ..... 14
Figura 5: *Ejemplo de perfiles de reconocimiento* ..... 14
Figura 6: *Esquema de la interacción mediante el habla en* Mayordomo ..... 26
Figura 7: *Regla inicial de las gramáticas* ..... 27
Figura 8*: Esquema del proceso seguido por el gestor del diálogo* ..... 30
Figura 9*: Interfaz gráfica de* Mayordomo ..... 32
Figura 10: *Obtención del corpus de grabaciones de frases* ..... 47
Figura 11: *Significado del nombre de un fichero del corpus de frases* ..... 48
Figura 12: *Transcripción Automática Ortográfica de un fichero del corpus de frases* ... 48
Figura 13: *Transcripción Automática Semántica de un fichero del corpus de frases* ..... 49
Figura 14: *Obtención de representaciones semánticas* ..... 50
Figura 15: *Cálculo de la exactitud de palabras* ..... 50
Figura 16: *Cálculo de la exactitud de conceptos* ..... 51
Figura 17: *Información obtenida en el reconocimiento* ..... 51
Figura 18: *Medidas estadísticas obtenidas por el gestor del corpus* ..... 52
Figura 19: *Estrategia de selección de frases en la simulación de usuarios* ..... 53
Figura 20: *Ejemplo turno de diálogo generado por el simulador de usuarios* ..... 54
Figura 21: *Diagrama de clases* ..... 59
Figura 22: *Errores de reconocimiento* ..... 87
Figura 23: *Reconocimiento de frases (SR)* ..... 88
Figura 24: *Comprensión de frases (SU)* ..... 88
Figura 25: *Exactitud de palabras (WA)* ..... 89
Figura 26: *Exactitud de conceptos (KWA)* ..... 89
Figura 27: *Opciones de estados del reconocedor* ..... 105
Figura 28: *Reconocimiento de voz en el Panel de Control* ..... 106
Figura 29: *Propiedades de voz en WSR* ..... 107
Figura 30: *Mensaje de advertencia* ..... 107
Figura 31: *Configuración de idioma* ..... 107
Figura 32: *Añadir un perfil de voz* ..... 108
Figura 33: *Elección del tipo de micrófono* ..... 108
Figura 34: *Acciones disponibles en WSR* ..... 109
Figura 35: *Propiedades del habla* ..... 110
Figura 36: *Pantalla inicial del entrenador de habla* ..... 110
Figura 37: *Información del procedimiento* ..... 111
Figura 38: *Fin del entrenamiento del habla* ..... 111