Diseño e implementación de un sistema telefónico interactivo que permita automatizar la toma de pedidos en un restaurante de comida rápida utilizando reconocimiento de voz

Eloísa Orozco[1], Alvaro Padilla[2], Gabriel Astudillo, Ing.[3]
[1-3] Facultad de Ingeniería en Electricidad y Computación – Escuela Superior Politécnica del Litoral (ESPOL), Campus "Gustavo Galindo V.", Km. 30.5 Vía Perimetral, Apartado 09-01-5863. Guayaquil, Ecuador
[1] Email: eorozco@fiec.espol.edu.ec
[2] Email: apadilla@fiec.espol.edu.ec
[3] Email: gastudil@fiec.espol.edu.ec

Resumen

El objetivo del presente proyecto es diseñar e implementar un sistema de respuesta de voz interactiva (IVR) que automatice la toma de pedidos en un restaurante de comida rápida utilizando reconocimiento de voz.

El desarrollo de este sistema consta de cuatro fases principales, siendo la inicial diseñar un modelo que se acople a la lógica de negocio de un restaurante de comida rápida y al proceso que se va a automatizar. La siguiente fase es implementar un asistente automático de llamadas con respuesta de voz (IVR) que presente menús y submenús por medio del ingreso y extracción limitados de datos de la base del restaurante. Luego hay que integrar reconocimiento de voz a nuestro sistema para permitir al usuario seleccionar los menús y opciones con comandos de voz. Finalmente se implementa una aplicación interna para uso de los empleados del restaurante que muestre un reporte de estado de los pedidos pendientes y permita modificar el estado de los mismos.

Implementando este IVR en un ambiente comercial, el restaurante estaría innovando con el uso de la tecnología de reconocimiento de voz que no es popular en nuestro medio.

Palabras Claves: *IVR, ASR, Reconocimiento automático de voz, STT, TTS.*

Abstract

The objective of this project is to design and implement an interactive voice response (IVR) system to automate the processing of orders in a fast food restaurant using voice recognition.

The development of this system consists of four main phases, the initial one being to design a model that fits the business logic of a fast food restaurant and specifically to the process to be automated. The next phase is to implement a telephonic auto-attendant with voice response (IVR) that presents menus and submenus through the income and limited extraction of data from the restaurant database. Then speech recognition is integrated into our system to allow users to select menus and options with voice commands. Finally an application is deployed for use by internal employees of the restaurant. It displays a status report of pending orders and allows users to modify the status of them.

Implementing the IVR in a commercial environment, the restaurant would innovate with the use of voice recognition technology that is not popular in our country yet.

Keywords: *IVR, ASR, Automatic Speech Recognition, STT, TTS.*

1. Introducción

En nuestro medio, la toma de pedidos en restaurantes de comida rápida, en especial de pizzerías, se realiza comúnmente de manera presencial, a través de un portal web o por vía telefónica. En el caso de la llamada telefónica, el empleado del restaurante debe pedir los datos completos de la persona que realiza la llamada si es un cliente nuevo o buscarlo en una aplicación si éste ya se encuentra registrado. Después de que el pedido sea definido y se haya especificado la forma de pago, éste debe ser despachado para la entrega.

Este sistema busca automatizar el proceso previamente descrito brindando al cliente final una experiencia de usuario novedosa, el reconocimiento de voz. Además se debe proveer a los empleados del restaurante una aplicación para revisar varios reportes y para actualizar el estado de los pedidos pendientes.

La solución se implementará unificando el reconocimiento de voz para recibir comandos del usuario que sirvan para el procesamiento de la llamada y el intercambio de datos limitado con la base de datos. Finalmente se busca optimizar recursos tanto al restaurante como a sus clientes.

2. Justificación de la investigación

El servicio en el restaurante consiste en minimizar los tiempos de preparación y entrega, atendiendo siempre al cliente con una sonrisa. En el caso del pedido telefónico para entregas a domicilio, el servicio se basa en satisfacer todos los requerimientos que pueda solicitar un cliente sin que sienta la necesidad de presentarse físicamente en el restaurante para lograr su objetivo. Al automatizar la recepción de llamadas entonces, se debe mantener la amigabilidad que caracteriza al restaurante tratando de personalizar lo más posible la llamada manteniendo a la vez un menú intuitivo y directo.

La aplicación de despacho de pedidos debe tener una interfaz amigable y de uso fácil y rápido para sus usuarios - los empleados del restaurante. Al ser un restaurante de comida rápida se dará el escenario donde pocos empleados deberán encargarse del manejo de varios pedidos en poco tiempo.

3. Marco teórico de reconocimiento de voz

3.1 Sphinx

Sphinx es un conjunto de sistemas de reconocimiento de voz continua, y en la actualidad incluye a Sphinx2, Sphinx3, Sphinx4 y otras herramientas como entrenadores de modelos acústicos [1]. El motor de reconocimiento de voz que se usa en este proyecto es Sphinx2, la versión original de Sphinx escrita en lenguaje C [2].

La versión 3 de Sphinx escrita en C++ y la versión 4, escrita en Java tienen una mejor fidelidad en el reconocimiento de voz, pero Sphinx2 es más rápido [3][4]. Sphinx2 debe funcionar correctamente en Linux, Solaris, varias versiones de Unix, y en Windows usando Visual Studio y cygwin.

La diferencia de Sphinx con respecto a otros SREs es que no requiere aprendizaje o entrenamiento para reconocer palabras sino sólo el diccionario de los términos a reconocer. El uso de RAM varía de acuerdo al tamaño del diccionario utilizado.

Aunque la versión 2 está considerada como obsoleta y de esta hay menos documentación existente en comparación con las siguientes versiones, es la que ha tenido más casos de éxito reportados integrándose con Asterisk [5] y es la versión de la que hemos encontrado más discusión y ejemplos [6] por parte de usuarios en Internet.

Por esta razón utilizaremos Sphinx2 para este proyecto y no sus versiones más actuales. Tener reconocimiento de voz y no poder utilizarlo en la llamada telefónica impediría completamente lograr el objetivo de este proyecto.

3.2 Modelo acústico

Un modelo acústico reconoce los sonidos naturales a un lenguaje específico. Para elaborar un modelo acústico eficiente en un idioma se necesita analizar más de 2000 horas de grabaciones de nativos leyendo diversos textos, aunque basta con 140 horas de grabaciones para elaborar uno de mediana fidelidad.

La elaboración de un modelo acústico es un proceso complejo que toma como mínimo seis meses teniendo un conocimiento avanzado de fonética, linguística, y conceptos afines. Por esta razón son pocos los modelos acústicos libres de buena calidad, y se da el caso de motores de reconocimiento de voz libres que usan por defecto un modelo acústico eficiente pero privativo.

Las versiones más recientes de Sphinx nos proporcionan varios modelos acústicos de calidad. Actualmente existen modelos acústicos en ingles-EE.UU., español, francés y chino los cuales han sido optimizados cuidadosamente para mejorar los resultados en el reconocimiento de voz. Existen modelos acústicos para micrófonos y para el habla en un teléfono.

3.3 Modelo de lenguaje

Este modelo se encarga de estudiar la lógica del lenguaje, 'entendiéndolo' para calcular probabilidades de qué palabra sucede a cuál al decir una oración. El objetivo de este modelo es ahorrar tiempo al detectar palabras teniendo una lista de palabras que probablemente deberían seguirla, por ejemplo, al decir un artículo, se espera que la siguiente palabra sea un sustantivo [7].

3.4 Comprensión del Lenguaje Natural - NLU

La compresión del lenguaje natural busca que un ordenador entienda el lenguaje natural de las personas [8]. Hasta ahora no se ha reportado ningún caso de éxito absoluto logrando entendimiento del lenguaje natural por varios detalles de los lenguajes que dependen del contexto de una oración, por ejemplo, los homófonos.

4. Diseño de la solución

4.1 Esquema general de la solución

Cuando el usuario llama a la extensión inicial del IVR, es redireccionado al script AGI de bienvenida. Desde este script inicial se llama a otros scripts AGI dependiendo de las variables de canal y de registros de la base de datos. De esta manera se pasará de un módulo a otro en la llamada, por ejemplo, de Autenticar Cliente a Menú principal, y luego al submenú que seleccione el usuario.

Como se ve en la figura 1, desde cualquier script o módulo se puede realizar cuatro acciones importantes: redireccionar a otro módulo, realizar acciones a la base de datos, 'decir' algo al usuario, y reconocer lo que el usuario dice.

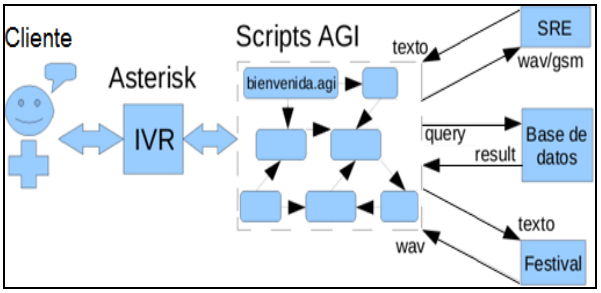

Figura 1. Esquema general de la solución

Un script AGI puede requerir una conexión a la base MySql para realizar registros, actualizaciones o consultas. Esto lo hará ejecutando un query y recibiendo un arreglo con los resultados.

Para comunicarse con el cliente se utilizará un conversor de texto a voz (TTS). El conversor a utilizar será la herramienta text2wave de Festival, este recibe texto y genera un archivo temporal wav que se 'dirá' utilizando la función stream_file() del AGI de Asterisk.

Para reconocer el habla del usuario se utilizará el motor de reconocimiento de voz (SRE) Sphinx2. Desde el script AGI se graba la voz del cliente en un archivo wav temporal que es convertido en texto por la función asr(). En el mismo script se utiliza esta respuesta para decidir qué acción realizar a continuación y de esta manera se navega en los menús.

4.2 Modelo de la base de datos

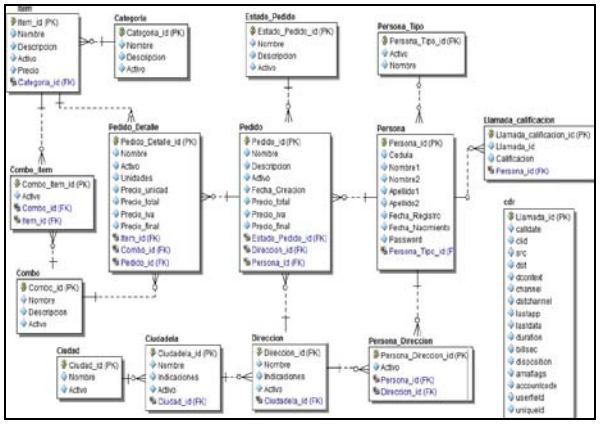

Figura 2. Modelo físico de la base de datos

En el desarrollo de este proyecto nos hemos desligado de tomar los datos personales del cliente, asumiendo que existe un servicio de terceros que nos provee estos datos. Las tablas relacionadas a la dirección existen para copiar esta información y poder utilizarla en nuestra aplicación eventualmente.

La tabla Categoría guarda todas las categorías de los productos que venda el restaurante. Ítem registra cada producto ofrecido indicando su categoría. El precio guardado en esta tabla es previo al IVA.

Todo pedido debe corresponder a una Persona y debe tener una fecha de ingreso y un estado que lo describa.

Un registro en Pedido_Detalle es la selección de un ítem y la cantidad deseada. Los campos Nombre y de precios se guardan aquí de manera redundante para mostrarse en un historial de pedidos. Se debe soportar el caso de que un ítem o combo cambie de precio, entonces en esta tabla se mostrarán los precios existentes al momento de la toma del pedido. Todo detalle debe permanecer a un pedido.

Los registros existentes en Estado_Pedido son: Creado, Confirmado, Comenzado, Despachado, Entregado y Pagado. Cuando el cliente guarda un pedido al confirmarlo, este se ingresa a la base de

datos como Confirmado. El estado Creado lo utilizamos en el desarrollo del proyecto para hacer pruebas ingresando datos directamente a la base. En la aplicación interna los empleados del restaurante tienen el módulo Actualización de estado de pedidos pendientes para cambiar de un estado de pedido al siguiente.

Si el cliente opta por colgar diciendo 'Chao' en cualquier estado de la llamada (a excepción de Seleccionar Items), la llamada se cerrará pero antes se pedirá una calificación del 1 al 10 del nivel de satisfacción de la llamada. No es obligatorio que el cliente responda. En Llamada_calificacion se guardará esta calificación junto al uniqueid de la llamada y el ID de la persona en caso de estar autenticada.

La tabla Cdr es por defecto el registro de detalle de llamadas de Asterisk.

5. Implementación de la solución

5.1 Especificaciones del servidor y software utilizado

En el servidor, se utilizará la versión 5.8 de Perl para los scripts AGI. La aplicación para los empleados será escrita en PHP versión 5.1 y se utilizará el motor de plantillas para PHP Smarty.

Tabla 1. Software utilizado para el funcionamiento de la aplicación en el servidor.

Servidor	
Nombre / Versión	**Descripción**
Centos 5.2	Sistema Operativo
Asterisk 1.8.3.2	PBX / Central telefónica
MySQL 5.0	Gestor de base de datos
Sphinx2	Motor de reconocimiento de voz
Festival	Convertidor de texto a voz
Especificaciones Equipo	Procesador Intel(R) Core(TM)2 Duo T5550 de 1.83GHz con 3GB de RAM y un disco duro de 250 GB

Tabla 2. Software utilizado para el funcionamiento de la aplicación en el cliente.

Cliente	
Software	**Descripción**
Mozilla Firefox 3.5.1 o superior / Internet Explorer 7 o superior	Navegador Web
ZoIPer (u otro)	Softphone

5.2 Módulos implementados en el IVR

Usamos el AGI de Asterisk para ejecutar en el canal los distintos módulos que conforman la llamada. Además logramos comunicarnos con la base, setear variables de canal y atender la entrada de datos por voz o DTMF.

5.2.1. Bienvenida. Contesta la llamada y redirige al módulo 'Autenticar/Registrar Cliente'.

5.2.2. Autenticar/Registrar Cliente. Si el número de cédula del cliente existe en la base, se verifica su contraseña. Si el cliente no existe, se le pregunta si se quiere registrar en el sistema. En el caso positivo, se le pide registrar su contraseña.

5.2.3. Menú Principal. Sus opciones dependen de si el usuario está autenticado o no.

Tabla 3. Opciones del menú principal.

Menú Principal	
No autenticado	**Autenticado**
Menú de categorías	
Autenticarse	Confirmar pedido (Si hay detalles de pedido por confirmar)
	Revisar estado del pedido (si existen pedidos no pagados)
Más Información	

5.2.4. Menú de Categorías. Se consulta las categorías activas; si no hay ninguna se le informa al cliente y se lo envía al menú principal, caso contrario, se las lista. Si no selecciona ninguna categoría y existen detalles para confirmar, se lo redirige a ‘Confirmar Pedido’.

5.2.5. Menú de ítems. Revisa si el cliente está autenticado para redirigirlo a 'Presentar ítems para mostrar' o a 'Listar ítems'.

5.2.6. Listar ítems. Se lista los ítems de la categoría elegida y luego se redirige al usuario al menú de categorías.

5.2.7. Presentar ítems para mostrar. A diferencia del módulo 'Listar ítems', aquí el cliente puede ingresar la cantidad de ítems a comprar. Estos detalles del pedido se acumulan en el archivo temporal.

5.2.8. Confirmar Pedido. Se lista los detalles de pedido existentes para que el cliente pueda optar por anularlos, luego se pide al cliente confirmar su pedido.

5.2.9. Revisar estado de pedidos pendientes. Se lista los pedidos no pagados del cliente indicando su estado, con la opción de revisar los detalles de cada pedido.

5.2.10. Revisar detalles de un pedido. Se consulta a la base los detalles de un pedido y se los lista.

5.2.11. Más información. Se lee secciones de información sobre el restaurante y el IVR.

5.2.12. Chao: Cerrar llamada. Se elimina el archivo temporal si existe y se solicita al cliente calificar su satisfacción con la llamada. Esto es opcional y lo puede realizar sin estar autenticado.

5.2.13. Error. En caso de no cumplir con alguna validación el cliente es redireccionado a este módulo que informa que ha existido un error y envía al cliente al menú principal.

6. Conclusiones

En este sistema se logró reconocer vocabulario del usuario en un canal de llamada a Asterisk y utilizar estos como comandos para dirigir el flujo de la llamada recorriendo las opciones y ejecutando acciones de consulta y registro a la base de datos.

Se utilizó el procedimiento contrario para comunicarse con el cliente, la conversión de texto a voz. Esto se logró utilizando el sistema Festival para 'leer' los mensajes que la aplicación genera para el usuario. El empleo de Festival nos ha permitido armar los mensajes que escuchará el usuario a partir de variables del canal o información obtenida de la base de datos.

Se optimizó recursos al utilizar el Festival para dar retroalimentación al usuario utilizando una voz artificial en lugar de mensajes pregrabados de voz.

No se encontró datos objetivos similares sobre la capacidad de atención de requerimientos concurrente para el servicio de conversión de texto a voz de Festival ni del servicio de reconocimiento de voz del Sphinx2.

No se pudo alcanzar un entendimiento del lenguaje natural porque para esto debíamos contar con un modelo acústico en español y utilizarlo para reconocer un diccionario extenso de posibles palabras y frases del usuario. Esto no se pudo lograr porque no existe un modelo acústico en español para el Sphinx2.

Se puede considerar cumplidos los objetivos de este proyecto ya que el sistema sí es funcional y representativo del modelo de negocios de restaurantes de comida rápida.

8. Recomendaciones

Para que sea factible ofrecer este sistema en un restaurante de comida rápida se debe explorar las limitaciones del Sphinx2 y el Festival experimentando en un ambiente de pruebas para hacer estimados para un ambiente comercial.

Se mejoraría la experiencia del usuario al buscar otras voces artificiales en español porque la actual puede sonar monótona al leer mensajes medianos y largos.

9. Agradecimientos

A todos quienes nos ayudaron en la investigación de motores de reconocimiento de voz y su instalación, también a quienes nos ayudaron a probar la aplicación.

10. Referencias

[1] "Overview CMUSphinx". Última actualización: 2011/05/04 22:10 por 'admin'. Disponible en: http://cmusphinx.sourceforge.net/wiki/tutorialoverview

[2] "Sphinx: An open Source Speech-to-Text Engine" - Julian Dunn, Agosto 29, 2006. Disponible en: http://www.aquezada.com/staff/julian/2006/08/29/sphinx-an-open-source-speech-to-text-engine/

[3] Redial: Interactive Telephony – "Shawn Van Every (2006)". Disponible en: http://itp.nyu.edu/~sve204/redial/week9.html

[4] Sourceforge - CMU Sphinx – "Asterisk PocketSphinx Integration". Disponible en: http://sourceforge.net/projects/cmusphinx/forums/forum/5471/topic/4007682

[5] Asterisk-Dev mailing list - Re: [Asterisk-Dev] Sphinx Integration – Última actualización: Abril del 2004. Disponible en: http://www.mail-archive.com/asterisk-dev@lists.digium.com/msg03067.html

[6] Syednetworks – "Asterisk integration with Sphinx Voice recognition system". Disponible en: http://www.syednetworks.com/asterisk-integration-with-sphinx-voice-recognition-system

[7] "Building Language Model". Última actualización: 2011/08/02 08:05 por 'admin'. Disponible en: http://cmusphinx.sourceforge.net/wiki/tutoriallm#building_statistical_language_model_using_cmuclmtk

[8] "Natural Language Understanding (NLU)" Disponible en: http://ecelab.com/natural-language-u.htm