ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

TESIS DE GRADO

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE BÚSQUEDA DE AUDIO MEDIANTE LA INTEGRACIÓN DE RECONOCIMIENTO AUTOMÁTICO DE VOZ Y BÚSQUEDA POR INDEXACIÓN."

Previa a la Obtención del Título de Ingeniero en Computación
Especialización Sistemas Multimedia

PRESENTADA POR:

VICENTE IGNACIO ORDÓÑEZ ROMÁN

GUAYAQUIL - ECUADOR

2008

AGRADECIMIENTO

Agradezco a todos quienes colaboraron en el desarrollo del presente trabajo de manera especial a Xavier Ochoa director de este proyecto de tesis, y a mis compañeros del Centro de Tecnologías de Información.

DEDICATORIA

A mis padres Abg. Vicente Ordóñez e Ing. Ivonne Román,

a mis hermanos Javier y Nadia,

a Rosa Campaña y Miguel Espinoza.

TRIBUNAL DE GRADO

PRESIDENTE

Ing. Holger Cevallos Ulloa

DIRECTOR DE TESIS

Msc. Xavier Ochoa Chehab

MIEMBROS PRINCIPALES

MSc. Cristina Abad Robalino

MSc. Carlos Monsalve Arteaga

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Vicente Ordóñez Román

RESUMEN

En el presente trabajo se muestran las consideraciones a tener en cuenta para desarrollar tanto un sistema que usa tecnologías de reconocimiento del habla como para sistemas que usen búsqueda por indexación.

Se muestra el estado del arte de las tecnologías de reconocimiento de voz y su evolución a través del tiempo hasta llegar a constituirse en lo que son actualmente, los diferentes usos que se ha dado a este tipo de tecnologías y los problemas que aún falta resolver o mejorar en esta área.

Se documentan las bases teóricas acerca de las tecnologías sobre las cuales funcionará el sistema a ser desarrollado, para con este conocimiento poder utilizar de forma más eficiente los recursos y herramientas disponibles que existen tanto para hacer uso de reconocimiento automático de voz como búsqueda por índice invertido. Además se revisan los estudios y enfoques que han sido realizados previamente encaminados a la integración de estas dos tecnologías, SRAH (Sistemas de Reconocimiento Automático del Habla) y los sistemas de búsqueda por indexación.

Con esta investigación se plantea un marco de trabajo para evaluar un sistema de búsqueda que utiliza reconocimiento automático de voz para el proceso de generación de descripciones. Se ha establecido una arquitectura modelo que describe de forma generalizada los principales componentes que

constituyen un sistema de este tipo y su interrelación estructural, se analizan además las características que permiten evaluar a este tipo de sistemas y categorizar estas características de tal forma que se constituyan en un verdadero marco de evaluación de sistemas de recuperación de documentos de audio.

Se implementa un sistema de búsqueda de documentos de audio siguiendo la arquitectura propuesta, para ello se analizan cada uno de los componentes que la constituyen y las herramientas que se adaptan para ser utilizadas en cada uno de ellos. Se explican detalles técnicos y de implementación y las limitaciones y bondades encontradas en las herramientas disponibles para realizar reconocimiento de voz para su uso dentro de sistemas de búsqueda.

Se presentan los resultados de las pruebas y experimentos realizados sobre el sistema, precisando las condiciones que llevan a obtener mejores resultados y las condiciones sobre las cuales se presenta un grado de precisión menor, así mismo se explican las consideraciones que se han usado o que podrían ser implementadas a futuro para obtener una mejor efectividad y/o eficiencia sobre el sistema.

Finalmente se establecen las conclusiones del presente trabajo de tesis, tratando de establecer la viabilidad de un sistema similar al propuesto, los costos que podría involucrar y las herramientas que podrían utilizarse, se realizan también las recomendaciones para futuros trabajos en el área de

reconocimiento automático de voz o en su integración con sistemas de búsqueda por indexación.

ÍNDICE GENERAL

ABREVIATURAS

ANN	*Artificial Neural Networks*
ASR	*Automatic Speech Recognition*
DTMF	*Dual-tone multi-frecuency*
HMM	*Hidden Markov Model*
IRS	*Information Retrieval Systems*
JSAPI	*Java Speech Application Programming Interface*
JSGF	*Java Speech Grammar Format*
LPC	*Linear Prediction Coding*
LVCSR	*Large Vocabulary Continous Speech Recognition*
LVR	*Large Vocabulary Recognition*
MFCC	*Mel Frecuency Cepstral Coefficient*
NN	*Neural Networks*
OOV	*Out of Vocabulary*
RNN	*Recurrent Neural Network*
RTP	*Real Time Transport Protocol*
SR	*Speech Recognition*
SRAH	Sistema de Reconocimiento Automático del Habla
SRI	Sistema de Recuperación de Información
TF	*Term Frecuency*
TFIDF	*Term Frecuency – Inverse Document Frecuency*
TTS	*Text to Speech*
WER	*Word Error Rate*
WA	*Word Accuracy*
XML	*eXtensible Markup Language*

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Una de las tecnologías que se analizarán y usarán en el presente trabajo son los SRI (Sistemas de Recuperación de Información), que son sistemas que implementan maneras de representar, almacenar, organizar y acceder a ítems de información (1). Este tipo de sistemas cumplen con algunas características que los diferencian de los sistemas de recuperación de datos: el lenguaje de consultas puede ser estructurado o lenguaje natural, la forma de recuperar los documentos se basa en probabilidades, los resultados de las consultas no son exactos, entre otras. Para realizar sistemas de este tipo se ha ocupado mucho esfuerzo para encontrar formas de obtener resultados cada vez más acertados para responder consultas y maneras más eficientes de recuperar la información.

Una de las técnicas más básicas para lograr recuperar información de manera eficiente es mediante la creación de índices o almacenes de índices que contienen colecciones de términos que a su vez poseen listas de apuntadores a ítems de información, lo que permite hacer búsquedas no directamente sobre documentos, lo cual consistiría en una búsqueda intensiva, sino que se busca en el índice de términos; de esta forma se reduce el nivel de complejidad de la búsqueda mediante algoritmos eficientes de acceso a los índices, y mediante técnicas adicionales que permiten calcular la relevancia de cada ítem de información para una consulta dada.

Otra de las tecnologías a utilizar es SRAH (Sistemas de Reconocimiento Automático del Habla), o comúnmente conocidos como sistemas de reconocimiento de voz, estos sistemas reciben una entrada de audio que representa lenguaje humano de manera hablada. El sistema analiza características dentro del audio para determinar un conjunto de propiedades establecidas que le permitan subdividir el audio en unidades más simples que analizar. Los reconocedores de audio modernos utilizan diversos enfoques para analizar y procesar las características del audio, así como para preprocesar el audio antes de ejecutar el reconocimiento. Además, el reconocedor de voz por lo general posee un conocimiento profundo del lenguaje sobre el cual hace el reconocimiento, como: el conjunto de palabras que puede reconocer, la probabilidad de ocurrencia de una palabra tras otra y la estructura de construcción gramatical, para de esta forma mejorar la búsqueda dentro del espacio de solución que suele ser muy amplio especialmente cuando se trata de reconocimiento del habla continuo, es decir sin pausas entre palabras, y cuando el vocabulario es extenso (miles de palabras). En el primer capítulo se ilustra diferentes tipos de problemas que enfrentan las tecnologías de reconocimiento de voz y los diferentes tipos de reconocedores de voz en base a sus capacidades, lo cual nos servirá para establecer el tipo de reconocedor de voz que se integrará con el sistema propuesto.

Los SRI tienen gran popularidad en la actualidad debido a su notable presencia en la Web, tanto así, que los motores de búsqueda actuales hace algunos años ya han indexado billones de documentos disponibles en la Web y su índice sigue creciendo con el tiempo (2) (3). De la misma forma se han construido herramientas de búsqueda por indexación para entornos de escritorio, que permiten realizar búsquedas sobre documentos locales, ejemplos de este tipo de sistemas son Google Desktop, Apple Sherlock y algunos buscadores de archivos integrados en los sistemas operativos más actuales.

CAPÍTULO 1

1 DESCRIPCIÓN DEL PROBLEMA

1.1 Problema

Muy frecuentemente es necesario grabar información hablada en forma de audio digital o en forma de audio/vídeo, en salas de exposiciones, estaciones radiales, centros de convenciones e inclusive en aulas de clase, constituyendo un material con una cantidad de información voluminosa y potencialmente reusable. El problema de la reusabilidad de este tipo de información es que muchas veces no se tiene una transcripción de lo hablado o se tiene información de metadatos muy poco descriptiva acerca del contenido de la información dentro del audio, y esto hace difícil la tarea de buscar información acerca de un determinado tema dentro de grandes cantidades de información hablada contenida en audio.

Sin la tecnología necesaria, la tarea de encontrar información dentro de grandes cantidades de audio, necesariamente implica intervención de un humano que escuche total o parcialmente cada una de las grabaciones de audio de forma que pueda inferir el contenido de cada una, con las limitaciones que puede tener la interpretación humana y el tiempo que implica revisar cada documento. Esto se hace más crítico cuando la cantidad de documentos en audio es más grande y el tipo de consulta más complejo.

Las herramientas para realizar un sistema de este tipo han sido analizadas en algunas ocasiones y se han desarrollado prototipos e inclusive sistemas a pequeña y mediana escala que se han ido mejorando con el tiempo a medida que los sistemas de reconocimiento del habla se han hecho más precisos y robustos, además se han planteado algunas alternativas y enfoques para atacar este problema que serán objetivo de análisis de esta tesis y que ademán serán las bases que serán usadas para la construcción de un sistema que integre una solución como la aquí descrita.

El sistema a desarrollar deberá proveer un componente de análisis o extracción de metadatos de cada documento de la colección de audio que tratará de recolectar la mayor cantidad posible de información del

contexto del documento como de su contenido utilizando reconocimiento automático de voz, con los resultados de este análisis habrá otro componente que servirá para tomar estos resultados y añadirlos a un índice, sobre el cual se podrán hacer búsquedas posteriormente mediante un lenguaje de consultas de documentos.

Existen varios enfoques en la construcción de sistemas de recuperación de información en audio, una de ellas es transformar la información de audio a información que describa sonidos o fonemas, con la restricción que las consultas también deben hacerse de forma hablada, para poder comparar la información a nivel de la descripción de estos fonemas o sonidos, aunque este enfoque aún presenta una serie de limitaciones en cuanto al tiempo que emplea una búsqueda. Otro enfoque es realizar la conversión de voz a texto de los documentos, para de esta forma poder indexar las transcripciones generadas en este proceso como un documento de texto cualquiera, con lo cual se obtiene la misma velocidad de respuesta a las consultas que tendría cualquier buscador de texto. Este último es el enfoque con el cual se desarrollará el presente trabajo.

Además de presentar el sistema, se desarrollará previamente una arquitectura sobre la cual montar este tipo de sistemas y que recoge

además muchas de las recomendaciones presentadas en modelos de arquitecturas implementadas en el área de bases de datos multimedia y recuperación de información multimedial, con lo cual se obtendrá un esquema modularizado y esto a su vez permitirá que sea fácilmente extensible, escalable e integrable con otros sistemas de manejo y administración de información existentes.

1.2 Motivación

"La tecnología multimedia creará un gran impacto en nuestra interacción diaria con las computadoras. Lo que se necesita en este punto son ejemplos de aplicaciones de la tecnología que sean fáciles de usar y que provean demostraciones claras de las ventajas de integrar voz e interfaces multimedia. Los sistemas que tengan éxito probablemente incluirán una variedad de fuentes de conocimiento que pueden complementarse unas a otras"

Alexander Hauptmann

Lo expuesto en esta cita es la principal motivación del presente trabajo, es decir demostrar la integración entre los sistemas actuales

y particularmente los sistemas de administración de información, con tecnologías de procesamiento de voz y aún más la integración con tecnologías de procesamiento de datos multimedia en general, con esto además poder mostrar los problemas o limitaciones que podrían encontrarse cuando se trabaja con estas tecnologías, los enfoques que se han planteado hacia su integración y los estándares existentes para el intercambio, manipulación o sindicación de contenidos multimedia y que son de consideración al elaborar un sistema como el expuesto.

El entendimiento de este tipo de sistemas servirá como soporte para futuras investigaciones en el área de recuperación de información de audio, mejoras al sistema expuesto, otras implementaciones a la arquitectura propuesta, evaluación de sistemas de este tipo utilizando el marco de trabajo expuesto. Este documento servirá además para presentar de una forma clara y precisa a la vez las tecnologías de reconocimiento de voz y sus posibles aplicaciones en sistemas que utilizamos en nuestra vida diaria.

1.3 Justificación

Permitirá divulgar en nuestro medio el estado de desarrollo de los sistemas de reconocimiento de voz, las características que definen a un motor de reconocimiento de voz, los enfoques que utilizan estas tecnologías, y los parámetros que me permiten su evaluación.

El auge de la multimedia hoy en día en diversos campos como el entretenimiento, la información, la educación, la Web, etc. han conllevado a que cada vez más información esté disponible no solamente en texto sino en audio y en vídeo, tanto así que grandes de la información como CNN, *The New York Times*, ABC, NBC han adoptado estándares para sindicación de contenido exclusivamente de audio, aunque en nuestro medio local también podemos encontrar grandes volúmenes de archivos de audio grabado en salas de conferencias, estaciones de radio, grupos de investigación, etc.

No existe una herramienta comercial en nuestro medio que brinde soporte para realizar búsquedas sobre grandes volúmenes de datos de audio, lo que se suele hacer son búsquedas manuales escuchando los archivos de audio uno a uno, el sistema aquí presentado hará que el computador analice de manera automática los

archivos de audio y luego se pueda buscar cualquier archivo presente en la colección basado en la información recolectada por el sistema.

El audio es un medio que permite accesibilidad a usuarios no videntes, por tanto la masificación en el uso de información disponible en audio favorece a estas personas, y este proyecto contribuye de alguna manera a fomentar que la información se disponga en formato audio de forma directa sin que esto impida que la información sea posteriormente recuperada con relativa facilidad.

Si bien sobre documentos de texto una búsqueda puede realizarse usando indexación para mejorar el tiempo de respuesta de la búsqueda, en un sistema de búsqueda de audio por contenidos sería la única manera útil de hacerlo pues de otra forma el tiempo de demora en la búsqueda sería demasiado extenso, pues tardaría un tiempo aproximadamente igual a la suma de el tiempo de duración de todo el conjunto de archivos de audio, usando indexación sería una búsqueda tan rápida como hacerla sobre documentos de texto. Este trabajo permitirá exponer de manera práctica estos criterios.

1.4 Alcance

Se presentarán los principales conocimientos y consideraciones que se deben tener en cuenta al elaborar un sistema de recuperación de información en audio de tal forma que sirvan como material de referencia para el desarrollo de los capítulos subsiguientes u otras investigaciones, además se definirá una arquitectura propia que en base a trabajos previos debidamente documentados y a la investigación aquí realizada cumpla con la mayor parte de requisitos indispensables y deseables para un sistema de este tipo, de tal forma que pueda ser implementada con diferentes variantes y usando distintas plataformas de implementación.

Se planteará un marco de trabajo para evaluar un sistema con las características descritas y basado en la arquitectura aquí propuesta o sus variantes, para que se constituya como una herramienta útil para evaluación de diversas características en base a las métricas aquí definidas.

Se implementará un buscador de documentos locales, con la restricción que los documentos de audio en lo posible corresponderán a un locutor único, pudiendo ser extensible a dar soporte para diferentes idiomas, dependiendo del motor de reconocimiento que se

esté usando para el efecto, por ello el sistema será independiente de la tecnología de reconocimiento de voz. El audio deberá estar muestreado a una frecuencia mínima de 22KHz y un tamaño por cada muestra de 16bits; es decir, con un SNR *(Signal to Noise Ratio*) de aproximadamente 100db, con lo cual el sistema podrá soportar sin problemas grabaciones de conferencias, noticias, clases a distancia, aunque no podrá utilizarse para grabaciones por teléfono, celular o calidades inferiores.

1.5 Objetivos

A continuación se enuncian los objetivos del presente trabajo:

- Investigar el estado del arte de las tecnologías de reconocimiento automático de voz.

- Analizar y diseñar una arquitectura que permita extraer transcripciones de grandes cantidades de archivos digitales de audio para luego almacenarlas de tal forma que posteriormente se puedan efectuar búsquedas de manera eficiente.

- Implementar un prototipo de la arquitectura propuesta utilizando un motor de reconocimiento de voz disponible en el mercado.

- Diseñar e implementar un modelo de aplicación que permita la integración de los componentes de esta arquitectura con un sistema ya existente.

- Implementar un buscador de archivos de audio accesible como una aplicación Web y que utilice la arquitectura propuesta

Resumen

En el presente capítulo se ha mostrado la problemática a resolverse y brevemente los enfoques encaminados hacia el planteamiento de una solución, algunas motivaciones personales, de interés institucional y general que generan el planteamiento de este trabajo de tesis. Posteriormente se presentaron los alcances y objetivos que persigue el presente trabajo. En el siguiente capítulo se presentan los fundamentos teóricos necesarios para realizar una propuesta de diseño.

CAPÍTULO 2

2 FUNDAMENTOS TEÓRICOS

2.1 Generalidades

Se han identificado dos problemas principales a resolver dentro del desarrollo general del sistema, uno de ellos es poder utilizar reconocimiento automático del habla sobre la información contenida en audio, y el otro es utilizar esta información de tal forma que pueda ser recuperada de manera eficiente. En los siguientes subcapítulos se presentarán los principales fundamentos en que se basan estas tecnologías.

2.2 Sistemas de reconocimiento automático de voz

El problema de reconocer voz ha recibido diversos enfoques para su solución a través del tiempo y dependiendo del problema específico

de reconocimiento de voz que se propone resolver. Existen muchas características que convierten el problema de reconocimiento de voz en algo mayor o menormente complejo. Por ejemplo, el reconocimiento de palabras aisladas de entre un conjunto de palabras pequeño comparado con el reconocimiento del habla conversacional con la participación de múltiples locutores. Para intentar resolver muchos problemas de reconocimiento de voz se ha recurrido a esfuerzos multidisciplinarios que han involucrado: ciencias de la computación, lingüística, reconocimiento de patrones, física acústica, fonética, procesamiento de señales, teoría de las comunicaciones y la información, fisiología y psicología. Se analizarán los sistemas de reconocimiento que permiten reconocer habla continua con vocabulario extenso y modelos de lenguaje probabilísticos, brevemente se describirán los sistemas de reconocimiento de comandos por voz que también forman parte de esta tecnología pero que restringen el vocabulario a menos palabras y cuya gramática se limita a definir un conjunto finito de reglas.

2.2.1 Historia

30's – 40's

Homer Dudley de los laboratorios Bell propone un modelo de sistema de análisis del habla y síntesis del habla (4).

El Departamento de Defensa (DoD) de los Estados Unidos empieza a patrocinar un proyecto para traducir de forma automática lenguaje hablado, aunque el proyecto no tuvo éxito, hizo que muchas instituciones tuvieran interés en el área.

50’s – 60’s

En los laboratorios Bell, se desarrolla el primer sistema capaz de reconocer dígitos en el habla por teléfono.

En esta época se hicieron los primeros sistemas para reconocer palabras aisladas, con vocabularios pequeños.

70’s – 80’s

Al comienzo de los años 70, en la Universidad de Carnegie-Mellon se desarrolla un sistema capaz de reconocer oraciones completas con limitadas estructuras gramaticales, además el sistema reconocía habla discreta, donde era necesario que existan pausas entre las palabras (5).

En los años 80 hubo un gran progreso, la tasa de error se reducía a la mitad cada 2 años.

En 1985, en IBM se desarrolla un sistema capaz de reconocer palabras aisladas en tiempo real con un vocabulario de 20000 palabras, el sistema necesitaba entrenamiento y su tasa de error era menor al 5%.

AT&T implanta un sistema para ruteo de llamadas mediante la extracción de palabras claves del habla, usando reconocimiento de voz independiente del locutor.

En esta época ya se hicieron sistemas con vocabularios medianos y extensos, aparecen los primeros esfuerzos por resolver el problema del habla continua, se usaron inicialmente comparación con patrones, luego se paso a utilizar modelos estadísticos y Modelos de Markov (6).

Años 90

Aparecen las primeras aplicaciones comerciales, en 1997 Dragon Naturally Speaking sale al mercado como el primer paquete de reconocimiento continuo de voz.

ViaVoice de IBM también aparece en esa época, permitiendo dar comandos al computador por voz, y soportando varios sistemas operativos, entre estos OS/2 (7).

Se desarrollaron sistemas de vocabulario muy extenso, se hicieron sistemas de reconocimiento del habla continua. En esta época se crearon modelos de aprendizaje estadístico combinando el uso de redes neuronales con Modelos de Markov.

2.2.2 Generalidades

Los sistemas de reconocimiento de voz pueden tener diversos alcances dependiendo del problema que intentan resolver, este problema varía dependiendo de ciertas características que se han resumido a continuación:

Vocabulario: Podría estar limitado a unas cuantas palabras o tener muchas palabras o al menos varias cuyo sonido al ser pronunciadas es similar.

Manera de hablar: Si existen pausas entre las palabras o se habla de forma continua, sin pausas.

Estilo del habla: Si el sistema permite que hayan errores en la pronunciación, o reconoce palabras que no existen en el diccionario.

Dependencia del locutor: Algunos sistemas requieren entrenamiento previo por parte del locutor antes de empezar a reconocer el habla del mismo. Por otro lado están aquellos que no necesitan entrenamiento y reconocen el habla de cualquier locutor.

Modelo del Lenguaje: Sirve para restringir el conjunto de combinaciones de palabras probables en un momento dado. Se suelen definir en forma de reglas fijas o puede ser sensitivo al contexto.

En la Tabla I se resumen los parámetros principales que caracterizan a un sistema de reconocimiento del habla y el rango de valores que podría haber en cada característica desde el problema más simple al más complejo.

Parámetros	**Rango**
Manera de hablar	Desde palabras aisladas hasta habla continua
Estilo del habla	Lectura o habla espontánea
Involucración del usuario	Dependiente del usuario o Independiente del usuario
Vocabulario	Pequeño (< 20 palabras) a Extenso (>50 mil palabras)
Modelo de lenguaje	De estado finito, hasta Sensitivo al Contexto
SNR	Alto (>30dB) hasta Menor (<10dB)
Transductor	Micrófono con cancelación de ruido, hasta audio producido con teléfonos celulares

Tabla I. Parámetros que caracterizan a los SRAH (Sistemas de Reconocimiento Automático del Habla) (8).

2.2.3 Reconocimiento de comandos por voz

Revisando las características descritas en el apartado anterior, los sistemas de reconocimiento de comandos por voz se caracterizan porque su vocabulario suele ser pequeño o mediano, dependiendo de la cantidad de comandos. El modelo de lenguaje se define mediante una gramática de estados finitos. Se definen algunas cuantas reglas, por lo general con respecto al orden o la secuencia de palabras esperadas. El modo de hablar es palabras o frases cortas aisladas. Todas estas características hacen que este sea un problema menos complicado de resolver comparado con el reconocimiento de habla continua, por lo cual con frecuencia estas implementaciones tienen también independencia del locutor y mayor tolerancia al ruido.

Esta tecnología se usa entre otras cosas para dar comandos en edificios inteligentes, trabajos donde se tienen manos y vista ocupada, para ayudar a personas con discapacidades, etc.

Se han tratado de definir estándares para escribir aplicaciones que usen este tipo de interfaz por comandos de voz, para definir la gramática específica para cada aplicación y la interacción entre la

aplicación y el usuario final. Este tipo de estándares también definen salidas por voz, por lo general mezclan también el uso de TTS (*Text to Speech*), texto a voz.

Entre los principales estándares o convenciones que se han definido están (9):

VoiceXML: Estándar XML definido por la w3c y que define la interacción entre el usuario y la aplicación con entradas y salidas por voz, es interpretado por un *navegador por voz*. También existe una implementación conocida como X+V que permite agregar reconocimiento de voz en documentos XHMTL.

SALT (*Speech Application Language Tags*): Competidor de VoiceXML, define como agregar reconocimiento de voz en documentos XHTML. Entre otros, Microsoft ha soportado este formato además de VoiceXML para el desarrollo de este tipo de aplicaciones Web.

SRGS (*Speech Recognition Grammar Specification*): Estándar XML de la w3c para definir la gramática que acepta una aplicación,

lo cual sirve para restringir el espacio de búsqueda cuando se hace el reconocimiento de los comandos.

JSGF (*Java Speech Grammar Format*): Propuesto por Sun Microsystems, derivado de SRGS. Permite definir una gramática con el estilo y convenciones de Java y las maneras tradicionales de definir gramáticas.

Este tipo de aplicaciones no serán utilizadas en este trabajo; sin embargo, se utilizan frecuentemente también para el diseño de interfaces multimodales en sistemas de recuperación de información. Además el reconocimiento de comando por voz utiliza muchas de las técnicas utilizadas también en reconocimiento automático del habla continua pero sin muchas de las complejidades de este tipo de reconocimiento.

2.2.4 Reconocimiento del habla continua

Habla continua significa que no existen pausas entre las palabras. El sistema que reconoce habla continua es capaz de reconocer unidades de lenguaje hablado completas basándose en las restricciones de los lenguajes. Estas unidades de lenguaje hablado

se conocen como *enunciados* y generalmente están separados por silencios.

2.2.4.1 Complejidades

En la Fig. 1 se muestra el modelo general de un sistema de reconocimiento del habla. Como entrada se tiene la señal de voz en forma digital, esta señal ingresa como entrada de un componente que analiza sus características. En primera instancia el sistema segmenta el audio en *enunciados*, para que luego la señal de audio de cada *enunciado* pueda sea analizada y obtener así una representación.

El problema de reconocimiento de audio se puede dividir en tres problemas más simples:

1. Encontrar la manera de representar la señal de audio con sus características.
2. Definir la forma en que se modelan las restricciones del lenguaje, acústicas, vocabulario, semántica, contexto.
3. Buscar la respuesta más óptima entre las posibles.

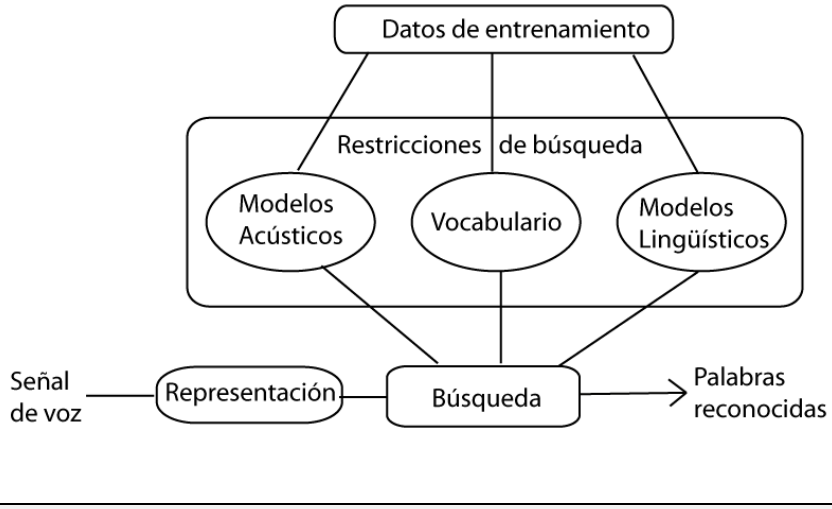

Figura 1. Modelo general de un sistema de Reconocimiento Automático del Habla (SRAH).

Para reconocer palabras en una señal de voz se tiene el conocimiento de los tipos de sonido que se pueden encontrar el lenguaje hablado. Las características físicas de los sonidos en la naturaleza dependen de la caja de resonancia donde se producen. En el caso de la voz humana, para llegar a un entendimiento de los sonidos del habla se han hechos estudios del tracto vocal, el cual restringe los sonidos que puede producir. Si el problema se restringe al lenguaje hablado entonces la cantidad de sonidos y el tipo de sonidos que se analizan se restringe aún más (10), así en algunos idiomas el sonido se restringe a:

- El de las vocales del idioma

- El de las consonantes sonoras
- El de consonantes nasales
- El de consonantes continuas

En (11) se hacen otras subdivisiones basadas en el uso de diptongos, semivocales y silencios en el inglés americano. El estudio de las características de cada uno de estos tipos de sonidos presentes en el lenguaje hablado se hace a través de un análisis en frecuencia o análisis espectrográfico.

El espectrograma permite visualizar las frecuencias que componen a cada sonido producido en el habla así como las características que servirán para asociarlo con los tipos de sonido mencionados antes y a cual sonido específico del lenguaje hablado se aproxima más. En la Fig. 2 se observa un gráfico que muestra con una línea verde las variaciones en amplitud a través del tiempo y el espectrograma de la señal de audio.

En este nivel de procesamiento se puede aplicar restricciones de tipo fonotáctico, que se refiere a combinaciones de sonidos que no suceden en el lenguaje hablado sobre el que se hace reconocimiento. En el caso de idioma español por ejemplo no existe

la secuencia de consonantes “lrt” por tanto esa respuesta sería descartable y se optaría por la siguiente más probable que sí cumpla con las restricciones del lenguaje español hablado. Este análisis es conocido como análisis acústico-fonotáctico.

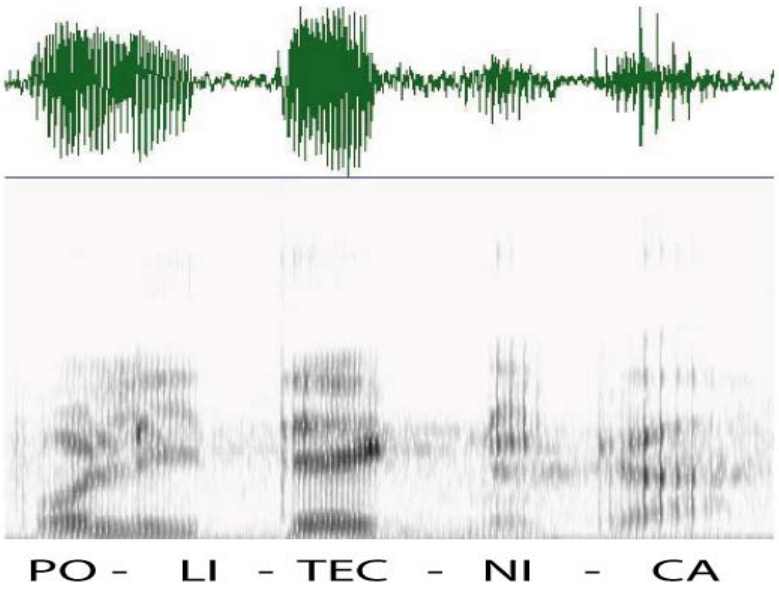

Figura 2. Visualización de amplitud en tiempo y espectrograma de la palabra *politécnica*, (Generado usando el software UCL Enhance de Mark Huckvale, University College of London).

2.2.4.2 Enfoques hacia la solución del problema

Rabiner & Juang (11) han establecido tres enfoques utilizados para hacer reconocimiento automático de voz:

1. **Análisis acústico fonotáctico**: Se basa en que en los lenguajes existe un número finito y distinto de unidades fonéticas en el lenguaje hablado y que cada una tiene características que las hace diferenciables entre sí. Este es el enfoque que se había mencionado previamente.

2. **Mediante reconocimiento de patrones:** Se basa en que si se tiene un conjunto de muestras o datos de entrenamiento previos se puede hacer que el sistema caracterice los patrones, sin la necesidad del conocimiento fonotáctico del enfoque anterior. En un siguiente paso se realiza la clasificación de sonidos para asociarlos a fonemas mediante la comparación con los patrones aprendidos durante la fase de entrenamiento.

3. **Mediante inteligencia artificial**: Es un híbrido entre los dos enfoques anteriores combinando el conocimiento del análisis fonético-acústico con el reconocimiento de patrones. Hace uso de un sistema experto para simular el análisis que haría un humano para decidir en base a las mediciones y parámetros obtenidos con los métodos fonético acústicos y conocimientos léxicos, lingüísticos y semánticos.

El sistema Sphinx-4

Un sistema de reconocimiento automático de voz independiente del locutor y de habla continua usado en muchas investigaciones como un ejemplo del estado del arte de las tecnologías de reconocimiento de voz es el sistema Sphinx desarrollado originalmente en la Universidad de Cargenie-Mellon.

Sphinx-4 es un marco de trabajo para desarrollar sistemas de reconocimiento automático de voz definiendo el prototipo y las interacciones entre los componentes que constituyen un SRAH. Sphinx-4 se constituye también como un sistema de reconocimiento de habla continua basado en Modelos Ocultos de Markov al proveer varias implementaciones para cada uno de los componentes definidos en el marco de trabajo (12). Se utilizará la arquitectura de este sistema como caso de estudio para mostrar las funcionalidades de un sistema de reconocimiento de habla continua, a continuación se discutirá la organización de este sistema y las funciones de sus componentes principales, ver Fig. 3.

Los tres módulos principales dentro de la arquitectura de Sphinx-4 son:

El *Frontend* o módulo de entrada

Este componente recibe la señal de entrada, es el encargado de transformarla en una secuencia de salidas encapsuladas en un objeto. Este módulo puede ser implementado para utilizar algunos tipos de representación característicos para señales de audio, es decir el formato digital como se transmitirá la información. La representación por defecto de este componente en el sistema Sphinx-4 es *Coeficientes Cepstrales en Frencuencia de Mel* (MFCCs) que consiste en representar la señal usando su representación en frecuencia escalado de manera logarítmica de acuerdo a la escala *Mel* que es una escala que se basa en la percepción auditiva humana (13). Otro tipo de representación para la señal es el uso de *Predicción Lineal Perceptual* (PLP), otra representación basada en la capacidad auditiva del ser humano.

La naturaleza configurable del sistema Sphinx-4 permite especificar el recurso que proveerá la señal de entrada para el sistema, un micrófono, un archivo de audio PCM (*Codificaciòn por Modulaciòn de Pulsos*). Un subcomponente interno a este componente implementa tareas comunes de procesamiento de señales, entre estas: transformada de Fourier rápida, transformada del coseno, extracción de los MFCCs, extracción de los coeficientes PLP,

distintos tipos de normalización a la señal de entrada y las señales resultantes. Muchas de estas representaciones se basan en los *vocoders* que son los métodos de codificación que se usan para la transmisión de voz.

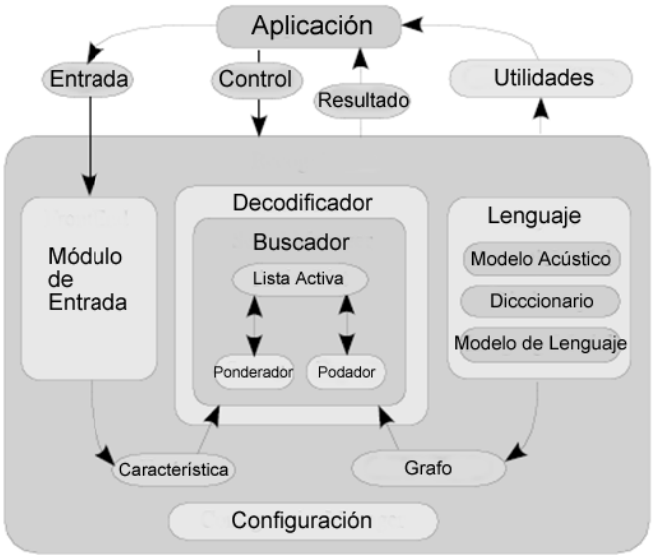

Figura 3. Arquitectura e interacción entre componentes del sistema Sphinx-4. Tomado de (12).

El módulo de lingüística

La salida del módulo de lingüística es un grafo que contiene el conjunto de posibilidades de respuesta para el estado actual del reconocimiento. Este espacio de búsquedas es generado en base a las restricciones que están definidas en este módulo y que se basan

en algunas de las características mencionadas anteriormente como la fonotáctica, el vocabulario, las restricciones del lenguaje. Estas restricciones se modelan en tres subcomponentes.

Modelos lingüísticos

Existen dos tipos de modelos lingüísticos, el más simple es aquel que consiste en un grafo dirigido de estados en donde cada estado representa una palabra y cada arco dirigido representa el orden de ocurrencia de cada palabra, este tipo de modelo es generado típicamente definido con gramáticas de estados finitos, ver Fig. 4.

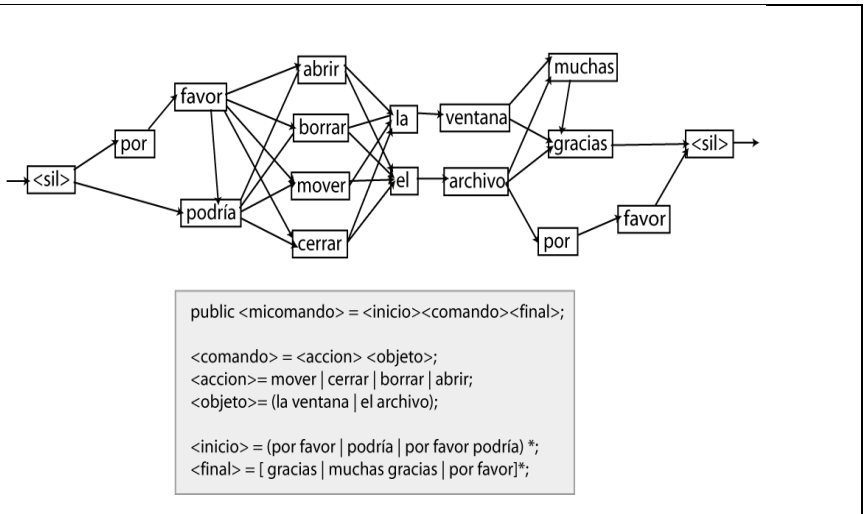

Figura 4. Ejemplo de un grafo generado con una gramática de estados finitos usando notación JSGF (Java Speech Grammar Format).

El segundo tipo de grafo es un modelo de tipo estocástico. No define las relaciones exactas de precedencia de palabras, pero define la probabilidad de una palabra basándose en las *n-1* palabras anteriores.

Para conocer esta probabilidad se hace uso de n-gramas, un n-grama es una subsecuencia de n símbolos de una secuencia de símbolos, en este caso de palabras. Las estructuras que usa Sphinx-4 para implementar los grafos con el uso de n-gramas son los Modelos Ocultos de Markov (HMM, *Hidden Markov Models*), este tipo de estructura basado en estados permite modelar las probabilidades condicionales de múltiples sucesos anteriores que se necesitan para definir la siguiente salida. Definir estas probabilidades de palabras basadas en las anteriores cuando se desea hacer reconocimiento de lenguaje requiere analizar una gran cantidad de información (corpora) para poder determinar estos valores probabilísticos y modelarlos en un HMM.

Diccionario

Este módulo subdivide las palabras en unidades más pequeñas que están representadas en el modelo acústico, ver Tabla II.

Palabra	**Representación**
áCIDO	A S I D O
éLITE	E L I T E
éPOCA	E P O K A
óRDENES	O R D E N E S
CABE	K A B E
CABELLO	K A B E LL O
CADA	K A D A
ESCAñOS	E S K A GN O S
ESCARMIENTA	E S K A R M I E N T A
INSULARES	I N S U L A R E S
MéDICO	M E D I K O
OPUESTAS	O P U E S T A S
ORíGENES	O R I J E N E S
PERTURBACIóN	P E R T U R B A S I O N
RATóN	RR A T O N
REAL	RR E A L
RECHAZAN	RR E CH A S A N
SUBDIRECTOR	S U B D I R E K T O R
SUBSAHARIANA	S U B S A A R I A N A
SUCEDIDO	S U S E D I D O
SUEñO	S U E GN O
SUIZA	S U I S A
SUMARIO	S U M A R I O

UVAS	U V A S
VEHíCULOS	V E I K U L O S
ZANAHORIAS	S A N A O R I A S
ZUMO	S U M O
_T_L_C	T E E L E S E
_T_V	T E V E

Tabla II. Una muestra de palabras para un diccionario de idioma español, con su correspondencia en unidades representadas en el modelo acústico.

Cada subdivisión de la palabra está asociada a un símbolo, estos símbolos deben estar definidos en el modelo acústico. El diccionario sirve para relacionar el modelo lingüístico con el modelo acústico.

Modelos acústicos

Los modelos acústicos mapean las unidades habladas con el HMM que será ponderado con las características de la señal de entrada que se reciben del *FrontEnd*. En los modelos acústicos cada unidad de sonido tiene un HMM asociado, esta información se combina con la información del contexto de las palabras y los modelos lingüísticos para generar el grafo de búsqueda donde actuará el siguiente componente. Para generar los HMM de cada unidad de

sonido se requiere también datos de entrenamiento para asociar características a cada unidad de sonido. Estos datos de entrenamiento conocidos como *speech corpora* son datos en audio con transcripciones alineadas en el tiempo.

El Decodificador

Su función es usar las características obtenidas del *FrontEnd* en conjunto con el grafo de búsqueda para generar hipótesis acerca de los resultados encontrados recorriendo el grafo y evaluando los HMMs. En este componente existen algunos subcomponentes que ayudan en el proceso de recorrer eficientemente el grafo. Al final del proceso este componente retorna el resultado más acertado y las siguientes hipótesis con su respectiva probabilidad.

Describiendo los componentes del sistema Sphinx-4 se ha descrito en términos generales el funcionamiento de un sistema de reconocimiento automático de voz genérico, extensible y que demuestra el estado del arte de esta tecnología.

2.2.5 Problemas comunes

2.2.5.1 Factores que influyen en la precisión

Muchos de los sistemas necesitan entrenamiento por parte del usuario para funcionar de manera acertada. Algunos sistemas que no requieren entrenamiento previo por ser independientes del locutor no rinden de manera precisa debido a la pobreza del *corpus acústico* que se utilizó para entrenar al sistema.

El estilo de hablar influye en el proceso de reconocimiento, esto tiene que ver con los distintos acentos que se suelen hablar de un mismo idioma, así el idioma español de Argentina pronuncia de manera diferente ciertas unidades de sonido que el idioma español de España o de Ecuador.

El uso frecuente de palabras que no se encontraban en los datos de entrenamiento del SRAH, o el uso de modismos propios de alguna región también afectan la precisión, pues el reconocedor confundirá los términos desconocidos con términos que sí conoce.

La presencia de ruido también afecta un buen reconocimiento, muchos sistemas recomiendan un nivel calidad mínima en el

transductor o en el audio de entrada, por ejemplo en el caso de micrófonos podrían recomendar micrófonos con cancelación de eco.

Cuando existen múltiples locutores podría existir imprecisión debido a que el sistema muchas veces trata de normalizar la voz asumiendo que está presente un solo locutor.

2.2.5.2 Limitaciones y restricciones

Si bien existen SRAH como Sphinx-4, que implementan técnicas de múltiples disciplinas y con mucho potencial de escalabilidad y una arquitectura que permite el desarrollo independiente de cada componente. Para que este tipo de sistemas puedan lograr reconocer de manera confiable se necesitan grandes volúmenes de datos de entrenamiento.

Para el caso de los modelos lingüísticos se necesitan grandes cantidades de texto en el idioma que se va a hacer el reconocimiento. Con este texto se pueden obtener bigramas, trigramas y n-gramas para poder estudiar las probabilidades de ocurrencia basándose en las palabras anteriores. El uso de este tipo de *corpus* o datos de entrenamiento no es exclusivo en el área

de reconocimiento del habla, sino en general en el estudio de procesamiento de lenguaje natural y recuperación de información. Estas grandes cantidades de datos hace un tiempo no eran tan accesibles por grupos de investigación, con el auge del Internet y la existencia de *crawlers*, que son programas que recorren la Web a través de los hipervínculos, se ha recolectado grandes volúmenes de texto para ser utilizado en sistemas de procesamiento de lenguaje natural.

En el caso de los modelos acústicos, los datos de entrenamiento o *corpus acústicos* son más escasos. Estos datos de entrenamiento sirven de manera análoga a los modelos lingüísticos para obtener bifonos y trifonos para saber las secuencias de sonidos consecutivos probables en un determinado idioma, además para generar el HMM de cada unidad de sonido del lenguaje hablado. Cuando se requiere que el reconocimiento de voz sea independiente del locutor, son necesarias muchas horas de grabaciones habladas con una calidad de audio aceptable y además deben estar disponibles las transcripciones de esas grabaciones y éstas deben alinearse en el tiempo. Este tipo de datos de entrenamiento son los que cuesta más trabajo generar, por la laboriosidad que implica. Aunque se han hecho esfuerzos para

que la generación de este tipo de datos de entrenamiento sea más fácil de hacer, se ha construido software para alinear las transcripciones al audio y software que alinea de manera automática las transcripciones (14) (15). Esta es la mayor de las limitantes en los sistemas de reconocimiento automático de voz libres. Por esta razón se han creado muchos esfuerzos como VoxForge y acuerdos de cooperación entre diversas instituciones para lograr obtener *corpus acústico* en múltiples idiomas y de calidad para ser usado en estos sistemas (16).

Otra de las limitantes que resalta Rabiner & Juang es la naturaleza disciplinaria que se requiere para resolver el problema de reconocimiento automático del habla (17), pues se necesita expertos en distintas ramas. Se requiere principalmente conocer de lingüística, acústica, fonética, fisiología humana, inteligencia artificial, estadística, psicología, recuperación de información, aprendizaje de máquina, procesamiento de lenguaje natural y todo este conocimiento es imposible que una sola persona los pueda poseer, así que durante los años muchas de las investigaciones destaca han estado fuertemente apegadas a una u otra disciplina, lo cual ha retrasado algunas veces el desarrollo de las tecnologías

de reconocimiento de voz.

2.2.6 Perspectivas a futuro

Se generarán grandes volúmenes de *corpus acústico* útil en varios idiomas para realizar reconocimiento automático de voz de forma independiente del locutor.

Se eliminarán algunas restricciones actuales como el tamaño del vocabulario, o la dependencia del contexto, dependencia del estilo del habla, del acento.

Los sistemas se harán más robustos en el reconocimiento de voz en condiciones de ruido, voz por teléfonos celulares, voz a través altavoces, eco, ver Fig. 5.

Se harán estudios para generar automáticamente el léxico o el diccionario, pues si bien se tienen herramientas para modelar los Modelos Ocultos de Markov con los datos de entrenamiento necesarios, sigue siendo un reto generar el diccionario indicando las unidades de sonido de forma completamente automática.

Se harán esfuerzos por lograr generar rápidamente modelos de lenguaje adaptados a tareas específicas dentro de un contexto conocido. Por ejemplo si se desea hacer reconocimiento sobre grabaciones científicas de Biología. Generar un submodelo de lenguaje que se adapte a ese contexto, lo que haría que el sistema falle menos y sea más eficiente.

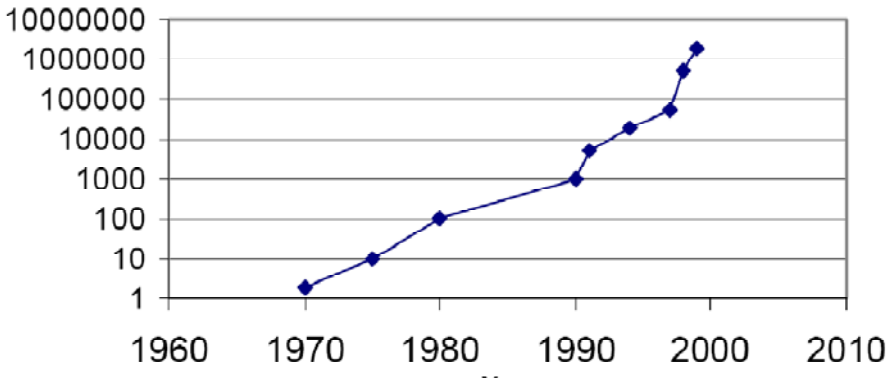

Figura 5. Crecimiento de la cantidad de palabras que pueden reconocer los SRAH. Tomado de (11).

Se desarrollarán sistemas multilingües, además se mejorarán los sistemas de traducción automática donde el individuo habla al sistema y este responde el mensaje de manera hablada en otro idioma.

2.3 Sistemas de recuperación de información

Los sistemas de recuperación de información tienen como objetivos idear la manera de representar, almacenar, organizar y acceder a ítems de información. Los ítems de información podría ser documentos estructurados como un libro, que posee capítulos, subcapítulos, o un documento no estructurado como una información en texto plano.

Una idea en los sistemas de recuperación de información utilizada inclusive desde mucho antes del tratamiento computarizado de datos es el uso de índices para acceder rápidamente a información. En los sistemas de recuperación de información se automatiza la creación de estos índices a través de palabras claves que describen cada documento o utilizando todas las palabras que contiene el documento.

Baeza-Yates y Ribeiro-Neto dan la siguiente definición formal para el modelo de recuperación de información (18):

“Un modelo de recuperación de información es la cuádrupla [D , Q , F , $R(q_i,d_j)$] donde:

(1) D es un conjunto compuesto de representaciones lógicas de los documentos en la colección.

(2) Q es un conjunto de representaciones lógicas de las necesidades de información del usuario. Estas representaciones son llamadas consultas o queries.

(3) F es un marco de trabajo para modelar las representaciones de los documentos, consultas y sus relaciones.

(4) $R(q_i,d_j)$ es la función de relevancia que asocia un número real con una consulta $q_i \in Q$ y una representación de un documento $d_j \in D$. Este ranking define un orden en los documentos concernientes a la consulta qi"

A continuación se describen algunas de las estrategias que se han usado y se están usando en los sistemas de recuperación de información, entre los principales están: teoría de conjuntos, álgebra vectorial y el enfoque probabilístico.

Teoría de conjuntos

Emplea lógica binaria, para cada documento se genera un conjunto de términos. Si un término está presente en la consulta entonces el valor de relevancia es 1, si el término no se encuentra en la consulta entonces su valor de relevancia es 0. Se pueden hacer consultas

empleando operadores de conjuntos como unión (*or*) o intersección (*and*).

Modelo algebraico vectorial

En este enfoque tanto las consultas como la representación de los documentos son consideradas como vectores en el espacio vectorial de los términos. La similaridad se calcula basándose en la similaridad de los vectores. Esta similaridad se expresa comúnmente en términos de la magnitud de la proyección de la consulta del usuario sobre la representación del documento.

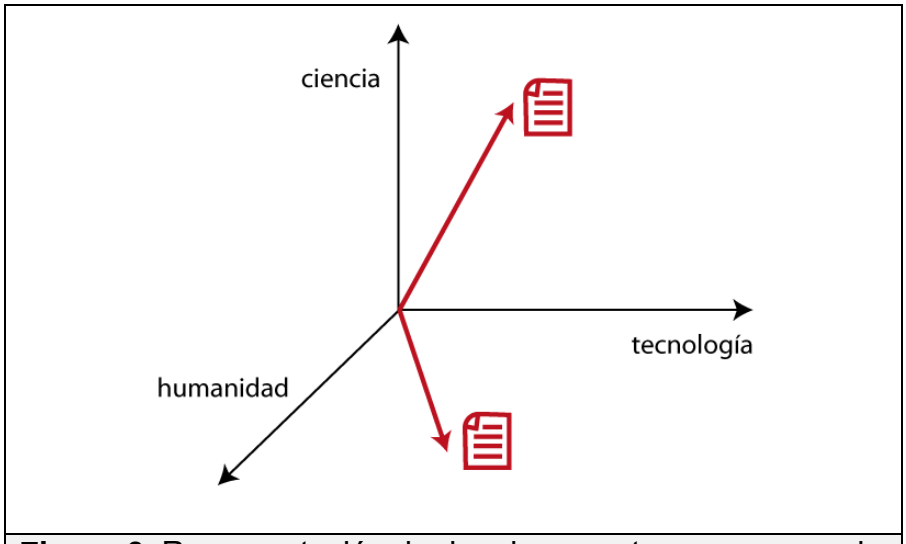

Figura 6. Representación de dos documentos en un espacio de tres dimensiones en el dominio de los términos (humanidad, tecnología, ciencia).

Modelo probabilístico

En este enfoque se calcula para cada término la probabilidad de aparecer en cada documento. Cuando se envía una consulta al sistema se retornan los documentos que contienen ese término y la similaridad se calcula en base a la probabilidad combinada de todos los términos coincidentes.

Otros enfoques

Conjuntos difusos: Un conjunto difuso es una colección de elementos que tienen asociado un valor que representa su nivel de membresía al conjunto. En este enfoque cada documento es mapeado a un conjunto difuso de términos. Cuando se realiza una consulta se usa este número asociado a cada elemento conjuntamente con operaciones de conjuntos difusos para calcular la relevancia de los documentos.

Redes neuronales: Se hace uso de redes neuronales para determinar la similaridad entre una consulta y un documento de la colección. Necesitan entrenarse previamente con respuestas a consultas con documentos relevantes y no relevantes.

Algoritmos genéticos: Inicialmente se estiman ponderaciones para cada término de la consulta o se asignan estas ponderaciones de manera aleatoria, luego se generan nuevas consultas modificando estos valores de ponderación. Se eliminan las consultas que no están cercanas a documentos relevantes conocidos y solo se mantienen en la iteración las que sí lo están (19).

Modelos de lenguaje: Se analiza el texto de cada documento y se generan modelos para poder evaluar para cada documento la probabilidad de generar la consulta que sea solicitada.

2.3.1 Uso de indexación

Existen dos estrategias principales para realizar búsquedas sobre un conjunto de ítems de información, búsqueda en línea y búsqueda por indexación.

La búsqueda en línea consiste en acceder de forma secuencial al texto de cada documento y buscar de forma intensiva algún patrón que coincida con el criterio de búsqueda o la consulta. Las técnicas por indexación consisten en crear estructuras basadas en la

colección de ítems de información mediante las cuales recuperar de manera rápida subconjuntos de esta colección que cumplan con el criterio de búsqueda.

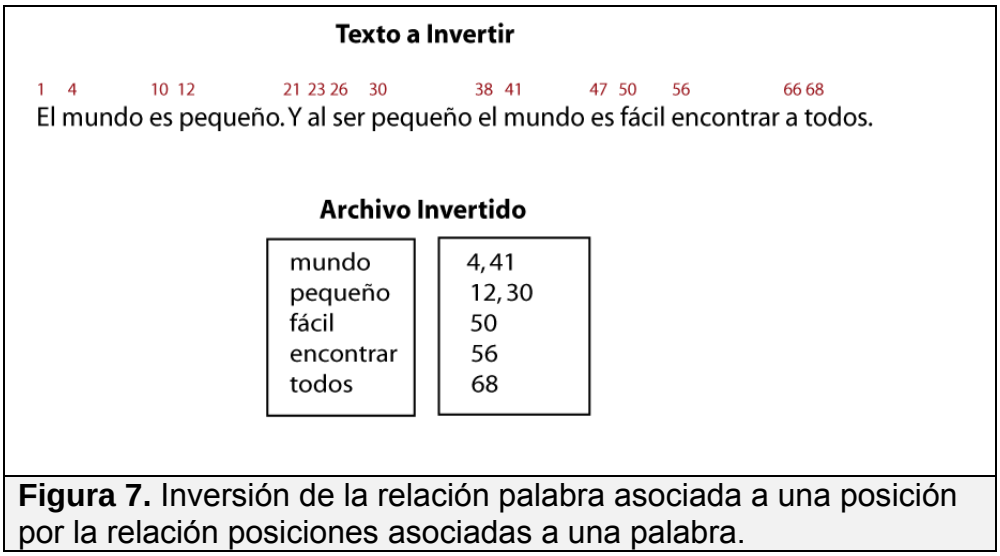

Figura 7. Inversión de la relación palabra asociada a una posición por la relación posiciones asociadas a una palabra.

Una complejidad de las técnicas de búsqueda por indexación consiste en mantener actualizada la estructura que permite realizar las búsquedas con respecto a la colección de documentos. Es decir, si el conjunto de documentos o el contenido de los mismos cambia constantemente entonces podría resultar difícil mantener actualizado el índice y en este caso la única opción sería búsqueda en línea o secuencial. En esta sección revisaremos las técnicas que utilizan indexación, puesto que los archivos de audio por lo general no varían

en el tiempo, un archivo de audio cambiado se considera como un nuevo archivo de audio.

Una estructura utilizada para realizar búsquedas por indexación es el uso de índice invertido o archivos invertidos. Un índice invertido es aquel que invierte alguna relación existente en un ítem de información o varios ítems de información. El caso ilustrado en la Fig. 7 muestra la indexación de un texto, en el cual cada posición a nivel de caracteres tiene asociada una palabra y a continuación se invierte la relación al tener cada palabra asociadas una o más posiciones a nivel de caracteres. Esta nueva estructura permite obtener todas las ocurrencias de una palabra en un documento de manera rápida (20). Nótese además que no se conserva mucha información original como el contexto de cada palabra y tampoco se indexan todas las palabras.

Este tipo de indexación también se usa para invertir la relación términos relevantes para un documento dado con su contraparte que sería documentos relevantes asociados a un término dado. En este caso esta relación tampoco se trata de una biyección, por tanto la relación resultante no es en realidad inversa en el sentido matemático, sin embargo permite realizar búsquedas de manera

eficiente sobre el conjunto original de datos. Para realizar la indexación de documentos se pueden tener en cuenta todas las palabras presentes en la representación del documento o solamente las palabras consideradas como palabras clave del documento. Si se utiliza el enfoque de usar todas las palabras presentes en el documento el tamaño de la estructura de búsqueda se vuelve de mayor volumen, por lo cual no se suelen indexar palabras muy utilizadas en el lenguaje (artículos, pronombres, conjunciones, entre otras). Otra forma de reducir la cantidad de entradas en el índice es reducir las palabras a su forma básica, por ejemplo, nótese en la Fig. 8 que las palabras “nación” y “naciones” corresponden a una única entrada, de la misma forma se produciría con diferentes conjugaciones de un mismo verbo.

En (21) se describen otras estructuras para realizar búsquedas rápidas en el proceso de indexación como arreglos de sufijos y firmas de archivos.

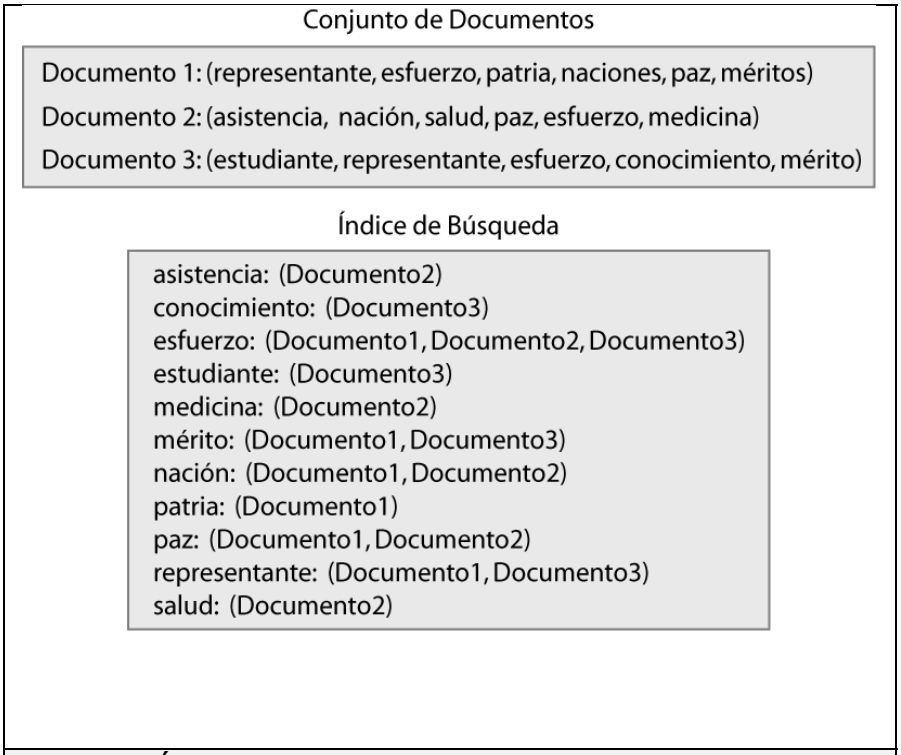

Figura 8. Índice que relaciona una lista de términos con listas de documentos relevantes, generado a partir de la lista original de documentos y su representación.

2.3.2 Cálculo de relevancia de documentos

En el modelo Booleano o en los sistemas de recuperación de datos tradicionales no existe la noción de relevancia de documentos, los resultados devueltos con estos métodos son conjuntos sin orden. Sin embargo en el modelo algebraico vectorial se puede hacer mediciones aproximadas de la distancia entre lo que el usuario desea y cada uno de los documentos. Esta aproximación se realiza mediante una aproximación entre la consulta q_i del usuario y la

representación de cada documento d_j del repositorio de documentos de búsqueda. Si un documento se aproxima más a la consulta q_i realizada por el usuario se dice que el documento representado por d_j es más relevante para esa consulta q_i. En los últimos años el método vectorial empírico ha sido racionado y extendido mediante modelos probabilísticos.

Existen dos medidas básicas para cuantificar la relevancia de un documento respecto a una consulta, la frecuencia de términos TF (*Term Frecuency*) y el discriminante de los términos con respecto a los documentos IDF (*Inverse Document Frecuency)*. Frecuentemente se utiliza una combinación de ambos (TF-IDF) para asignar los pesos a cada término con respecto a cada documento.

La frecuencia de un término para un documento se basa en el número de ocurrencias del término en el mismo. Este número de ocurrencias es normalizado para no depender de las variaciones en los tamaños de los documentos.
Sea *n(d,t)* el número de veces que aparece el término *t* en el documento *d.*

Normalización usando el número total de términos en el documento

$$TF(d,t) = \frac{n(d,t)}{\sum_{\tau} n(d,\tau)}$$

Normalización utilizando como referencia el término con el mayor número de ocurrencias

$$TF(d,t) = \frac{n(d,t)}{max_{\tau}(d,\tau)}$$

Normalización utilizada por el sistema Cornell SMART (22).

$$TF(d,t) = \begin{cases} 0 & si\ n(d,t) = 0 \\ 1 + \log\big(1 + log\big(n(d,t)\big)\big) & otherwise \end{cases}$$

El factor discriminante de un término con respecto a un conjunto de documentos representa el número total de documentos en contraste con el número de documentos que son relevantes para ese término. De la misma manera este factor es escalado en una escala de 0 a 1 mediante una escala logarítmica.

Sea *D* el conjunto de documentos y |*D*| la cardinalidad del conjunto y D_t el subconjunto que contiene al término *t.*

Normalización utilizando una función logarítmica por el sistema SMART

$$IDF(t) = \log \frac{1 + |D|}{|D_t|}$$

El TF y el IDF son combinados para ser utilizados en el modelo vectorial mencionado en secciones anteriores.

$$d(t) = TF(d, t)\, IDF(t)$$

Esta medida es la magnitud del vector de representación del documento en el eje del término *t,* además también se representa vectorialmente a la consulta del usuario en el espacio de los términos. La relevancia del documento con una consulta está dada por la diferencia entre el vector de representación del documento y el vector de la consulta.

Para un espacio vectorial de n dimensiones de términos, el valor de similaridad se calcula utilizando el coseno del ángulo entre el vector q que representa a la consulta del usuario y el vector d_i que es el vector de representación del documento. Este valor está dado por el

producto punto entre ambos vectores dividido para el escalar producto de la magnitud de ambos vectores.

$$SIM(q, d_i) = \frac{\sum_j q_j d_{ij}}{\sqrt{\sum_j q_j^2}\sqrt{\sum_j d_{ij}^2}}$$

En un sistema de recuperación de información además de este cálculo de relevancia inicial se utilizan métodos probabilísticos basados en el comportamiento del usuario y que sirven como retroalimentación para el nuevo valor de relevancia de cada documento.

En el caso particular de la Web es muy útil la información de relación que ofrece la estructura de grafo que se recrea con el uso de hipervínculos. El análisis de estas relaciones y su incidencia en el cálculo de la relevancia se han utilizado en algunos algoritmos como *PageRank* y *Hits* (23). Particularmente *PageRank* parte de la idea que cada documento Web tiene un prestigio y ese prestigio se basa en la suma de los prestigios de los documentos Web que vinculan a ese sitio, este proceso es altamente recursivo.

2.3.3 Problemas comunes

La utilización por parte del usuario de consultas poco descriptivas de sus verdaderas necesidades de información. Esto representa un problema porque provoca que los resultados de la búsqueda no sean los esperados por el usuario. Para tratar de resolver este problema se utilizan algunas técnicas como expansión de consultas basadas en diversos criterios como el contexto (24), como sería el conocimiento previo del sistema o el usuario. Si el usuario es un abogado o se conoce que su historial de búsquedas es acerca de asuntos legales y ejecuta una consulta “artículo cuarto Ecuador” el sistema conocerá que el término artículo en el área legal se relaciona con leyes o reglamentos y por tanto agregará está información a la consulta para generar resultados más relevantes para el usuario.

La representación de los documentos no refleja de manera óptima a la información contenida en el documento original. Muchos de los documentos en un sistema no son solamente texto plano sino que se encuentran en formatos diversos, inclusive en forma de audio y vídeo. Cuando la información está en forma de audio o vídeo se

necesita hacer un análisis más avanzado que involucra muchas otras tecnologías como procesamiento de audio e imágenes.

La falta de modelos de evaluación robustos, pues la relevancia de un documento basado en una consulta también depende de la percepción de cada usuario individual y de cómo cada uno expresa su necesidad de información como una consulta.

2.3.4 Perspectivas a futuro

Se adoptará el uso de tecnologías de procesamiento de lenguaje natural para generar modelos que representen de una mejor forma los contenidos de cada ítem de información para proveer resultados puntuales a consultas realizadas por el usuario utilizando lenguaje natural.

Se desarrollarán modelos de evaluación para sistemas de recuperación de información que permitan determinar con más precisión el nivel de mejoría de distintos algoritmos utilizados en la parte de cálculo de relevancia de documentos y que involucren al usuario.

Desarrollar técnicas de procesamiento y almacenamiento distribuido más efectivas en el área de recuperación de información para mejorar tiempos de respuesta y reducir el espacio físico que ocupa generar índices al nivel de la Web.

2.4 Caracterización de un sistema de recuperación de audio

Un sistema de recuperación de audio incorpora las tecnologías de procesamiento de audio en sistemas de recuperación de información, específicamente en el componente de extracción de información o de generación automática de metadatos. El proceso de recuperación de información además de incorporar estas tecnologías debe conocer algunos detalles al momento de implementar el extractor de información pues debe lidiar con la imprecisión de las técnicas de procesamiento de audio y que afecten en lo menos posible el desenvolvimiento general del sistema.

Un sistema de recuperación de audio restringido para el reconocimiento de voz hablada integra las tecnologías de reconocimiento del habla en el extractor de información. Este tipo de información de voz por lo general son documentos hablados con

mayor información que otros tipos de documentos de audio como música o grabaciones de sonidos arbitrarios.

Resumen

En este capítulo se describieron las tecnologías que serán usadas para implementar el sistema de recuperación de audio con información hablada. En primera instancia se analizaron de forma generalizada los sistemas de reconocimiento del habla con énfasis en las tecnologías que trabajan con habla continua y de vocabulario extenso. En el mismo apartado se analizó la arquitectura y la implementación del sistema de reconocimiento de voz Sphinx-4 mostrando así el estado del arte de las tecnologías de reconocimiento del habla en cada una de sus etapas de procesamiento.

En la segunda parte se analizaron los sistemas de recuperación de manera general y se mostró la estructura de archivos invertidos para realizar una indexación que permita recuperar ítems de información de manera eficiente. Se mostraron también varios modelos y enfoques para trabajar con recuperación de información y técnicas de relevancia para documentos respecto a las consultas generadas por el usuario.

Finalmente se describe a los sistemas de recuperación de información en audio y su relación con el modelo global de sistemas de recuperación de información.

CAPÍTULO 3

3 ANÁLISIS Y DISEÑO

3.1 Análisis de la solución

El sistema será capaz de resolver una operación principal que consiste en realizar búsquedas de acuerdo a consultas realizadas por el usuario al sistema. El sistema utilizará como fuente de ítems de información un repositorio externo al mismo. Entre otras operaciones especializadas del sistema estarán aquellas encargadas de mantener el índice como agregar o quitar ítems de información del índice del sistema y funciones de más bajo nivel para decodificación de archivos multimedia.

El planteamiento del problema establece la existencia de un almacén o repositorio de documentos con contenido de audio. Este repositorio

podría ser un directorio en un sistema operativo, un directorio virtual accesible por red, o un repositorio lógico definido mediante estándares de publicación y sindicación de contenido. La idea principal es mantener este repositorio de manera externa e independiente.

El sistema se encargará de contactar con el repositorio de documentos de audio y luego extraerá la mayor cantidad de metadatos presentes en el contexto del documento, como: el nombre del documento, el idioma, el tamaño, las palabras clave, la descripción, el autor, etc. Este contexto, en muchas ocasiones, no está presente o no aporta con mucha información para describir el documento, por lo que se requiere hacer un análisis más profundo sobre los contenidos binarios. Al tratarse con datos en audio, se requiere un análisis más avanzado de sus características para estar en la capacidad de extraer de forma automática información del documento.

Para esto último se han hecho muchos esfuerzos aproximadamente desde los años 50 en el área de procesamiento del habla o procesamiento de voz (25). El estado del arte de este tipo de sistemas ha probado moderado éxito con el reconocimiento de habla

continua, con vocabularios extensos, con algún tipo de tolerancia al ruido, y con independencia del locutor. Los resultados obtenidos con los sistemas de reconocimiento del habla más modernos, bajo condiciones controladas y con algunas restricciones, como la calidad del transductor y el ruido han alcanzado porcentajes de acierto superiores al 90% (26). Así mismo se han hecho investigaciones para obtener otro tipo de información de la señal de audio, como detección de música y reconocimiento de tonos musicales, extracción de características de la música (27), detección automática de idioma hablado (28), entre otros.

Los documentos de audio con información hablada aportan con mucha cantidad de información y se usan para distribuir varios tipos de información como: transmisiones de noticieros, clases a distancia, conferencias, programas radiales, etc. Además de incorporar estas tecnologías se analizarán las consideraciones a tomar en cuenta en este tipo de sistemas y que complementan la arquitectura de un sistema de recuperación de información tradicional. Se han usado muchas de las recomendaciones y prácticas en el diseño de arquitecturas para sistemas de bases de datos multimediales (29).

3.1.1 Modelamiento del problema

El problema de hacer búsquedas se puede subdividir en cuatro problemas principales. El primero consiste en el acceso transparente al repositorio de documentos en audio y el modelo de agregación de contenido al sistema. Luego está la tarea de analizar cada documento en audio intentando obtener la mayor cantidad de información posible del mismo y hacer uso de las tecnologías de reconocimiento del habla para ayudar en este propósito. Seguidamente estaría el problema de utilizar esta información obtenida en el paso anterior para agregarla a una estructura de archivo invertido que permita realizar búsquedas de manera eficiente. Finalmente se necesita un modelo de consultas y de acceso al sistema que permita acceder a sus servicios desde múltiples aplicaciones y plataformas, ver Fig. 9.

Por cada proceso se necesitará un componente o módulo que a su vez tendrá subcomponentes que realizarán tareas más específicas relacionadas a cada proceso. Cada componente de este sistema representaría una capa de independencia dentro de la arquitectura, de tal forma que se puedan desarrollar y mejorar de forma también independiente y sin afectar los componentes de las otras capas. Para

ello se establecerán acuerdos entre cada capa para que sirvan como protocolos y aseguren su interoperabilidad.

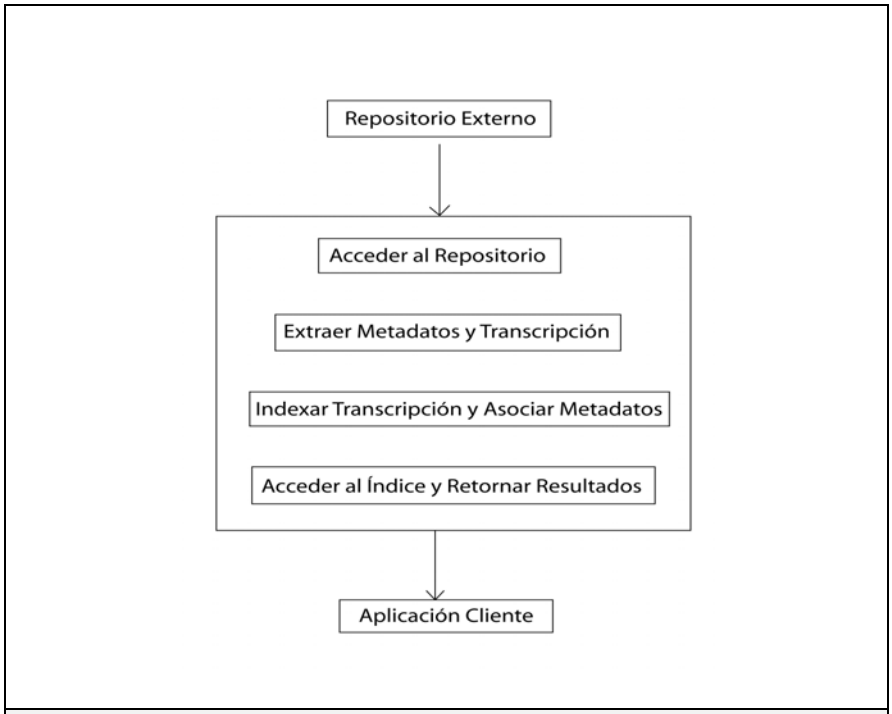

Figura 9. Flujo de procesos del sistema.

3.1.2 Casos de uso

Se había mencionado que la principal funcionalidad del sistema consistía en la consulta de ítems de información del repositorio externo al sistema, a partir de esta consideración se definen a continuación formalmente los actores y casos de uso.

Especificación de actores

Nombre: Cliente

Descripción: Realiza búsquedas sobre el repositorio a través del sistema

Notas: Podría en algún momento obtener privilegios para convertirse en un administrador y realizar mantenimiento del índice

Nombre: Administrador

Descripción: Es aquel que está a cargo de la administración del sistema, define la ubicación del repositorio o colabora con el mantenimiento del índice

Notas: Este actor podría proveer acceso a otros sobre el mantenimiento del sistema

Nombre: Repositorio

Descripción: Es el lugar de donde se extraerán los ítems de información en formato de audio

Notas: Es definido en el momento de la configuración o por el administrador del sistema

Especificación de casos de uso

Nombre: Realizar consulta de información

Descripción: El usuario envía una representación de sus necesidades de información al sistema a través de una consulta y el sistema debe devolver un listado de documentos que considere relevantes para esa consulta y ordenados por este criterio de relevancia.

Notas: Además de mostrar el listado el sistema deberá proveer un vínculo para que el usuario acceda a cada ítem de información.

Nombre: Definir repositorio de suscripción

Descripción: El usuario autorizado envía un enlace al recurso de sindicación de donde se obtendrá nueva información de audio para ser analizada e indexada

Notas: El usuario autorizado se asume es el administrador

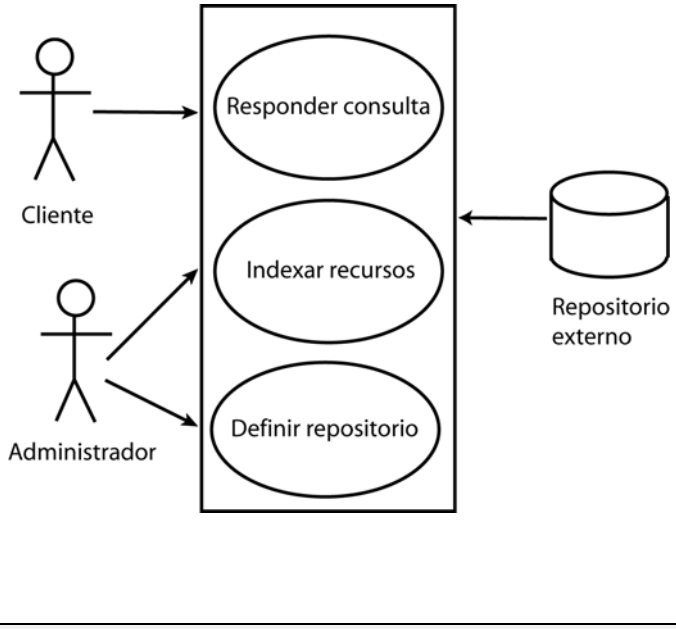

Figura 10. Diagrama de casos de uso del sistema.

3.1.3 Requerimientos funcionales

Entre los requerimientos funcionales del sistema están los siguientes:

- El sistema deberá ser capaz de ser altamente configurable y extensible para poder soportar distintos tipos de repositorios.

- Capacidad para decodificar el audio en el formato en el cual es distribuido en el repositorio y realizar un nuevo muestreo del

mismo en el caso que componentes de procesamiento de audio posteriores así lo requieran.

- Integración con un motor de reconocimiento automático de habla en la fase de extracción de información y metadatos.

- Capacidad de analizar la información extraída de forma automática para filtrar imprecisiones e información poco descriptiva.

- Integración con un marco de trabajo de desarrollo de sistemas de recuperación de información aplicando técnicas que permitan indexar la representación de los documentos y realizar búsquedas de manera eficiente.

- Exposición de servicios de consultas y servicios de agregación mediante el uso de tecnologías Web.

- Desarrollo de una interfaz accesible mediante un navegador Web que permita realizar búsquedas y tener acceso a los ítems del repositorio de información.

3.1.4 Requerimientos no funcionales

Algunos aspectos que deberán ser considerados durante la implementación del sistema y su uso posterior:

- El sistema debe ser escalable, se debe tener una arquitectura que permita cambiar componentes dentro del sistema sin afectar a otros. Esto debido a que se están usando diversos componentes con varias tecnologías que evolucionan a su propio ritmo, en este caso las tecnologías de reconocimiento automático del habla y las tecnologías de recuperación de información. Además debe asegurar que pueda extenderse para obtener escalabilidad de carga cuando se requiera procesar cantidades de información más grandes o manejar una cantidad de usuarios voluminosa.

- El sistema debe ser adaptable, algunas características de configuración podrán ser ajustadas para un funcionamiento más personalizado de la aplicación o para situaciones específicas.

- El sistema será independiente del motor de reconocimiento de voz que esté siendo usado, dado que las tecnologías de reconocimiento de voz tienen costos elevados, es necesario que este componente pueda ser actualizado dependiendo de las necesidades del usuario. Un usuario podría necesitar indexar las grabaciones de su blog personal o podría necesitar indexar grabaciones en audio de varios autores y con distintos niveles de ruido, en cuyo caso necesitaría un motor de reconocimiento de voz más robusto.

- El cliente Web deberá ajustarse a los estándares de la interfaz de los motores de búsqueda en la Web modernos y los sistemas de recuperación de información multimedia actuales.

3.2 Diseño de la solución

3.2.1 Modelo general

Existen muchas tecnologías y técnicas que pueden ser implementadas en cada uno de los componentes del sistema que se muestran en la Fig. 11. Se hace necesaria una adecuada selección del diseño generar que permita fácilmente intercambiar distintas

implementaciones de los componentes. Para lograr este objetivo se usará el patrón de diseño *Reflection*, con lo cual se podrán cargar las clases concretas que implementen las interfaces que representan a cada componente en tiempo de ejecución.

Aunque se ha solucionado parcialmente el problema con el uso del patrón *Reflection,* aun queda pendiente el problema de la dependencia que existirá entre las distintas implementaciones de cada componente. Cada uno no es un ente independiente sino que actúan o bien de manera colaborativa o de manera secuencial. Este problema requiere el uso de inyección de dependencia, que es una técnica de inversión de control comúnmente usada para eliminar las dependencias entre clases concretas. Existen muchos marcos de trabajo que contienen componentes para inyectar dependencia, entre los más populares están Spring Framework, Pico Container y Apache Avalon. Sin embargo para el diseño de este sistema se consideró más conveniente el uso de un patrón de diseño *Service Locator*, dado que al igual que un inyector de dependencia puede solucionar el problema desde un enfoque distinto y más simple.

Un patrón *Service Locator* consiste en tener una clase que conozca las implementaciones concretas de cada servicio que el sistema

provee, de tal manera que cuando algún componente cualquiera desee acceder a algún servicio tiene que pedir el servicio a esta clase que funciona como *Service Locator*. Además el *Service Locator* debería usar reflexión para poder cargar en tiempo de ejecución las implementaciones concretas de todos los servicios del sistema constituyéndose con esto además en un cargador de clases del sistema.

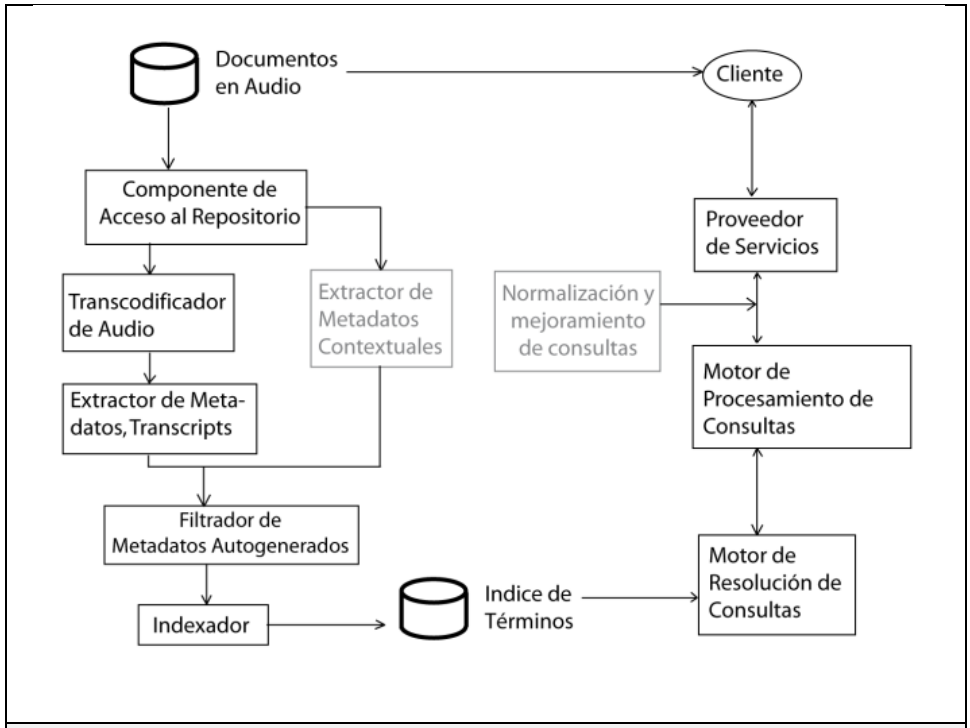

Figura 11. Flujo detallado de procesos en el sistema de recuperación de información de audio propuesto.

La principal diferencia entre usar un Inyector de Dependencia y un *Service Locator* consiste en que para el primero las llamadas a los

servicios son inyectadas por el contenedor de inversión de control proporcionado por el marco de trabajo mientras que con el segundo las llamadas a los servicios deben hacerse de forma explícita.

3.2.2 Modelo detallado

3.2.2.1 Componente de acceso al repositorio

Este componente es responsable de obtener los datos que están en el repositorio de archivos y disponerlos para el componente extractor de metadatos. Para esto necesita entender cómo se debe leer cada ítem de información presente en el repositorio. De esta forma se pueden implementar componentes que puedan leer documentos de audio desde varios tipos de repositorios como: directorios locales, directorios virtuales, recursos podcast. Este último proporciona más ventajas al ser un estándar basado en el Web que consiste publicación de archivos a los cuales es posible suscribirse y recibir actualizaciones con nuevos archivos cada cierto tiempo y ha sido utilizado ampliamente para distribuir contenidos en audio (30).

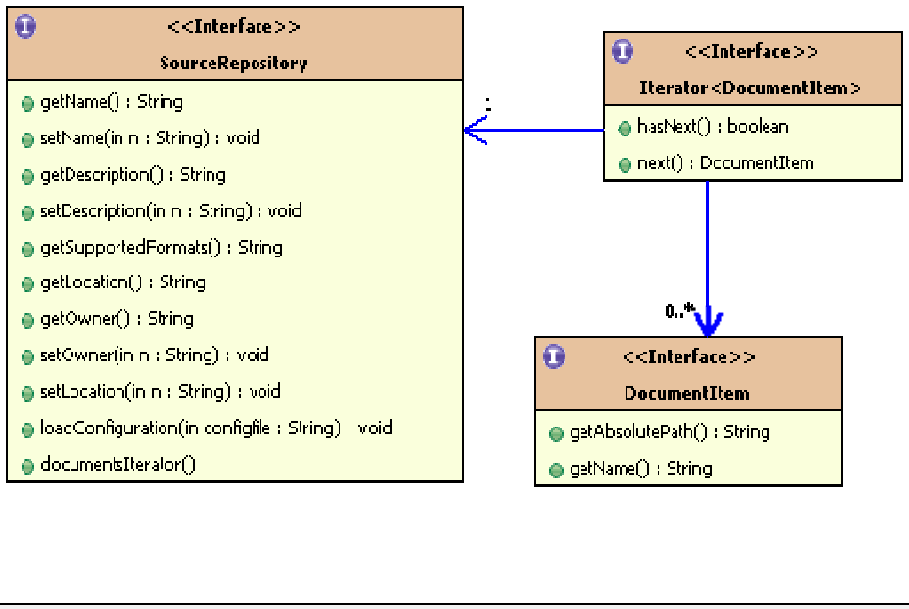

Figura 12. Diagrama de modelamiento de interfaces para la representación del repositorio de origen.

En la Fig. 12 se muestran las interfaces que deberán implementar los componentes encargados de acceder al repositorio y que servirán para que los componentes subsecuentes puedan tener un acceso tanto a los contenidos binarios como a los metadatos.

3.2.2.1 Componente transcodificador de audio

Este componente actuará dependiendo de las necesidades del componente de extracción de información y recibirá típicamente como entrada contenidos obtenidos desde el componente de acceso al repositorio. La principal función de este componente es

brindar un servicio de conversión entre formatos de almacenamiento y codificación de los contenidos de audio. Este servicio está representado mediante la interfaz mostrada en la Fig. 13.

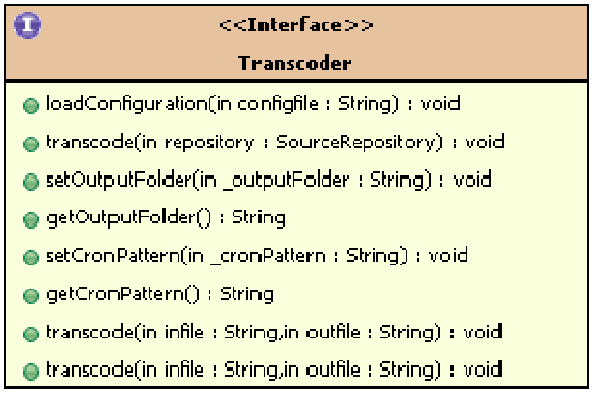

Figura 13. Interfaz *Transcoder* que deberán implementar los conversores entre distintos tipos de formato de audio.

Un transcodificador es necesario puesto que muchos algoritmos que implemente el componente de extracción de información podrían requerir que los datos binarios se encuentren en algún formato especial diferente al que soporta el repositorio de datos. Típicamente los documentos de audio se encuentran en formatos comprimidos dada la cantidad de información que poseen entre estos el de mayor uso es el formato *MPEG2 Capa 3*. Por otro lado

cuando se trata de procesamiento de señales se requiere usualmente que los datos estén en formato sin comprimir, por lo cual el transcodificador por lo general implementará o hará uso de decodificadores de audio.

3.2.2.2 Componente de extracción de información

Este componente es responsable de extraer la mayor cantidad posible de información que describa el contenido del audio y debe generar esta información en un formato entendible para el componente de indexación. En la extracción de metadatos existen dos subcomponentes principales que actuarán en conjunto para recabar la mayor cantidad de datos posibles que describan el documento de audio. Un subcomponente que extraiga los metadatos del contexto del ítem de información y otro subcomponente que extraiga los metadatos analizando los contenidos del audio (31). Existe además la necesidad de un tercer subcomponente principal que será el que encapsule el transcodificador de audio (32). Este componente será el encargado de entender el formato y el tipo de compresión del audio para poder decodificarlo y codificarlo nuevamente en el formato apropiado para ser usado por el motor de reconocimiento de voz. La Fig. 14

muestra el diagrama de como interactuarán estos tres subcomponentes.

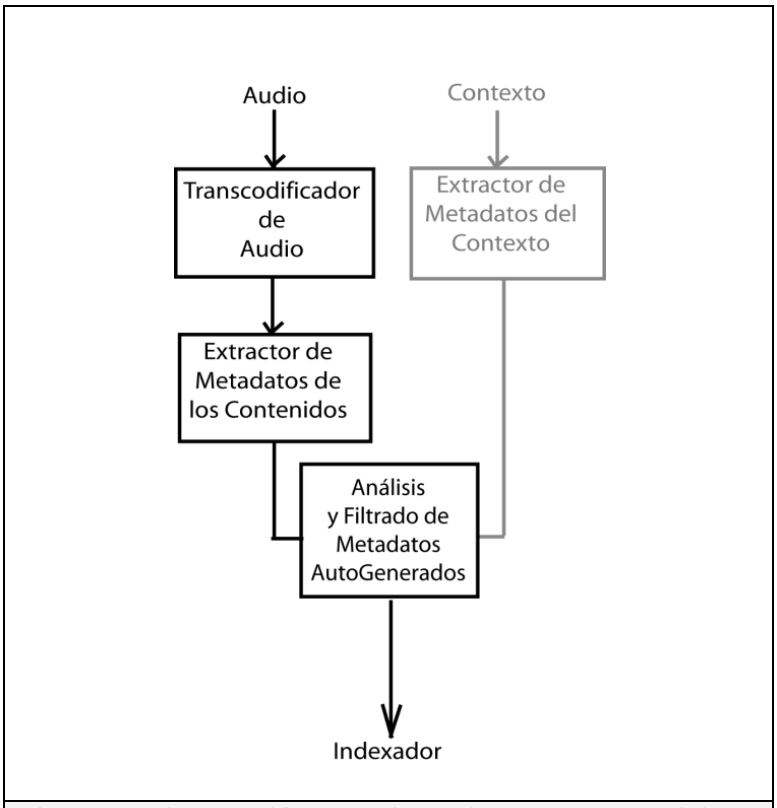

Figura 14. Interacción entre los subcomponentes del componente de extracción de información.

El subcomponente de extracción de metadatos de los contenidos de audio es el de mayor complejidad del sistema en cuanto a los requerimientos de procesamiento. Deberá soportar que se implementen nuevos algoritmos o técnicas de procesamiento para obtener mayor información del contenido de audio. Se ha recopilado varias técnicas que han sido usadas anteriormente en la

implementación de bases de datos multimedia y sistemas automáticos de recuperación de información de audio como *Speechbot*, *SpeechFind*, *FindSounds*, los mismos que han sido usados con propósitos investigativos y comerciales. Estos sistemas incorporan varias de las técnicas y algoritmos utilizados para extraer información de audio. Entre las técnicas que se han usado están: clusterización y segmentación del audio cuando existen múltiples locutores (33) (34), reconocimiento automático del habla continua, de vocabulario extenso e independiente del locutor (35), identificación automática del lenguaje (36), análisis de tonos, detección de sonidos por instrumento musical, detección de sonidos diversos (37) (animales, ruido, automóviles, aves, insectos), entre otras posibles implementaciones de extracción de información.

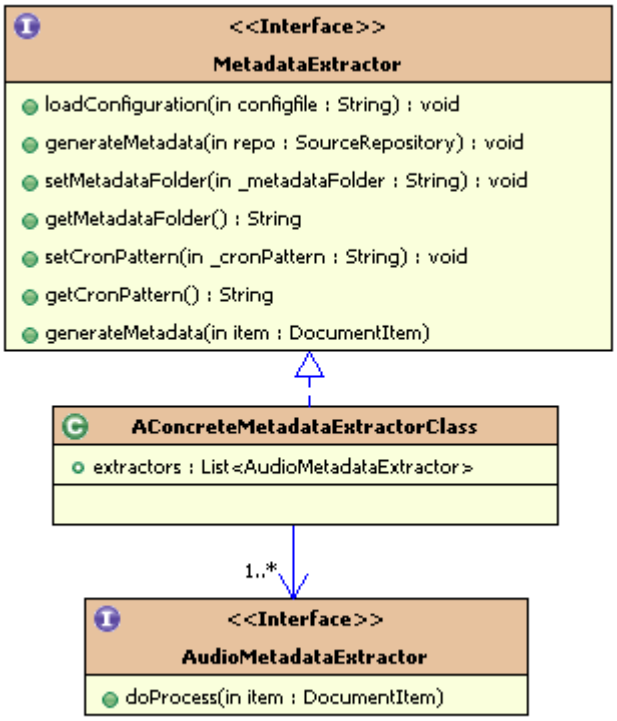

Figura 15. Interfaces que definen la interacción entre los subcomponentes del extractor de información.

Para realizar el envío de los metadatos generados y la transcripción aproximada del audio se utilizarán los metadatos encontrados en la descripción de un ítem *RSS* para cada recurso de audio cuyos metadatos hayan sido generados (38). Adicionalmente se obtendrá la transcripción generada automáticamente, sin importar el tipo de repositorio de donde se obtuvieron los documentos en audio.

Se pude considerar un subcomponente adicional que actuaría luego del extractor de metadatos de contenidos o extractor automático de transcripciones. Este componente se encargaría de aliviar la incidencia de la precisión del motor de reconocimiento de voz que se utiliza en el extractor de transcripciones. Entre las posibles opciones están: Filtrado de palabras, filtrado de palabras en base al contexto, técnicas de selección de palabras clave (39), o algún otro mecanismo que permita reducir la incidencia del error en las aproximaciones que generan los analizadores de contenido de audio. Para utilizar este enfoque resulta valiosa la información que se pueda obtener del contexto, por ejemplo: para reducir el diccionario de palabras clave que serán filtradas dentro de las transcripciones.

La Fig. 15 se muestra la interfaz *MetadataExtractor* que será la que deberá implementar el administrador de algoritmos de extracción de metadatos. La clase concreta que implementa *esta interfaz* será cargada en tiempo de ejecución por el localizador de servicios representado en el sistema por la clase *cargadora de los componentes.*

Además la Fig. 15 ilustra un patrón de diseño *Estrategia*, esto permitirá al sistema ser independiente de los algoritmos implementados en las clases concretas que implementan la interfaz *AudioMetadataExtractor*. Así la clase concreta que implementa *MetadataExtractor* será la que administre las clases concretas que implementan la interfaz *AudioMetadataExtractor* y sobreescriben la funcionalidad del método de procesamiento que analizará un ítem recuperado desde el objeto que representa al repositorio.

Modelo de eventos del generador de metadatos

Se ha modelado un patrón de diseño *Listener* a ser implementado de manera interna por el generador de metadatos (40). Este patrón servirá para hacer de interfaz con el motor de reconocimiento de voz que será integrado al sistema. Este modelo se ilustra en la Fig. 16 y muestra la interfaz del *Listener* y la clase que encapsulará la información obtenida en cada evento.

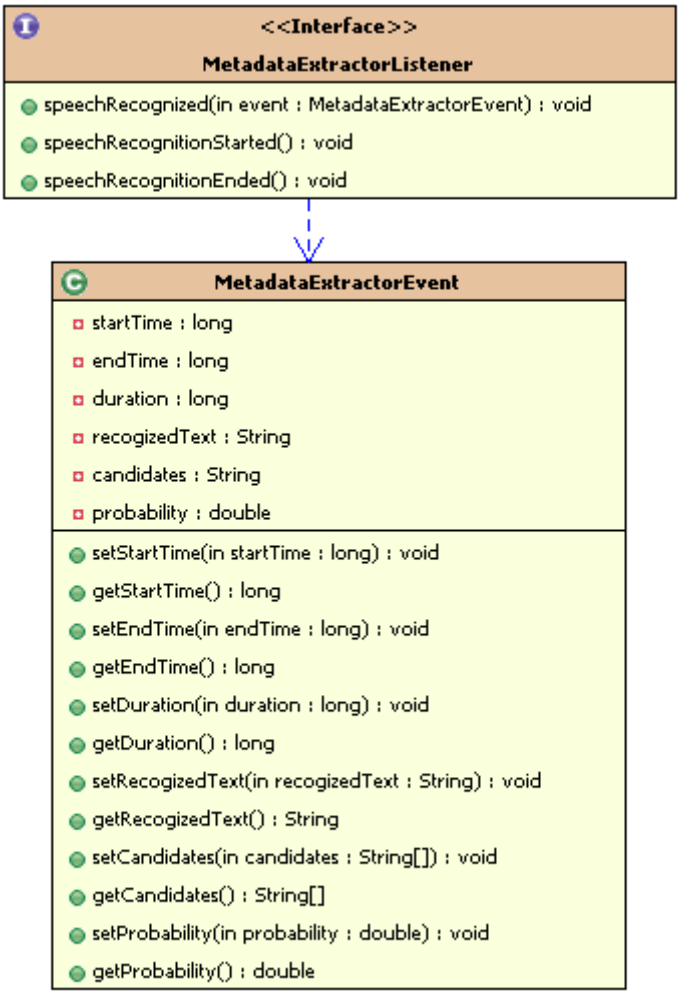

Figura 16. Modelo de eventos del componente de extracción de información, modelado de un evento para reconocimiento de texto de forma hablada.

Formato de descripción de metadatos de documentos de audio

Para cada documento de la colección se realiza el proceso de extracción de metadatos, siendo necesario persistir esta información para posteriormente realizar el proceso de indexación de metadatos en el índice de términos. El formato elegido para almacenar los datos es XML debido al soporte que posee este formato de descripción y a la cantidad de herramientas disponibles para manipular datos en este formato.

Se ha planteado un esquema de datos que sea capaz de almacenar conjuntamente la información del contexto y la información del contenido extraído en el proceso de reconocimiento de voz. La información del contexto que se almacenará está siguiendo la recomendación hecha en (41).

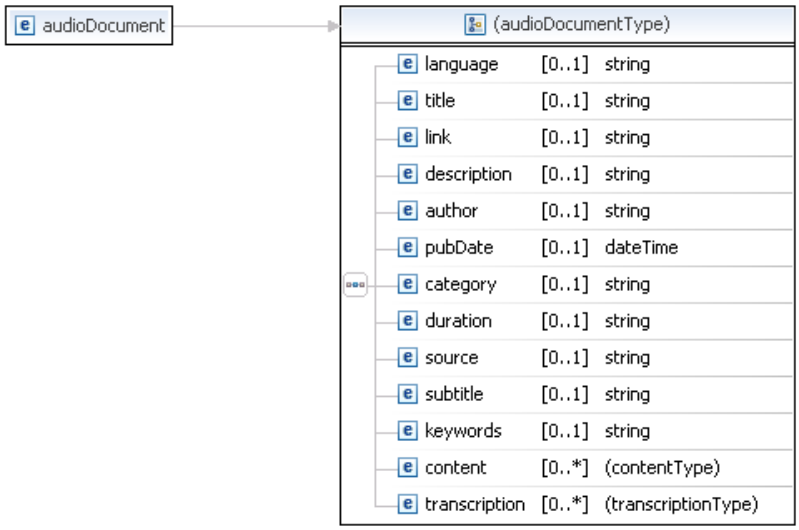

Figura 17. Descripción del esquema de metadatos asociados a un documento de audio en formato XML.

audioDocument/language

Código ISO del lenguaje hablado que contiene el documento de audio. Ejemplo: en,en-US,es,es-ES,es-EC,es-AR

audioDocument/title

Título del documento

audioDocument/link

Enlace mediante el cual se puede acceder al documento binario

audioDocument/description

Descripción del contenido de audio.

audioDocument/author

Nombres completos del autor, autores o entidad que reconoce la autoría del documento.

audioDocument/pubDate

Fecha de publicación del documento.

audioDocument/category

Categoría en la que se ha ubicado al componente

audioDocument/duration

Tiempo de duración del documento de audio en milisegundos

audioDocument/source

Fuente de donde se obtuvo el documento, pudiendo ser un sitio Web o el nombre del publicante.

audioDocument/subtitle

Subtítulo del documento

audioDocument/keywords

Palabras clave que son relevantes para el documento

audioDocument/content

Descripción del formato del contenido del documento, compresión y formato del archivo.

audioDocument/transcription

Transcripción generada usando reconocimiento de voz

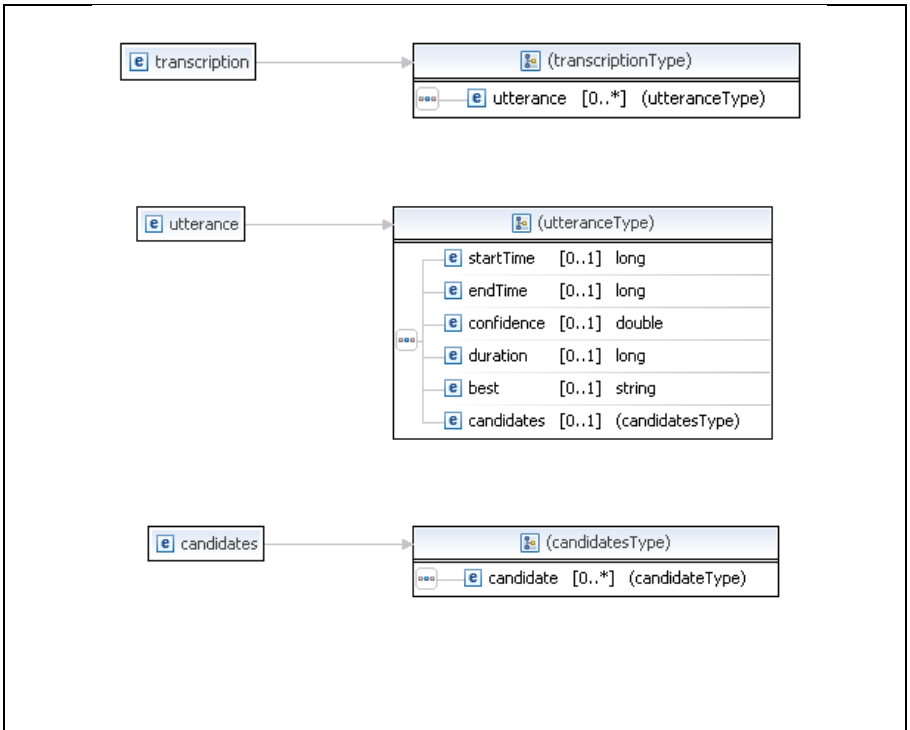

Figura 18. Esquema XML de una transcripción de un documento de audio.

audioDocument/transcription/utterance

Es la representación XML de una mínima unidad de contenido hablado

audioDocument/transcription/utterance/startTime

Tiempo en milisegundos en el cual empezó el *utterance.*

audioDocument/transcription/utterance/endTime

Tiempo en milisegundos en el cual terminó el *utterance.*

audioDocument/transcription/utterance/confidence

Es el grado de fiabilidad del mejor de los resultados obtenidos en el reconocimiento de texto del *utterance.*

audioDocument/transcription/utterance/duration

Es el tiempo en milisegundos que dura el *utterance.*

audioDocument/transcription/utterance/best

Es el texto reconocido en el *utterance* con mayor grado de fiabilidad.

audioDocument/transcription/utterance/candidates

Es el conjunto de variantes de texto reconocidas para el *utterance.*

audioDocument/transcription/utterance/candidates/candidate

Representa un texto que se consideró como posible transcripción del *utterance* durante el proceso de reconocimiento de voz.

3.2.2.3 Componente de indexación

Este componente es el encargado de crear la estructura de datos que será consultada para recuperar de manera eficiente los ítems de información que se encuentran en el repositorio externo. Una técnica básica para crear un índice de búsqueda es el uso de la técnica de índice invertido, mediante la cual se crea una lista ordenada de términos de búsqueda que tienen asociadas listas de apuntadores a ítems de información (42). Asociada a esta idea están muchas otras que tratan de resolver complejidades como el acceso más eficiente a esta estructura de índice o cómo reducir el tamaño de los índices de búsqueda. Muchas de estas técnicas y otras de mayor complejidad han sido implementadas en muchos sistemas y librerías en la actualidad.

Este componente indexador recibirá como entrada la descripción XML del documento de audio y guardará esta información como parte del índice, también utilizará la transcripción generada en el proceso de extracción de metadatos para realizar la indexación.

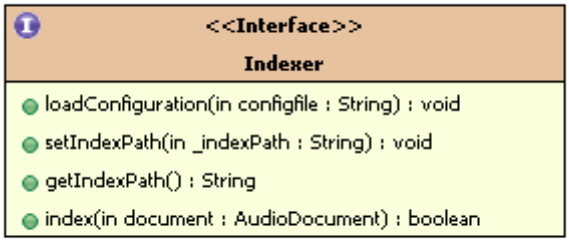

Figura 19. Interface que implementara el componente de indexación.

3.2.2.4 Componente de consultas y servicios

Es el encargado de retornar el listado de ítems de información correspondientes a una consulta determinada. Existen dos niveles de abstracción que actúan en este componente: El primero es el motor de procesamiento de consultas de bajo nivel, el cual se encarga de acceder directamente al índice generado por el

componente de indexación, analizar la consulta que se ha enviado para ser respondida y retornar los resultados. En el segundo nivel está un servicio que funciona como interfaz entre el motor de consultas y el cliente de la aplicación. Un subcomponente de este sistema sería un preprocesador de consultas que podría encargarse de algunas funcionalidades que han sido investigadas e implementadas en sistemas de recuperación de información como proveer normalización de consultas o expansión de las consultas (43) (44). Inclusive a este nivel puede considerarse la conversión de lenguajes de consultas mediante el uso de *parsers*. En la Fig. 20 se muestran las funcionalidades estarán delegadas a este componente.

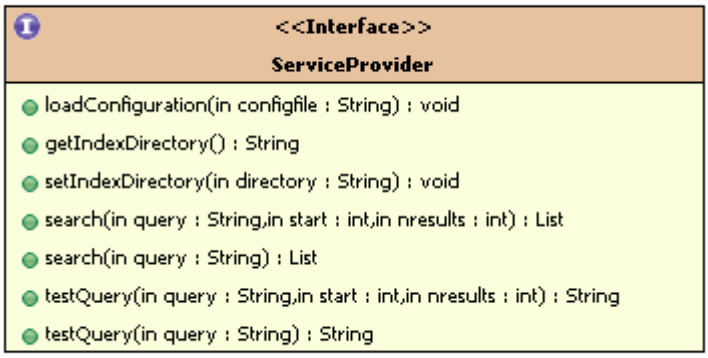

Figura 20. Interface que implementara el componente de servicios.

3.2.3 Modelo por capas

Hasta ahora se han definido las interfaces que permitirán encapsular cada una de las funcionalidades del sistema. En la Fig. 21 se muestran las principales interacciones entre los componentes que definen estas interfaces. A continuación se procede a establecer un modelo dividido por capas de abstracción.

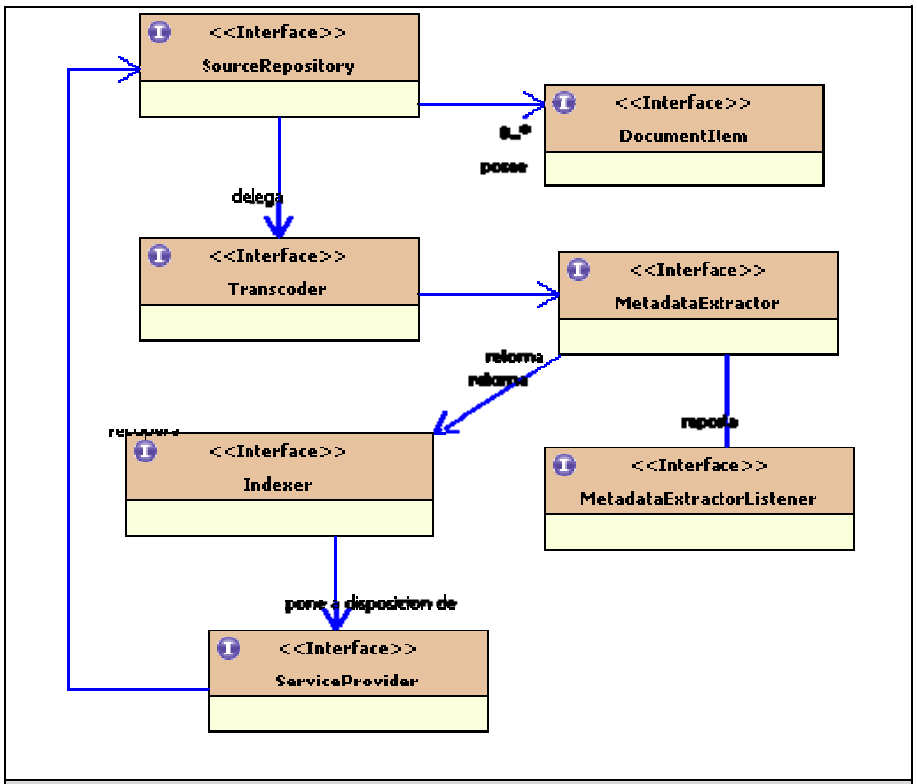

Figura 21. Interacción entre los componentes más generales del sistema.

Se han definido cuatro capas de procesamiento de información, cada una con componentes que efectúan tareas más específicas. A

continuación se mencionan las capas desde la que provee servicios de más bajo nivel hasta la que provee servicios de más alto nivel.

- Capa de acceso al repositorio:
    - Repositorio de documentos de audio
    - Componente de acceso al repositorio
- Capa de extracción de metadatos
    - Transcodificador de audio
    - Extractor de metadatos del contexto
    - Extractor de metadatos del contenido
- Capa de indexación
    - Componente de indexación
    - Índice de términos
- Capa proveedora de servicios
    - Motor de consultas
    - Preprocesador de consultas
    - Componente proveedor de servicios

En la Fig. 22 se muestra de manera resumida la arquitectura del sistema por cada capa y los componentes involucrados.

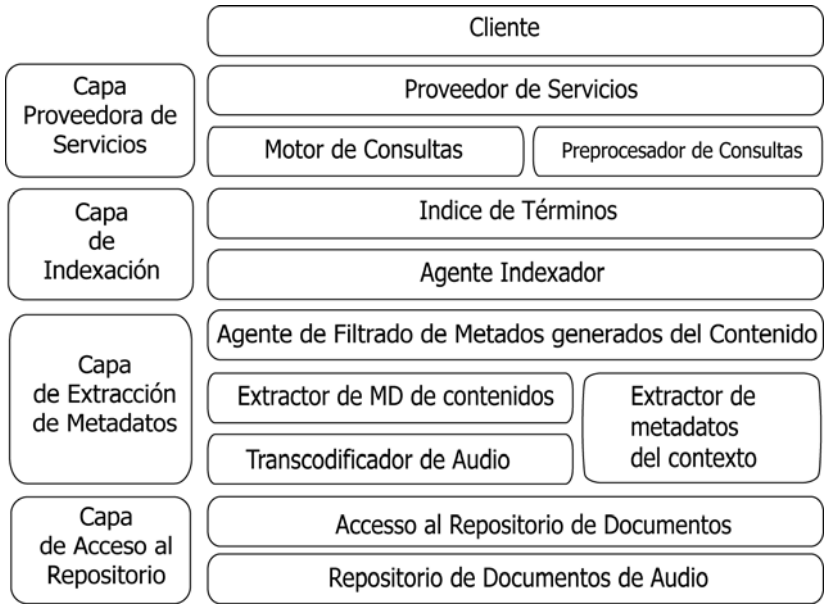

Figura 22. Modelo de eventos del componente de extracción de información, modelado de un evento para reconocimiento de texto de forma hablada.

3.3 Definición de métricas

Las métricas son las medidas de distintas propiedades del sistema que son evaluables o cuantificables, de tal forma que se pueda hacer un análisis y comparativa con otros similares.

3.3.1 Métricas de rendimiento

Miden el consumo de varios tipos de recursos del sistema ante distintos escenarios o estados. Entre estos recursos están: el tiempo, la capacidad de procesamiento, el consumo de memoria y espacio.

3.3.1.1 Métricas locales

Aquí se han puesto las medidas que están relacionadas directamente con cada componente individual del sistema.

Proveedor de servicios

- Tiempo que toma leer en el índice de búsqueda para responder una consulta.
- Tiempo que toma hacer *parsing* a la consulta.

Indexador

- Tiempo que toma indexar la información extraída de un documento.
- Tiempo que toma dar de baja un documento del índice.

Transcodificador

- Tiempo que toma transcodificar un documento de un formato comprimido a PCM.
- Tiempo que toma ajustar los parámetros de un archivo PCM de salida.

Extractor de metadatos

- Tiempo que demora obtener metadatos del contexto.
- Tiempo que demora analizar el contenido del audio.

Acceso al repositorio

- Tiempo que demora leer un archivo desde el repositorio.

3.3.1.2 Métricas de rendimiento sobre carga

Permiten conocer la escalabilidad del sistema ante cantidades de documentos extensas o condiciones extremas.

- Número aproximado de consultas que pueden resolverse simultáneamente.
- Tiempo de respuesta a consultas para un tamaño de índice conocido

- Número máximo de documentos que se pueden analizar simultáneamente

3.3.1.1 Métricas globales

Aquí se han listado las métricas que describen el desenvolvimiento general del sistema como un todo.

- Tiempo de respuesta a las consultas generadas por el cliente del sistema.
- Tiempo de latencia desde que un nuevo ítem se agrega al repositorio del sistema hasta que aparece en los resultados de las búsquedas.

3.3.2 Métricas de precisión y retentiva

3.3.2.1 A nivel global

Precisión: Esta medida cuantifica cuantos de los documentos que se devuelven para una consulta dada son relevantes para la misma. Para conocer esto usualmente se requiere conocimiento externo al

sistema que permita cuantificar esta relevancia, generalmente se recurre a los mismos usuarios para aproximar esta medida (45).

Retentiva: Esta medida cuantifica la proporción de documentos relevantes presentes en la respuesta a una consulta dada en comparación a la cantidad total de documentos relevantes presentes en el repositorio. Para repositorios con muchos documentos es difícil conocer este valor por lo cual se recurre a aproximaciones como las mostradas en (46) y en (47).

3.3.2.2 A nivel del motor de reconocimiento automático de voz

Los motores de reconocimiento de voz se miden generalmente por las siguientes dos características:

Tasa de Errores en Palabras (*Word Error Rate*, WER)
Representa la cantidad de palabras que fueron reconocidas por el sistema pero que no corresponden a la palabra en el documento de audio.

Tasa de Aciertos de Palabras (*Word Accuracy*, WA) Es el porcentaje de palabras que fueron reconocidas exitosamente con respecto al total de palabras.

Resumen

Se analizaron los requerimientos que el sistema debía cumplir para cumplir los objetivos planteados, se hizo un modelamiento del problema y se derivaron las funcionalidades generales del sistema. Además se presentó un análisis y discusión acerca de los componentes de software que forman parte de la arquitectura del sistema y se detallaron las responsabilidades de cada uno. Finalmente se plantearon las métricas de rendimiento, precisión y retentiva que serán usados como guías para las pruebas del sistema.

En el siguiente capítulo se hablará acerca de las implementaciones de cada componente descrito en el modelo presentado y las consideraciones técnicas a tener en cuenta.

CAPÍTULO 4

4 IMPLEMENTACIÓN

4.1 Selección de herramientas de desarrollo

Para seleccionar una herramienta de desarrollo adecuada se consideraron algunos criterios generales:

Facilidad de uso: El tiempo que toma a los desarrolladores a aprender a usar la herramienta de desarrollo de manera efectiva.

Funcionalidad: El conjunto de tareas que puede ejecutar la herramienta para resolver un problema.

Eficiencia: Tiene que ver con la calidad en que ejecuta la herramienta su funcionalidad en términos de recursos que utiliza, tiempo de ejecución y espacio.

Interoperabilidad: Es la capacidad de la herramienta para acoplarse a otras herramientas de distinta naturaleza.

Portabilidad: Tiene que ver con la facilidad de llevar la funcionalidad de la herramienta a otras plataformas de ejecución.

Flexibilidad: Es la capacidad de la herramienta para ajustarse a usos específicos o extenderse para ser usada en otras circunstancias.

Mantenibilidad: Se asocia a la facilidad para hacer cambios en la aplicación posteriores al despliegue de la misma.

4.1.1 Tecnologías de reconocimiento de voz

Entre las tecnologías de reconocimiento de voz están aquellas que permiten hacer reconocimiento de comandos o navegación a través de comandos y por otro lado están las herramientas para reconocimiento continuo de voz. Los distribuidores de motores de reconocimiento de voz han tratado de establecer alguna manera estándar para acceder a los servicios del motor que distribuyen.

Un estándar que ha tenido acogida por muchos distribuidores es *VoiceXML*. Éste es un lenguaje de marcas que no permite interactuar directamente con el motor de reconocimiento de voz y su uso está orientado a desarrollar interfaces de usuario por voz. Conjuntamente con *VoiceXML* se han hecho variantes combinándolo con XHTML, conocido como *VoiceXML+XHTML* para interfaces Web con soporte para entradas y salidas de voz. Microsoft conjuntamente con otros grupos propuso para efectos similares la especificación SALT (*Lenguaje de Marcas para Aplicaciones por Voz*).

Los estándares mencionados no permiten acceder a los desarrolladores directamente a los servicios del reconocedor de voz. Los distribuidores de estas tecnologías también han intentado ponerse de acuerdo respecto a una forma estándar de acceder a estos servicios sin llegar a un consenso. Existen tres especificaciones que se han creado para acceder a los servicios de un motor de reconocimiento de voz:

- JSAPI – Interfaz de Programación de Aplicaciones por Voz Java
- SAPI - Interfaz de Programación de Aplicaciones por Voz
- MRCP – Protocolo de Control de Recursos Multimedia

SAPI es una especificación desarrollada por Microsoft. Es conveniente debido a que Microsoft provee un motor de reconocimiento de libre distribución, además es soportado por motores de reconocimiento de otras empresas dedicadas a reconocimiento continuo de voz como Nuance, Philips e IBM. También permite hacer uso de las tecnologías de reconocimiento de voz integradas en sistemas operativos Windows. El lenguaje de desarrollo en su versión 5.1 es C++, pudiendo usarse desde otros lenguajes mediante objetos COM. En la versión 5.3 de SAPI se puso a disposición de los desarrolladores un API como parte de la plataforma *.NET*, con lo cual se puede programar aplicaciones que usen SAPI desde código administrado por el entorno de ejecución común de *.NET*.

JSAPI es una especificación desarrollada por Sun, de igual manera con soporte por parte de terceros y la programación se la realiza usando el lenguaje Java. Una de las ventajas de este API es la portabilidad del código entre plataformas, la misma que es derivada del entorno de programación Java. Aunque en este caso la aplicación en sí no es portable sino más bien el código que hace las llamadas al

motor de reconocimiento de voz, el cual sí está por lo general dependiendo de la plataforma.

MRCP es un estándar propuesto por la IETF (*Internet Engineering Task Force*), que es una organización dedicada a establecer estándares en Internet. MRCP no es un API sino un protocolo que permite acceder a los servicios del motor de reconocimiento de voz y servicios de síntesis de voz. MRCP tiene un comportamiento similar a otros protocolos como HTTP. Ha recibido algún soporte por compañías como IBM, Loquendo y LumenVox. Se han lanzado dos versiones de MRCP, la versión más reciente no posee compatibilidad hacia atrás con la anterior, lo cual ha hecho el desarrollo con MRCP más laborioso que usar un API específico de un proveedor.

La mayor parte de los sistemas de reconocimiento continuo de voz independiente del locutor es de tipo comercial y en algunos casos no se distribuyen como software para los usuarios finales sino como servicios. Desarrollar este tipo de tecnologías requiere mucho esfuerzo en investigación. Entre las causas están muchos factores mencionados en el capítulo 2, como el hecho de requerir de expertos en diversas áreas y la necesidad de recopilar grandes cantidades de datos de entrenamiento.

Entre las herramientas disponibles bajo licencias que permiten libre distribución y que se analizaron para tomar una decisión respecto al componente de reconocimiento del habla están las siguientes:

Motor de reconocimiento del habla de Microsoft

Se puede descargar gratuitamente para ser usado en el sistema operativo Windows XP y viene integrado en Windows Vista. Además implementa las interfaces de SAPI para ser usado por los desarrolladores. Implementa SAPI versión 5.1 en Windows XP y SAPI versión 5.3 en Windows Vista.

Ventajas

- Provee acceso a rutinas comunes de reconocimiento de voz independientemente del motor de reconocimiento de voz que implementa la interfaz.
- Microsoft provee SAPI conjuntamente con este motor de reconocimiento del habla continua en idioma inglés que es de libre distribución. Opcionalmente también provee la posibilidad de

obtener de forma gratuita motores de reconocimiento de voz para idioma japonés y chino simplificado.

- Es soportado por varias empresas distribuidoras de motores de reconocimiento continuo de voz, dado que SAPI se integra fácilmente con el sistema operativo Windows.
- El sistema operativo Windows Vista provee a los usuarios con motores de reconocimiento de voz en muchos más idiomas usando SAPI 5.3 y usualmente se instala conjuntamente con el sistema operativo.

Desventajas

- Su ejecución depende del sistema operativo Windows, es decir no hay implementaciones del entorno para otras plataformas.
- No es un sistema orientado a ser independiente del locutor, aunque puede ser usado sin entrenamiento con resultados ciertamente útiles bajo condiciones controladas.

Sphinx-4

Es un marco de trabajo de reconocimiento del habla que brinda además al menos una implementación para cada uno de los componentes e interfaces descritas por el marco de trabajo. Tiene como predecesores a los sistemas Sphinx-II y Sphinx-III que se desarrollaron en la Universidad de Carnegie-Mellon. El proyecto Sphinx-4 estuvo a cargo de desarrolladores e investigadores de la Universidad de Carnegie Mellon, Universidad de California Santa Cruz (UCSC), el Instituto Tecnológico de Massachusetts (MIT), el Laboratorio de Investigación de Mitsubishi (MERL), las compañías Sun Microsystems y Hewlett Packard (HP).

Ventajas

- Está escrito enteramente en el lenguaje de programación Java, por lo cual puede ejecutarse en diversas plataformas que implementen el entorno de ejecución Java.
- Además de ser extensamente configurable es de código abierto, por lo cual puede ser modificado y adaptado.

- Implementa un motor de reconocimiento de voz independiente del locutor.

Desventajas

- El consumo de recursos de procesador y memoria de la implementación del motor de reconocimiento Sphinx-4 hacen que el proceso de reconocimiento continuo de vocabulario extenso sea extremadamente costoso en cuanto al tiempo que toma realizar una transcripción.
- No existen más implementaciones del marco de trabajo Sphinx-4 aparte del que proveen sus creadores.
- No se acoge o implementa ningún estándar para procesamiento de contenido multimedia como MRCP
- No existen datos de entrenamiento para usar Sphinx-4 en muchos lenguajes, y la mayoría de los que existen están sujetos a licencias que no permiten la libre distribución.
- Su configuración y los pasos para adaptarlo a un uso específico no abstraen muchas de las complejidades del problema de reconocimiento de voz, por lo tanto se requiere un cierto nivel de conocimiento de estas tecnologías para usarlo. La documentación de Sphinx4 no está orientada a desarrolladores

sino más bien a investigadores en el área de reconocimiento del habla.

Análisis

La eficiencia en cuanto consumo de recursos de ambas herramientas es notoria, siendo el motor de reconocimiento incluido en la implementación base de SAPI 5.1 el que consumió menos recursos comparados con el motor de reconocimiento Sphinx-4, el cual demandó mayor tiempo y uso de recursos del computador, confirmando la investigación hecha por Ayres & Nolan (48) acerca de estos dos motores.

Se descartó el uso de Sphinx-4 para la elaboración del sistema debido al costo en recursos y la poca fiabilidad en ambientes no controlados, además su uso está restringido por las limitaciones de los datos de entrenamiento del motor. Aunque cabe señalar que con el desarrollo que está teniendo Sphinx-4 y su característica de ser independiente del locutor se proyecta como una herramienta que tendrá un potencial a medida que se obtengan más datos de entrenamiento y de mejor calidad.

El motor de reconocimiento a utilizarse para las pruebas del componente de referencia del sistema será el motor de reconocimiento del habla de Microsoft en su versión 7 (Windows XP) y versión 8 (Windows Vista). Las pruebas demostraron una ejecución eficiente, aunque también un nivel de errores alto, por lo cual se requerirá utilizar algún criterio posterior para minimizar la incidencia de este error en las búsquedas del sistema final.

4.1.2 Marcos de trabajo de recuperación de información

Entre las principales características que poseen está su capacidad para crear índices de manera eficiente, la posibilidad de realizar consultas utilizando un lenguaje de consultas flexible y la posibilidad utilizar diversos analizadores de contenidos.

Apache Lucene

Es la especificación de un motor de búsqueda de texto con implementaciones en varios lenguajes de programación y entornos de ejecución, siendo la versión Java la que más atención tiene en la comunidad de desarrollo.

Ventajas

- Al estar implementado en Java, es soportado por cualquier plataforma con un entorno de ejecución Java
- Provee soportes para análisis de texto en muchos idiomas, y varios analizadores de texto más especializados.
- Posee facilidad de uso para el programador, es fácil familiarizarse con el uso de las interfaces y los componentes.
- Es extensible permitiendo escribir nuevos analizadores e indexadores de manera fácil.
- Existen implementaciones en otros lenguajes de programación que conservan el formato del índice, pudiendo migrarse fácilmente los datos del mismo.

Sphinx (el motor de búsqueda de texto)

Ventajas

- Provee soporte para almacenar el índice en una base de datos MySQL.

- El código es multiplataforma, es decir puede ser compilado para ejecutarse en variadas plataformas. Se pueden conseguir binarios para las plataformas más utilizadas.

Análisis

Ambas herramientas han probado ser capaces de trabajar con grandes cantidades de datos (49). Sphinx al ser una implementación escrita en código nativo en cada plataforma tiende a tener ventaja en cuanto a rapidez de ejecución. Lucene posee más funcionalidades para modificar el índice en tiempo de ejecución y tiene una mejor integración con la plataforma Java.

Se ha elegido utilizar Apache Lucene para la elaboración del sistema porque aparte de las pruebas de rendimiento de ambos motores de búsqueda está la facilidad de uso, extensibilidad, el gran soporte de la comunidad y la integración con la plataforma Java.

4.1.3 Middlewares para exposición de servicios

Un middleware es una pieza de software que provee servicios de comunicación entre componentes de una aplicación, un servicio

importante es brindar abstracción en la comunicación. La abstracción permite esconder detalles de protocolos de comunicaciones y formatos de intercambio de datos. En el presente trabajo se ha decidido usar servicios Web para tal propósito, los servicios Web se montan sobre los protocolos de transmisión de Internet y sobre HTTP.

Existen varios tipos de servicios Web, siendo aquellos más aceptados como un estándar los servicios Web que utilizan el protocolo SOAP para el intercambio de mensajes. Aparte de SOAP están los servicios llamados *RESTful* que utilizan los diversos métodos de transmisión de HTTP para asociarlos a acciones específicas (GET, POST, PUT, DELETE). Otra opción son los servicios XML-RPC que fueron los predecesores de los servicios Web SOAP donde el intercambio de mensajes también se basa en llamadas y respuestas en formato XML, pero en una versión mucho más ligera.

Se decidió usar Apache Axis2 como la capa de middleware para exponer los servicios Web del sistema debido a que es un proyecto que se está desarrollando continuamente y tiene gran soporte de la comunidad y la industria.

Apache Axis2 presenta las siguientes ventajas respecto a versiones anteriores de Axis y otros middlewares de exposición de servicios Web SOAP como XFire.

- A diferencia de la versión original de axis fue pensado en un entorno de producción y no como una primera implementación base de middleware para servicios Web.
- Permite empaquetamiento de servicios Web para facilitar el despliegue de servicios en caliente o para migrar servicios Web a otra plataforma de manera eficiente.
- Se integra fácilmente con varios entornos de desarrollo, provee herramientas para generación de código cliente y servidor.

4.2 Desarrollo del sistema

4.2.1 Acceso al repositorio

Se implementó la interfaz *SourceRepository* para poder manejar un repositorio de documentos representado por un directorio local. Para ello se implementó además un iterador que retorne los documentos de manera que el componente que acceda al repositorio abstraiga el

hecho de que los documentos están siendo recuperados desde un directorio. Esta relación se muestra en el diagrama de la Fig. 23.

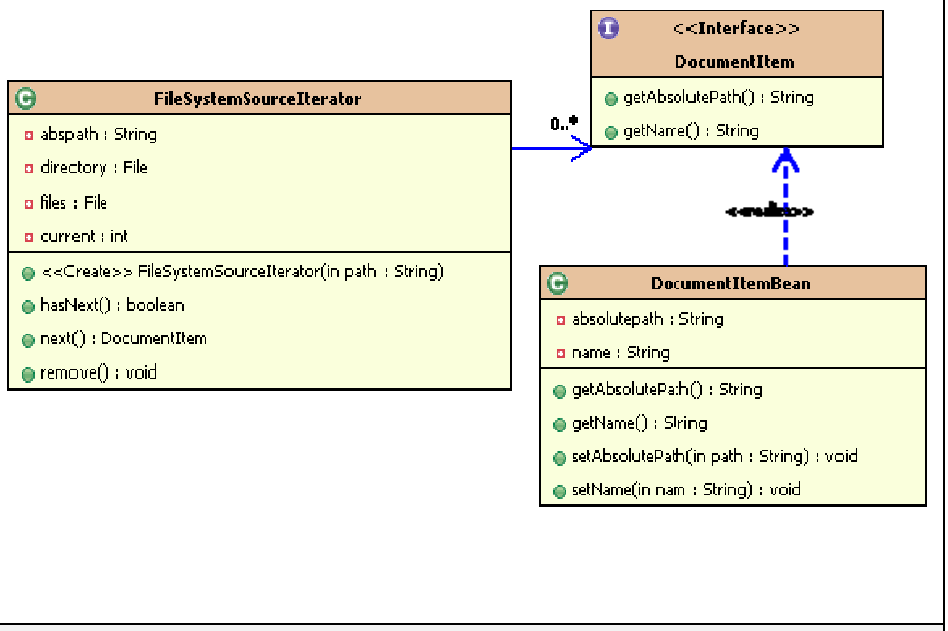

Figura 23. Implementación de un patrón de diseño Iterador para el acceso al repositorio de documentos

4.2.2 Transcodificación de audio

Este componente surge de la necesidad de cambiar la codificación del audio de entrada hacia un formato y codificación diferente al original. El propósito es disponer la información de audio en una forma que el extractor de información que utiliza los datos binarios del audio pueda obtener metadatos.

El formato de un archivo de audio se define por la estructura de almacenamiento que utiliza el archivo, las cabeceras del archivo, información de metadatos, y la información de la codificación que utiliza el contenido binario del archivo.

La codificación del audio se define por la manera en que se almacena la parte binaria del archivo. Podría almacenarse usando esquemas sin compresión como PCM (Codificación por Modulación de Pulsos) o LPC (Codificación Lineal Predictiva) o utilizar sofisticados codificadores para almacenar la información en formatos de compresión que se aprovechan de las características psicoacústicas del audio como MPEG2 Capa 3 o el formato Ogg Vorbis.

Generalmente el audio de entrada estará en un formato comprimido dado que un repositorio que almacenare documentos en audio sin comprimir tendría un tamaño notablemente grande. Guardar 1000 archivos de 1 hora de duración cada uno podría ocupar en espacio físico de disco dependiendo de las características del audio sin comprimir unos 620 Giga Bytes. Ver la Tabla III.

Longitud del Audio	Sonido Estéreo, 44KHz, 16 bits por muestra	Sonido Mono, 22KHz, 8 bits por muestra
1 segundo	172 KB	22 KB
1 minuto	10.3 MB	1.32 MB
5 minutos	51.6 MB	6.6 MB
30 minutos	309 MB	39.6 MB
1 hora	619 MB	79.2 MB

Tabla III. Comparación entre datos de audio comprimido y sin comprimir

En la implementación inicial del transcodificador de este sistema se usarán repositorios de archivos codificados en MPEG2 Capa 3 (MP3), soportando además los formatos de codificación sin comprimir.

Este transcodificador de audio de archivos en formato MP3 consta de dos fases. En la primera fase se lee el archivo original codificado usando MP3 y se lo transforma a su equivalente sin comprimir

codificado en PCM. En la segunda fase se transforma el archivo PCM obtenido y se lo procesa para ajustar sus parámetros al formato requerido para extraer información en fases subsiguientes. Entre los parámetros que se requiere modificar están el tamaño de cada muestra de audio, la frecuencia de muestreo y el número de canales.

Para desarrollar este componente se ha hecho uso de la librería *JavaLayer* y el proveedor de servicios para mp3 que implementa. Esta librería se acopla nativamente en la implementación base de Java para el manejo de archivos de sonido. Una ventaja de usar este método es que al usar las librerías estándar de Java, el código que se escribe es independiente del proveedor de servicios de codificación de archivos. Usando esta opción se puede de manera transparente y sin modificaciones dar soporte a más formatos de audio con compresión cuando los proveedores estén disponibles. Una desventaja de usar este método es que las librerías estándar de Java no se distribuyen con soporte para ningún formato que use compresión y los proveedores desarrollados por terceros son escasos y limitados.

Dada la flexibilidad de la arquitectura se ha considerado una segunda implementación para este componente que utiliza otro método para la transcodificación y el remuestreo. Este método consiste en usar el

software libre para reproducción y codificación de archivos *Mplayer* para la transcodificación y extracción del audio PCM. Además el software libre *Sox* para realizar el remuestreo del audio PCM para ajustarlo a los parámetros deseados por el sistema. La principal ventaja de usar este método es que soporta una amplia gama de formatos de audio con compresión como entrada, inclusive soportando la extracción de audio de documentos en video. Una desventaja de este enfoque es que la implementación de este componente se hace dependiente de programas externos al sistema y por lo tanto cada vez que se desee hacer una transcodificación se crearan nuevos procesos en el sistema operativo. Sin embargo dadas ambas implementaciones el sistema puede ser configurado para usar alguna de las dos en tiempo de ejecución. Registrando el proveedor de servicios de transcodificación en el cargador de clases del sistema.

4.2.3 Extracción de metadatos

El primer componente implementado de la arquitectura propuesta fue el administrador de los subcomponentes genéricos de extracción de información. Este componente se aprovecha del cargador de clases

para dinámicamente agregar los subcomponentes en tiempo de ejecución. También se generaron automáticamente a partir de los esquemas XML, usando la librería *Castor XML*, las clases que servirán de contenedores para los metadatos. Estas clases representarán a los documentos de audio y serán los objetos donde se guarde la información obtenida con los extractores de información genéricos.

Se implementaron dos extractores de información en el sistema:

- El extractor de información del contexto
- El extractor de información que usa reconocimiento de voz

Ambos extractores actúan de manera independiente para capturar la mayor cantidad de información del documento de audio.

El extractor de información del contexto, en el caso del sistema aquí presentado intenta obtener la mayor cantidad de información desde el contenido no binario del archivo de audio original. Los creadores de contenido multimedia no han llegado a un consenso acerca de la manera estándar a usarse para agregar información de metadatos a los archivos multimedia. En el caso del formato MP3 se utiliza un sistema de etiquetado de acuerdo al formato ID3. Este formato no es

propio de MP3 y es usado en otros formatos de distribución de archivos de audio. Una ventaja de usar la información de etiquetas ID3 es que el formato MP3 es bastante popular y por tanto muchos reproductores también usan la información de las etiquetas ID3 para mostrar información al usuario. Por esta razón suele colocarse información relevante al usuario en este formato. La forma de guardar metadatos para otros tipos de formatos es muy dependiente de cada uno. Aparte de MP3 existen otros formatos como AIFF, WMV, WMA, MP4 que también soportan de alguna manera ID3.

El componente de extracción de metadatos del contexto usa esta información de las etiquetas ID3 y la agrega como campos en el objeto que representa a los documentos de audio.

El extractor de metadatos que usa reconocimiento de voz hace uso además de los servicios del componente de transcodificación para poder analizar el contenido de audio. Una vez que invoca a los servicios de transcodificación procede a analizar el texto usando las interfaces SAPI. Para lograr hacer uso de SAPI, que es un conjunto de interfaces en C++ se ha recurrido a la utilización de las interfaces nativas de Java JNI (Interfaz Nativa de Java). La integración de SAPI con Java permite utilizar de forma no obstrusiva desde código Java

funcionalidades de cualquier motor que implemente la interfaz SAPI. Estas características hacen que este componente se constituye en potencialmente reusable por otros desarrolladores.

La integración con Java desde el punto de vista de arquitectura se presentó como un patrón de diseño *Observer* a través del uso de una clase de escucha (*Listener*). En la Fig. 24 se muestra de forma más detallada la interacción entre los componentes y se especifica además el ámbito del entorno de ejecución de cada objeto.

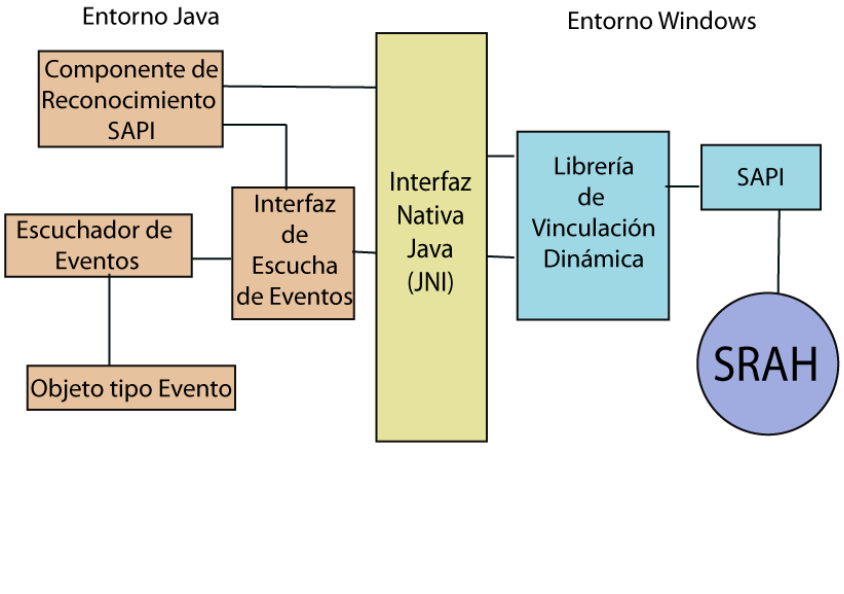

Figura 24. Interacción entre los componentes de reconocimiento de voz.

4.2.4 Indexación

Se usará el marco de trabajo Lucene para realizar la indexación en un directorio definido en el archivo de configuración global del sistema. En la Fig. 25 se muestra el diagrama de la implementación de la interfaz de indexación. El principal método que sobrescribe es el método *de* indexación que recibe como parámetro un objeto documento de audio que contiene todas las características recopiladas en fases anteriores de procesamiento.

El formato de datos del índice está determinado por el formato de datos de un índice de Lucene. Una ventaja derivada de esto es que existen otras implementaciones del marco de trabajo Lucene que también utilizan este formato. La complejidad de la operación de indexar un documento es independiente del tamaño del índice y del documento, de tal forma que permite lograr cierto nivel de escalabilidad.

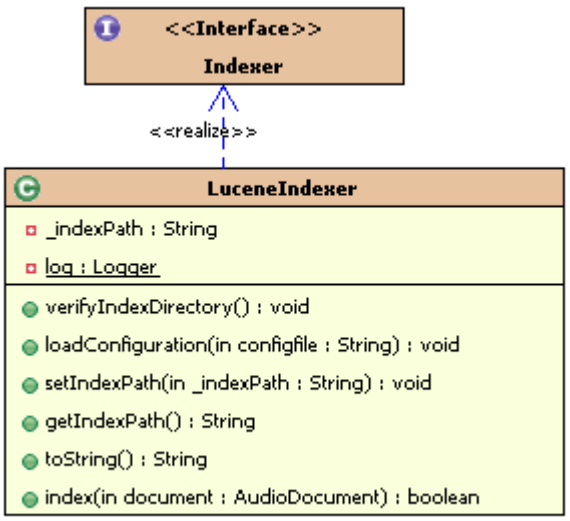

Figura 25. Implementación de la interfaz de indexación usando Lucene.

Al momento de indexar se debe tomar en cuenta la manera en que serán procesados los datos obtenidos en la extracción de información para ser agregados al índice. Existen algunos metadatos que se almacenarán conjuntamente con cada ítem dentro del repositorio, mientras que otros serán analizados y usados para construir el índice. Usando la herramienta de desarrollo Lucene se diferencian estos dos tipos de tratamiento de los datos mediante la definición de campos TOKENIZED y campos UN_TOKENIZED. Además se puede especificar si el campo será almacenado o solo se

usará para generar el índice. Si el campo se usa para generar el índice se usa el analizador sintáctico definido por defecto en el sistema que es el analizador estándar.

4.2.5 Servicios de consulta

Se han desarrollado en primera instancia los componentes que acceden directamente al índice del sistema utilizando el marco de trabajo Lucene. Así se tiene el método de bùsqueda para recuperar los resultados recibiendo como parámetros: la consulta en una cadena de caracteres y el rango de los resultados relevantes que se desean obtener con esa consulta. Se ha considerado usar un analizador de consultas estándar que es el que se utilizo para indexar el documento, de esta forma se consigue normalizar la consulta usando el mismo tipo de normalización que se aplicó al momento de la indexación.

Se ha definido además un contenedor para encapsular cada ítem devuelto por el buscador. El componente que accede directamente al índice retorna una colección de ítems del tipo *AudioDocumentItem* que representa cada elemento de los resultados de búsqueda.

Fueron desarrollados también otros subcomponentes para poder retornar resultados en formatos de transmisión de datos específicos. Entre estos está un componente que implementa una interfaz SOAP y un componente que retorna resultados en el formato JSON (Notación de Objetos JavaScript).

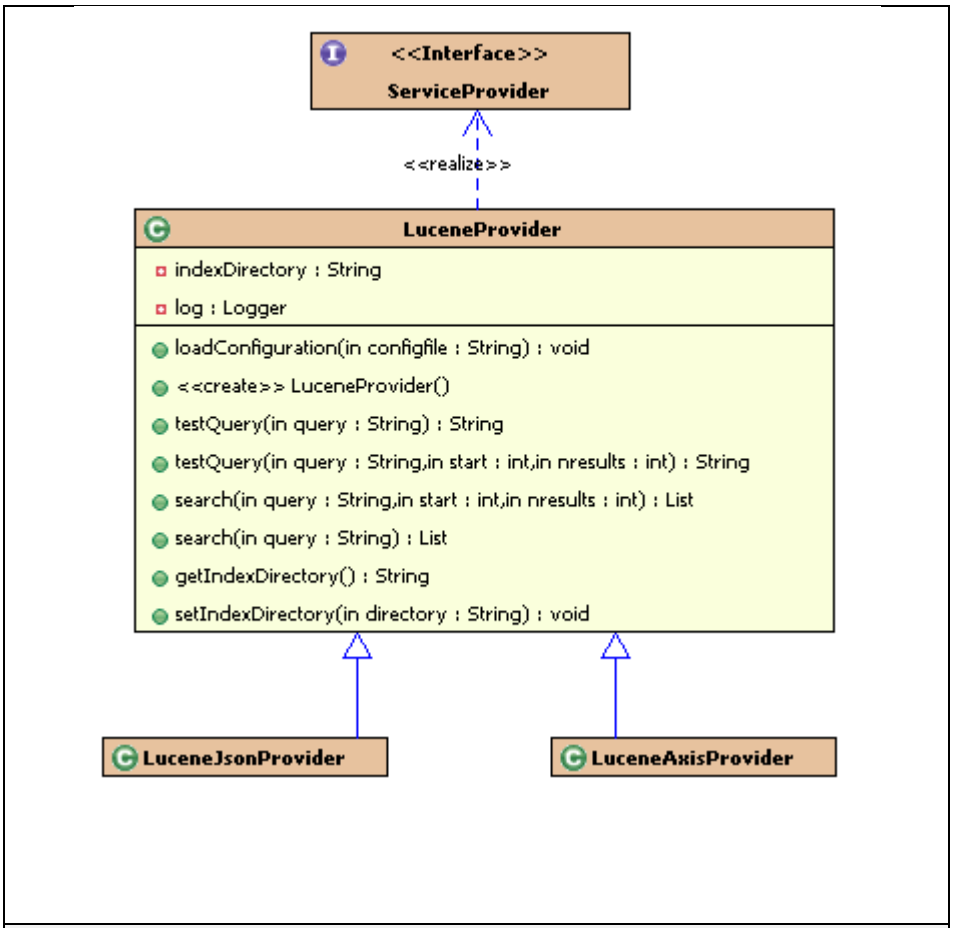

Figura 26. Implementación de la interfaz de indexación usando Lucene.

En la Fig. 26 se muestran los servicios que ofrece este componente. Los servicios de prueba de consultas, se utilizan para conocer si la consulta ha devuelto algún error que no ha podido recuperar los datos como un tipo de sintaxis de entrada no permitida o un error en el servicio. Además están los servicios de búsqueda de documentos y de conteo de resultados totales para una consulta dada.

4.3 Desarrollo del cliente Web

Usando los componentes aquí desarrollados se pueden crear fácilmente aplicaciones que posean la capacidad de indexar archivos de audio y recuperar archivos desde los índices para poder realizar búsquedas.

Los principales componentes que se usarán en el desarrollo del cliente serán el extractor de metadatos, el indexador para agregar nuevos archivos a través del Web y el componente de acceso a servicios para realizar búsquedas sobre el índice.

El cliente Web se ha desarrollado usando el Modelo 2 que es una adaptación del patrón modelo vista controlador (MVC) al entorno de

las aplicaciones Web. La arquitectura por capas del sistema y su integración con el sistema de búsqueda de audio se ilustra en la Fig. 27.

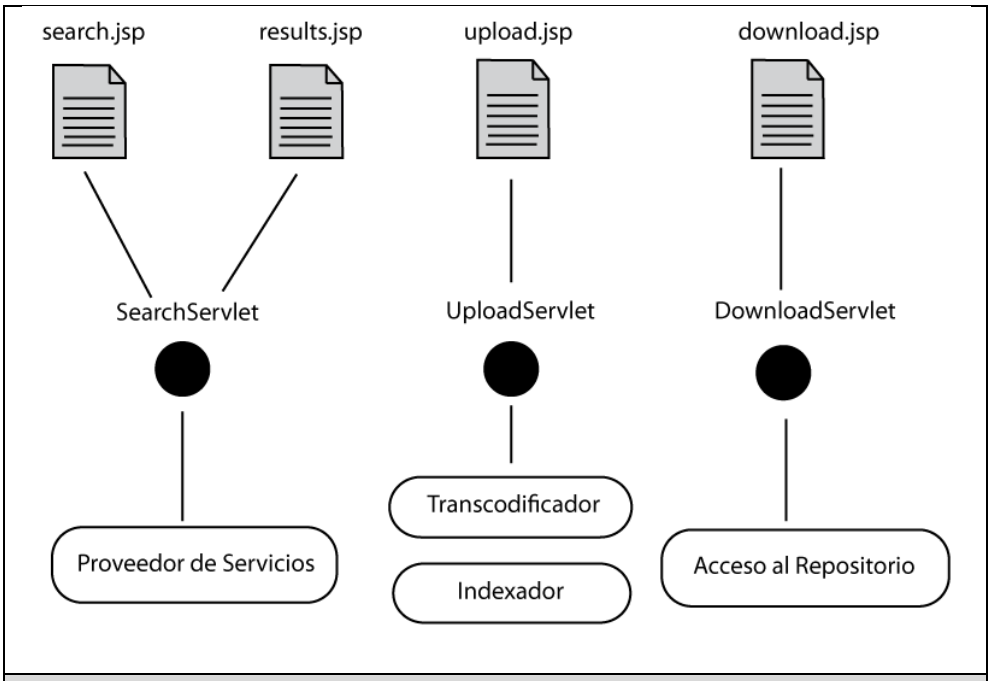

Figura 27. Arquitectura de 3 capas del cliente Web.

4.4 Instalación y configuración del sistema

El sistema hace uso de la plataforma Java, por lo cual la mayor parte de los componentes pueden escribirse de manera independiente de la plataforma. El transcodificador puramente

Java permite ejecutar este componente en distintas plataformas, sin embargo la implementación nativa hace uso de los códecs del programa externo mplayer. Mplayer está disponible para distintas plataformas, por lo cual su uso no restringe mayormente la portabilidad del sistema. El componente de extracción de metadatos hace uso de la interfaz SAPI para reconocimiento automático de voz, por lo cual restringe su ejecución a plataformas que implementen esta interfaz. SAPI es implementado en la mayoría de los sistemas Windows, sin embargo aún no existen implementaciones para otras plataformas.

El cliente Web del sistema se ejecuta sobre un contenedor de servlets que implemente la especificación Servlet 2.4 y el sistema fue probado sobre la especificación de Java 1.5 y 1.6.

La configuración se realiza a través de un archivo XML que contiene los parámetros de inicialización para cada componente. A continuación se explican los parámetros para cada uno de los componentes implementados. La clase que implementa cada componente es pasada como parámetro al sistema.

Componente: *repositoryAccesor*

Tipo: org.vikingo.repository.filerepository.FileSystemSourceRepository

Propiedades:

sourcename: Nombre del repositorio de origen de datos.

publisher: Contacto del responsable del repositorio de origen.

description: Breve descripción del repositorio.

mediaType: Tipo de multimedio

absolutePath: Directorio de origen de datos.

outputRepositoryAbsolutePath: Directorio de datos después del procesado.

Componente: *transcoder*

Tipo: org.vikingo.transcoder.mpeg.MpegTranscoder

Propiedades:

outputFolder: Directorio donde se guardan los multimedios transcodificados.

Componente: *metadataExtractor*

Tipo: org.vikingo.extractor.StandardExtractorManager

Propiedades:

metadataFolder: Directorio donde se guardará la metadata autogenerada.

extractors: Lista de clases representando un método de extracción automática de metadatos. Actualmente se tienen dos implementaciones disponibles:

org.vikingo.extractor.context.FileContextExtractor

org.vikingo.extractor.sapi.SapiExtractor

Componente: *indexer*

Tipo: org.vikingo.indexer.lucene.LuceneIndexer

Propiedades:

indexPath: Directorio donde se guardará el índice invertido.

inputFormat: Formato de metadatos a usarse.

luceneAnalyzer: Analizador léxico a ser utilizado en el proceso de tokenización.

Componente: *serviceProvider*

Tipo: org.vikingo.provider.luceneaxis.LuceneAxisProvider

Propiedades:

outputFormat: Formato de datos que devuelve el proveedor de servicios. Ejemplo: objetos, xml, json.

4.5 Problemas encontrados y posibles mejoras

En la comunidad de software no comercial se está trabajando en algunos proyectos para mantener un sistema de reconocimiento automático del habla independiente del locutor. Se están haciendo esfuerzos para desarrollar modelos del lenguaje y modelos acústicos para ser usados con alguno de los motores de reconocimiento de voz de código abierto que actualmente se están desarrollando como *Sphinx4* o *Julius*. Sin embargo esto aún no es una realidad, entre otros motivos por la dificultad del problema en un entorno de habla espontánea con múltiples acentos y múltiples calidades de sonido. Una posible mejora para este componente dependerá de la mejora en los sistemas de entrenamiento de modelos para reconocimiento de voz y un acuerdo entre los fabricantes de tecnología de productos de reconocimiento continuo de voz.

Resumen

Se construyó un sistema mediante el cual el computador puede analizar documentos de audio de manera masiva e indexarlos en una base de datos

local para su posterior búsqueda. Este sistema se integra con tecnologías de reconocimiento automático del habla y tecnologías de recuperación de información de texto y permitirá evaluar el diseño propuesto y la efectividad de la propuesta en general. Se definió un formato de descripción de contenidos de audio genérico para la representación interna de la información de descripción de cada documento.

CAPÍTULO 5

5 PRUEBAS

Este capítulo analizará tanto las pruebas previas realizadas sobre algunos componentes como las pruebas posteriores a la finalización del sistema. Además se ha considerado presentar las pruebas de estabilidad y de integridad de cada componente.

5.1 Plan de pruebas

Se realizaran los siguientes tipos de prueba en el sistema:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de rendimiento
- Pruebas de precisión y retentiva

Las pruebas unitarias consisten en procedimientos mediante los cuales se evalúan unidades individuales de funcionalidad de un componente determinado. El objetivo de una prueba unitaria es comprobar que cierta funcionalidad específica de un componente se comporta correctamente ya sea porque retorna los resultados esperados o pone el estado del componente en los valores esperados. Es muy importante para una prueba unitaria aislar la funcionalidad de otros componentes o entidades externas. Cuando se evalúa la funcionalidad de un componente sin tomar en cuenta independizar los componentes externos se trata de pruebas de integración. Las pruebas de integración por tanto evalúan la funcionalidad de varios componentes en conjunto.

Las pruebas de rendimiento sirven para determinar la cantidad de carga que soportará el sistema ya sea por accesos concurrentes al mismo o volumen de datos de procesamiento. Permitirán conocer factores como la velocidad de indexación o el tiempo de recuperación de consultas ante cantidades masivas de datos. También en estas pruebas se considerarán las pruebas del rendimiento del motor de reconocimiento de voz con respecto a velocidad y consumo de recursos de cómputo.

En las pruebas de precisión y retentiva se evalúa la eficacia del sistema con relación a la satisfacción del usuario con los resultados de las búsquedas que realiza. Para esto se evaluará si los documentos que aparecen en los resultados son aquellos que hubiera esperado.

A continuación se detallan las pruebas específicas que se desarrollaron sobre el sistema aquí expuesto y los resultados obtenidos.

5.2 Pruebas unitarias

Para realizar las pruebas unitarias se hizo uso de la librería de pruebas *jUnit*. Usar una librería para hacer las pruebas de unidad ofrece muchas ventajas respecto a comprobar usando primitivas propias. Entre las ventajas de usar *jUnit* están:

- Provee una manera estándar para crear pruebas de unidad que permiten la ejecución simultánea de todos o un grupo de pruebas de unidad para un proyecto dado.
- Se integra con la herramienta *Ant*, pudiendo automatizar la compilación y ejecución de casos de prueba.

- Se integra con muchos entornos de desarrollo que proveen generación de reportes y entornos de ejecución de *jUnit*.
- Provee clases con funciones de ayuda para comprobar los resultados de las funcionalidades de los componentes que están probándose.

En el presente proyecto se han desarrollado paralelamente a cada componente del sistema, casos de prueba que utilizan estos componentes y comprueban las salidas básicas que debería presentar cada función o método.

La Tabla IV muestra un listado de los componentes o clases individuales del sistema y las pruebas de unidad implementadas para cada elemento.

Componente	Pruebas de Unidad	Descripción
DocumentItemBean	testInterface	Prueba la creación de varios objetos DocumentItemBean
	testNullValues	Prueba la creación de objetos con valores nulos
ResourceLoader	testLoadValidResource	Prueba cargando un recurso válido existente
	testLoadInvalidResource	Prueba cargar un recurso inexistente
FileSystemSourceRepository	testInitialization	Prueba a inicializar el componente
	testBadInitialization	Prueba a inicializar el

		componente con propiedades erróneas
FileSystemSourceIterator	testBadDirectoryInput	Crea el iterador con un directorio inválido
	testValidDirectory	Crea el iterador con un directorio inválido
AudioSystem	testGetFormat	Carga un archivo y trata de determinar el formato
	testGetEncoding	Carga un archivo y trata de determinar la codificación
AudioCommon	testPcmFile	Intenta confirmar si un archivo es PCM
FileContextExtractor|	testDirectoryInput	Prueba ingresando un directorio de datos válido
	testNullInput	Prueba ingresando un directorio inválido
	testGetMetadata	Obtiene metadata y comprueba que es accesible
	testCorrectMetadata	Obtiene metadata y comprueba si cumple con los valores conocidos de la metadata
MpegTranscoderTest	testBadInputFile	Se envía a decodificar un archivo inexistente
	testConvertFile	Se envía a transcodificar un archivo mp3
MicrosoftSapiExtractor	testBadInputFile	Envia a reconocer texto en un archivo inválido
	testTranscribeFile	Transcribe el texto de un audio de corta duración
LuceneIndexer	testIndexSimpleAudioDocument	Envía a indexar un documento
AudioConverter	testmp3towavConversion	Envía a convertir un archivo mp3 a wav
	testmp3towavConversionFake	Envía a convertir un archivo inválido a wav
LuceneProvider	testSearch	Realiza una búsqueda sencilla
	testSearchZero	Realiza una búsqueda sencilla que retorna 0 resultados
	testAnotherSearch	Realiza una búsqueda sencilla y comprueba los resultados de esa

		búsqueda
	testASearch	Realiza una nueva búsqueda y compara resultados
	testGetDocProperties	Realiza una nueva búsqueda y comprueba los valores del primer resultado
	testBounded	Realiza una búsqueda en la cual se han especificado el rango de datos deseado
AudioProviders	testProviders	Comprueba que el sistema tenga instalados los proveedores de sonido deseados
LuceneJsonProvider	testSearch	Realiza una búsqueda y observa el resultado Json

Tabla IV. Listado de pruebas unitarias realizadas sobre el sistema.

Además se hizo uso de la herramienta *EclEmma* 1.3.1 que es una extensión para el entorno de desarrollo Eclipse que permite descubrir el porcentaje de cobertura que tienen los casos de prueba sobre el código del sistema que está siendo sujeto de prueba. En la Fig. 28 se muestra el porcentaje de cobertura de los casos de prueba desarrollados sobre cada uno de los paquetes del sistema.

Se puede además apreciar en la Fig. 28 que los componentes principales del sistema tienen un porcentaje de cobertura alto. Algunas funcionalidades críticas del sistema también serán sometidas a

pruebas posteriores para verificar el comportamiento ante volumen de datos o procesamiento.

Element	Coverage	Covered Instructio...	Total Instructions
repository	23.9 %	2036	8526
src	23.9 %	2036	8526
org.vikingo.util.configuration	0.0 %	0	794
org.vikingo.util	19.9 %	57	287
org.vikingo.transcoder.sound	60.2 %	74	123
org.vikingo.transcoder.mpeg	51.1 %	495	969
org.vikingo.transcoder	3.1 %	3	96
org.vikingo.test	15.0 %	98	655
org.vikingo.repository.services	0.0 %	0	46
org.vikingo.repository.filerepository	58.0 %	184	317
org.vikingo.repository	66.7 %	6	9
org.vikingo.provider.luceneaxis	19.8 %	130	656
org.vikingo.indexer.lucene	42.3 %	243	575
org.vikingo.extractor.sapi.recognizer	100.0 %	6	6
org.vikingo.extractor.sapi	24.7 %	84	340
org.vikingo.extractor.events	48.9 %	22	45
org.vikingo.extractor.context	90.2 %	397	440
org.vikingo.extractor	0.0 %	0	222
org.vikingo.beans	41.3 %	62	150
org.vikingo.authentication	0.0 %	0	80
org.vikingo.audio.test	100.0 %	51	51
org.vikingo.audio.metadata	4.7 %	124	2665

Figura 28. Cobertura de pruebas unitarias del código de la aplicación.

5.3 Pruebas de rendimiento

5.3.1 Rendimiento de indexación

Las pruebas para medir el rendimiento de la indexación se hicieron generando objetos del tipo *AudioDocument* con datos aleatorios. A cada ítem fueron agregadas exactamente mil palabras. Las pruebas

presentadas fueron ejecutadas sobre un procesador Intel Core 2 Duo 2.2GHz. Los resultados de la ejecución se muestran en la Tabla V.

	Prueba 1	Prueba 2	Prueba 3
Cantidad de Documentos	10000	100000	250000
Promedio de tiempo para una indexación	67 ms	57 ms	44 ms
Desviación estándar del tiempo para una indexación	78 ms	67 ms	94 ms
Tiempo en realizar toda la indexación	11 min 12 s	1 h 36 min	3 h 1 min
Tiempo en realizar la operación de optimización	891 ms	27 s	52 s
Tamaño resultante del índice	12.4 MB	125 MB	300 MB

Tabla V. Resultados de las pruebas de indexación.

Se ha verificado que la operación de indexación es independiente de la cantidad de documentos agregados al índice, sin embargo los datos presentan una desviación estándar alta. El promedio de tiempo para una indexación individual disminuye a medida que se agregan más documentos al repositorio, sin embargo la desviación estándar aumenta. La variabilidad de los datos está dada principalmente por la operación de escritura en disco, muchas veces se escribe sobre la misma página de disco y en otras ocasiones se escribe en otra página del disco. Además después de cada cierta cantidad de inserciones se hace una compactación rápida del índice lo cual crea

datos que están fuera de la desviación estándar. En general la operación de indexación no sobrepasa los 200 ms en el peor de los casos.

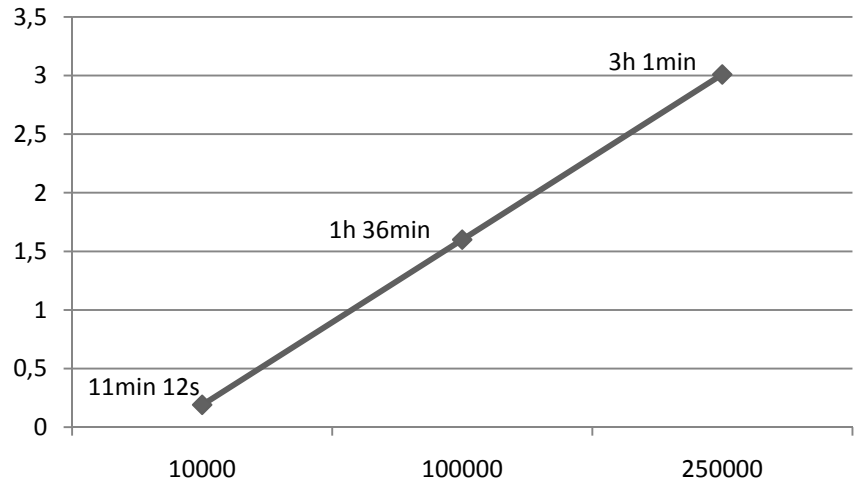

Figura 29. Costo de operación de reconstrucción completa del índice para 10000, 100000 y 250000 documentos.

La Fig. 29 muestra la naturaleza lineal del costo en tiempo de la operación de recrear el índice totalmente a medida que se incrementa el número de documentos en la colección.

La operación de compactación y optimización explícita del índice aumenta dependiendo del tamaño del índice a compactar. Mientras que para un índice de 12 MB demora menos de 1 segundo, para compactar un índice de más de 300 MB demora alrededor de 1

minuto. Esta operación debe ser invocada por el cliente del sistema usando esta consideración.

5.3.2 Rendimiento de búsqueda

Estas pruebas consistieron en medir el tiempo de respuesta de hacer una consulta al índice de búsqueda utilizando datos autogenerados. Para esto se utilizaron los índices obtenidos en la prueba anterior y se usó el mismo hardware.

	Prueba 1	Prueba 2	Prueba 3
Cantidad de documentos total en el índice	10000	100000	250000
Tamaño resultante del índice	12.4 MB	125 MB	300 MB
Tiempo promedio para cada consulta	4.17ms	16.78 ms	35.84 ms
Desviación estándar de los tiempos de consulta	17.3 ms	13.29 ms	15.49 ms
Máximo tiempo registrado	174 ms	130 ms	155 ms
Mínimo tiempo registrado	1 ms	11 ms	26 ms

Tabla VI. Resultados de las pruebas de búsqueda.

La primera observación es que la latencia en el tiempo de búsqueda aumenta a medida que el repositorio posee más información. También se observa que el aumento en los tiempos promedios de

búsqueda es inferior a un décimo de segundo aún para el caso de tener 250000 documentos indexados. Se puede notar además que el tiempo más alto que se registró excede notablemente al tiempo promedio en cada uno de los casos y algo que no se ha registrado en la tabla es el hecho de que este caso siempre ocurrió en la primera observación. Esto se deriva del hecho de que las siguientes veces que se realiza la búsqueda se hace uso efectivo de cacheo, tanto de las operaciones por parte de la máquina virtual Java como del acceso al índice en disco por parte del sistema operativo.

5.4 Pruebas con usuarios

Para este tipo de pruebas se generó un índice a partir de datos reales tomando archivos desde recursos podcast de algunas fuentes en internet y con contenidos variados, de tal forma que se pueda apreciar mejor la satisfacción de las necesidades de búsqueda de los usuarios.

Se utilizaron en total 1000 documentos en audio y audio/vídeo de diversas fuentes para construir el conjunto de datos de prueba para la realización de experimentos con el sistema.

Algunos de estos contenidos estaban en audio y otros en audio/vídeo. Se configuró el sistema para usar el transcodificador nativo para soportar vídeo. Se utilizó a cinco usuarios para evaluar la eficacia del sistema para resolver consultas que devuelvan resultados relevantes o útiles. Se han elegido cuatro consultas para las cuales los usuarios evaluarán los resultados.

Se seleccionaron los siguientes 3 temas de búsqueda:

- Investigación científica
- Tecnologías de computación
- Carrera presidencial

Se pidió a varios usuarios que evalúen la relevancia de los diez primeros resultados devueltos para cada consulta. Se enviaron a los usuarios los contenidos de audio sin ningún tipo de descripción o metadatos, de tal forma que esto no produzca un sesgo en sus respuestas. Los resultados finales de la evaluación se muestran en la Fig. 30. El detalle de las evaluaciones individuales se muestra en el Anexo A.

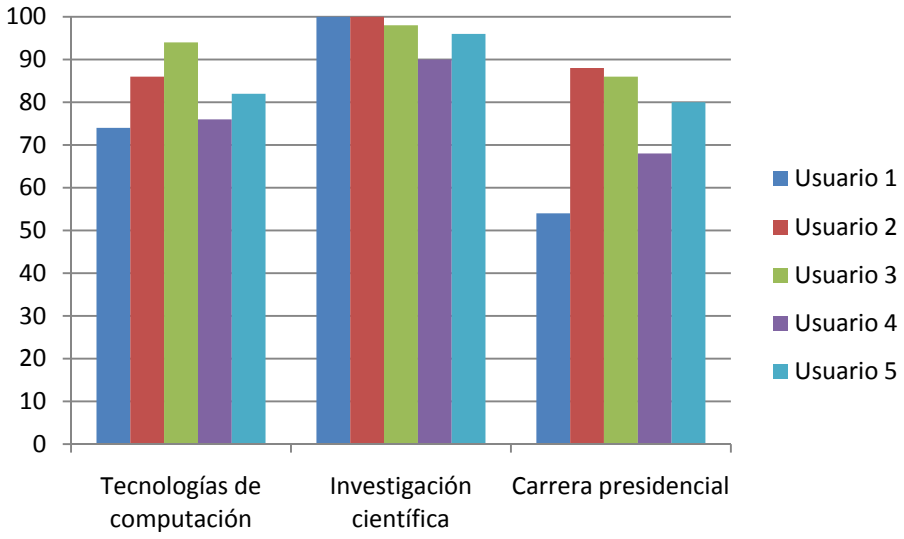

Figura 30. Resultados de la evaluación de relevancia con usuarios.

Los resultados mostraron ser relevantes para los usuarios evaluadores en la mayoría de los casos. La consulta *carrera presidencial* resultó obtener los resultados menos relevantes según los evaluadores, llegando a obtener hasta casi un 50%, seguramente debido al uso en diferentes contextos de la palabra *carrera*.

5.4.1 Precisión y retentiva

Precisión del motor de reconocimiento

Es el cociente entre el número de palabras reconocidas correctamente por el motor de reconocimiento y el total de palabras reconocidas correcta o incorrectamente.

Para aproximar este valor se han usado transcripciones generadas automáticamente y transcripciones por personas escuchando el audio (ver Anexo B).

Transcripción	**Palabras reconocidas correctamente**	**Total de palabras reconocidas**	**Precisión**
Transcripción1	123	385	32%
Transcripción2	53	171	31%
Transcripción3	136	230	60%
Total	312	786	**40%**

Tabla VII. Precisión del motor de reconocimiento de voz.

En la Tabla VII se presentan los resultados individuales para tres transcripciones distintas y el resultado general encontrado para el motor de reconocimiento del habla utilizado, el cual es de aproximadamente 40%. Este valor es razonable teniendo en cuenta que el estado del arte actual de los sistemas de

reconocimiento automático de voz obtienen entre el 30 a 50% de eficacia cuando se trata de situaciones no controladas (50).

Retentiva del motor de reconocimiento

Es el cociente entre el número de palabras reconocidas correctamente por el motor de reconocimiento y el total de palabras presentes de manera hablada.

Para aproximar este valor se han usado transcripciones generadas automáticamente y transcripciones por personas escuchando el audio (ver Anexo B).

Transcripción	**Palabras reconocidas correctamente**	**Total de palabras en el audio**	**Retentiva**
Transcripción1	123	277	44%
Transcripción2	53	134	39%
Transcripción3	136	302	45%
Total	312	713	**44%**

Tabla VIII. Retentiva del motor de reconocimiento de voz.

En la Tabla VIII se muestra el valor de retentiva para el motor de reconocimiento de voz en uso. El valor de retentiva me

indica que tantas palabras del conjunto total de palabras está dejando de reconocer el motor.

Precisión del sistema de búsqueda

Se define como el cociente entre el número de documentos relevantes que se obtienen en una búsqueda y el número total de documentos retornados en la búsqueda.

Transcripción	**Cantidad de Documentos Relevantes**	**Total de Resultados Considerados**	**Precisión**
Tecnologías de Computación	8	10	80%
Investigación Científica	9,8	10	98%
Carrera Presidencial	6,8	10	68%
Total	24,6	30	**82%**

Tabla IX. Resultados de precisión calculados usando las pruebas con los usuarios.

Una aproximación para la precisión en las búsquedas se puede obtener con los datos de evaluación a usuarios presentados anteriormente. Para esta prueba se restringe el conjunto de documentos retornados a los 10 primeros y se asume que una

calificación de relevancia menor o igual a 3 es un documento no relevante para esa búsqueda. Se ha contado la cantidad de documentos relevantes de esta manera para cada usuario y se ha obtenido un promedio para cada una de las consultas. Ver resultados en la Tabla IX.

Retentiva del sistema de búsqueda

Se define como el cociente entre el número de documentos relevantes que se obtienen en una búsqueda y el número total de documentos relevantes presentes en el repositorio de búsqueda. Aproximar el valor de retentiva presenta el reto de tener que conocer previamente el número total de documentos relevantes que existen en el repositorio para una consulta dada, esta cantidad es difícil de calcular especialmente si el repositorio contiene muchos documentos. Otra limitante para aproximar este valor es la variación en la relevancia percibida por cada usuario.

5.5 Análisis de resultados

Se han completado satisfactoriamente las pruebas unitarias y de integración sobre los componentes del sistema, asegurando mayor

fiabilidad y mantenibilidad.

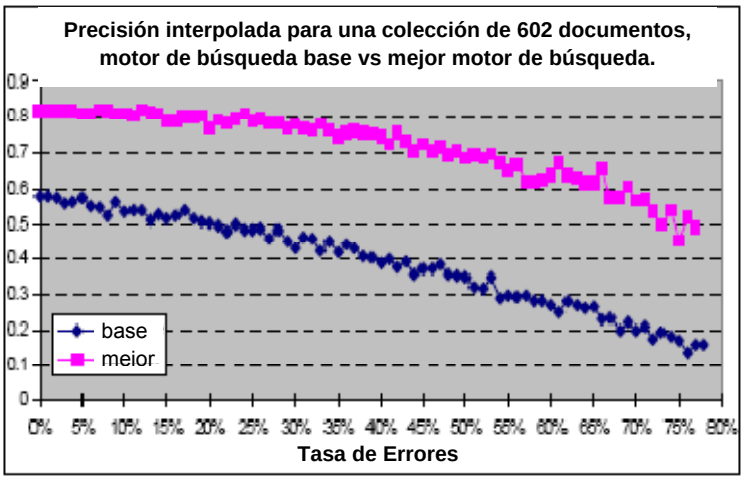

Figura 31. Efectividad de búsquedas versus porcentaje de errores del reconocedor de voz. Tomado de (51).

El valor de precisión para el motor de reconocimiento de voz del sistema es mucho menor al valor de precisión del sistema de búsqueda de manera global. Esta relación entre el porcentaje de errores del reconocedor del habla y la efectividad del buscador se detalla en el artículo escrito por M. Witbrock y A. Hauptmann (51). En la Fig. 31 se observa que la precisión del sistema no mejora considerablemente cuando el porcentaje de errores del reconocedor es menor al 40%. Y en cualquier caso la precisión del sistema de búsquedas siempre es mejor que la precisión del motor de reconocimiento de voz que se usa internamente, confirmando el

comportamiento observado en el sistema de búsqueda aquí presentado.

Resumen

En este capítulo se empezó definiendo las pruebas específicas para cada componente del sistema para asegurar su fiabilidad y su eficiencia con datos autogenerados y datos obtenidos de diversas fuentes. Se recolectaron datos de prueba construyendo un conjunto de datos en audio que abarcan distintos tópicos. Se realizó experimentación y se comprobó que los valores obtenidos confirmaban la teoría e investigaciones realizadas en el área de recuperación de información de audio. A continuación se presentan las conclusiones y recomendaciones.

CONCLUSIONES

1. La no existencia de una manera estándar y generalizada para la utilización de tecnologías de reconocimiento del habla continua provoca que los desarrolladores programen para aplicaciones y plataformas específicas, perdiendo así mucho de portabilidad.

2. La existencia de aún muchos desafíos por superar en el área de reconocimiento automático del habla bajo condiciones no controladas sigue siendo un reto para los investigadores.

3. El patrón de diseño de localización de servicios en tiempo de ejecución e inyección de dependencias sirvieron efectivamente para poder independizar el estado del arte de las tecnologías con la implementación del sistema.

4. El uso del patrón de diseño *Estrategia* permitió asegurar la posible incorporación de nuevas técnicas de extracción automática de metadatos analizando los contenidos binarios.

5. La especificación SAPI (Interfaz de Programación de Aplicaciones por Voz) permitió abstraer en cierta medida la utilización de un motor de

reconocimiento de voz y evitar programar para un motor de reconocimiento de voz específico.

6. El sistema devolvió resultados relevantes a los usuarios evaluadores sin haber utilizado ningún tipo de información textual o generada con intervención de humanos.

7. La precisión y retentiva del sistema en general resultó mayor que la precisión y retentiva de la tecnología de reconocimiento del habla usada internamente. Lo cual significa que aún con tecnologías de reconocimiento del habla poco precisas se pueden obtener resultados relevantes en el sistema de búsqueda.

RECOMENDACIONES

1. Reconsiderar en un futuro el uso de otras convenciones para acceder a tecnologías de reconocimiento automático de voz en cuanto MRCP logre imponerse como protocolo o más empresas den soporte a JSAPI.

2. Considerar invocación a métodos remotos para la implementación de los componentes del sistema que requieren capacidad de procesamiento intensivo.

3. Invertir en tecnologías de reconocimiento automático de voz y tecnologías del habla para su aplicación en proyectos de investigación.

4. Investigar acerca de las maneras de mejorar la accesibilidad en los contenidos y las aplicaciones mediante el uso de tecnologías de procesamiento de voz.

5. Generar una base de datos multimedia que pueda ser utilizada como repositorio general de contenidos multimedia de diversa índole.

6. Promover la generación de contenidos en audio y audio/vídeo en la comunidad académica local como medios de enseñanza y comunicación.

ANEXOS

Anexo A. – Desglose de los datos recabados usando la evaluación a los usuarios.

Consulta: Tecnologías de Computación.

	Usuarios				
Resultados	**1**	**2**	**3**	**4**	**5**
Resultado 1	5	5	5	5	5
Resultado 2	3	5	5	4	4
Resultado 3	1	4	5	3	4
Resultado 4	5	5	5	5	5
Resultado 5	3	5	5	4	5
Resultado 6	3	4	4	4	4
Resultado 7	5	5	5	2	5
Resultado 8	5	4	4	4	4
Resultado 9	5	4	5	5	5
Resultado 10	2	2	4	3	3

Consulta: Investigación Científica.

	Usuarios				
Resultados	**1**	**2**	**3**	**4**	**5**
Resultado 1	5	5	5	3	5
Resultado 2	5	5	4	5	4
Resultado 3	5	5	5	5	5
Resultado 4	5	5	5	5	5
Resultado 5	5	5	5	4	5
Resultado 6	5	5	5	5	4
Resultado 7	5	5	5	4	5
Resultado 8	5	5	5	4	5
Resultado 9	5	5	5	5	5
Resultado 10	5	5	5	5	5

Consulta: Carrera Presidencial.

	Usuarios				
Resultados	**1**	**2**	**3**	**4**	**5**
Resultado 1	5	5	5	5	5
Resultado 2	5	5	5	5	5
Resultado 3	5	5	5	5	4
Resultado 4	5	5	4	5	5
Resultado 5	2	5	5	4	3
Resultado 6	1	5	5	5	4
Resultado 7	1	4	3	1	4
Resultado 8	1	3	3	1	4
Resultado 9	1	3	4	1	5
Resultado 10	1	4	4	2	2

Anexo B. – Resultados de la generación de transcripciones usada para evaluar la precisión y retentiva del motor de reconocimiento de voz.

Recurso 1	
Codificación	MPEG2 Capa 3
Nombre	CNN (4-10-2008 12 PM).mp3
Palabras reconocidas	123
Total de palabras obtenidas	385
Total de palabras en el audio	277

Recurso 2	
Codificación	MPEG2 Capa 3
Nombre	CNN (4-20-2008 7 AM).mp3
Palabras reconocidas	53
Total de palabras obtenidas	171
Total de palabras en el audio	134

Recurso 3	
Codificación	MPEG2 Capa 4
Nombre	TED (2007).mp4
Palabras reconocidas	136
Total de palabras obtenidas	230
Total de palabras en el audio	302

BIBLIOGRAFÍA

(1) Gerald Salton and Michael J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

(2) Bill Coughran, V.P., Engineering Google Inc., Google's Index Nearly Doubles, 11 de Octubre del 2004, http://googleblog.blogspot.com/2004/11/googles-index-nearly-doubles.html

(3) Tim Mayer, Yahoo! Search, Our Blog is Growing Up And So Has Our Index, 8 de Agosto del 2005, http://www.ysearchblog.com/archives/000172.html

(4) B.H. Juang & Lawrence R. Rabiner, Automatic Speech Recognition – A Brief History of the Technology Development, Georgia Tech. Rutgers University, University of California – Sta. Barbara, Encyclopedia of Language and Linguistics, pp. 19-22

(5) Daniel Thalman, Speech Recognition, Ecole Polythecnique Fédérale de Laussane –VRLab, Accesado el 20 de Julio del 2008, pp 3, http://vrlab.epfl.ch/~thalmann/VR/VRcourse_Speechrec.pdf

(6) B.H. Juang & Lawrence R. Rabiner, Automatic Speech Recognition – A Brief History of the Technology Development, Georgia Tech. Rutgers University, University of California – Sta. Barbara, Encyclopedia of Language and Linguistics, pp. 20-23

(7) Daniel Thalman, Speech Recognition, Ecole Polythecnique Fédérale de Laussane –VRLab, Accesado el 20 de Julio del 2008, pp 3-4, http://vrlab.epfl.ch/~thalmann/VR/VRcourse_Speechrec.pdf

(8) Jim Glass, Massachusetts Institute of Technology, Lecture 1, Session 2003, Introduction to Automatic Speech Recognition, MIT - Open Courseware, Slide 11, http://ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-345Automatic-Speech-RecognitionSpring2003/E8336954-4743-47AC-AE39-40743F34572B/0/lecture1.pdf

(9) James A. Larson, *Standard Languages for Developing Multimodal Applications*, Intel Corporation, 2005, pp. 3 – 5

(10) Jim Glass, Massachusetts Institute of Technology, Lecture 3-4, Session 2003, Speech Sounds of American English, Slide 1

(11) Lawrence Rabiner, Biing-Hwang Juang, Fundamentals of Speech Recognition, Prentice-Hall International, USA, 1993b, pp 25-35

(12) Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, Joe Woelfel, 2004, *Sphinx-4 a Flexible Open Source Framework for Speech Recognition*). Mitsubishi Electric Research Labs, Sun Microsystems, Carnegie Mellon University

(13) Petr Motlíček and Jan Černockỳ, All-Pole Modeling for Definition of Speech Features in Aurora3 DSR Task, 2003, 6th Internation Conference in Text,Speech and Dialogue.

(14) C Barras, E Geoffrois, Z Wu, M Liberman, Transcriber: development and use of a tool for assisting speech corpora production, Speech Communication, 2001.

(15) C Van Bael, L Boves, H van den Heuvel, H Strik , Automatic phonetic transcription of large speech corpora, Computer Speech & Language, 2007.

(16) VoxForge, http://www.voxforge.org/home, accesado en Octubre 2007

(17) Lawrence Rabiner, Biing-Hwang Juang, Fundamentals of Speech Recognition, Prentice-Hall International, USA, 1993d, pp 1-5

(18) Ricardo Baeza-Yates, Berthier Ribeiro-Neto, 1999, Modern Information Retrieval, New York, pp 23

(19) Ricardo Baeza-Yates, Berthier Ribeiro-Neto, 1999, Modern Information Retrieval, New York, pp 191-192

(20) Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. pp 560–563

(21) Javier Caicedo, Gonzalo Parra, Xavier Ochoa, 2007, Marco De Trabajo Para Indexación, Clasificación Y Recopilación Automática De Documentos Digitales, Escuela Superior Politécnica del Litoral, Tesis de Grado, pp 35-41

(22) Soumen Chakrabarti, Minning the Web, Discovering Knowledge from Hypertext Data, 2005, Elsevier Science USA, pp 56

(23) Soumen Chakrabarti, Minning the Web, Discovering Knowledge from Hypertext Data, 2005, Elsevier Science USA, pp 209 - 216

(24) Norbert Fuhr, Information Retrieval Current and Future Research, University of Duisburg-Essen, Germany, 2003, http://www.is.informatik.uni-duisburg.de/bib/pdf/talks/Fuhr:03tc.pdf

(25) Milan Sigmund, Voice Recognition by Computer, Tectum Verlag DE, 2003, pp. 8-9.

(26) Peter Heeman, John-Paul Hosom, Yonghong Yan. Automatic Speech Recognition at CSLU, http://cslu.cse.ogi.edu/asr/

(27) Chafe, C. Jaffe, D. Acoustics, “Source separation and note identification in polyphonic music”, Speech, and Signal Processing, IEEE International Conference on ICASSP '86, 1986

(28) Diamantino Caseiro, Isabel Trancoso, “Spoken Language Identification using the Speechdat Corpus”, ICSLP'98 - International Conference on Spoken Language Processing, 1998.

(29) Luis R. Salgado-Garza, Juan A. Nolazco-Flores, Pablo D. Díaz-López,“A Proposed Architecture for a Spoken Information Retrieval with Multimedia Databases”, Computer Science Department, ITESM, 2000.

(30) Richard Mansfield, The Savvy Guide To Digital Music, Indy Tech Publishing, 2005, pp. 5-6

(31) M Meire, X Ochoa, E Duval, “Samgi: Automatic metadata generation v2. 0”, Proceedings of the ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications, 2007

(32) Bowen Zhou, John H.L. Hansen, “SpeechFind: an Experimental On-Line Spoken Document Retrieval System for Historical Audio Archives”, ICSLP-2002: International Conference on Spoken Language Processing, 2002. pp 2.

(33) Bowen Zhou, John H.L. Hansen, “SpeechFind: an Experimental On-Line Spoken Document Retrieval System for Historical Audio Archives”, ICSLP-2002: International Conference on Spoken Language Processing, 2002. pp 3.

(34) M. Siegler, U. Jain, B. Raj, R. Stern, “Automatic Segmentation, Classification and Clustering of Broadcast News Audio,” Proc. DARPA Speech Recognition Workshop, 1997

(35) Lee, K-F, “Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system”, Dissertation Abstracts International Part B: Science and Engineering, 1989.

(36) Boon Pang Lim Haizhou Li Yu Chen, “Language identification through large vocabulary continuous speech recognition”, Chinese Spoken Language Processing, 2004 International Symposium, 2004

(37) Stephen V. Rice, Stephen M. Bailey, “Searching for Sounds:A Demonstration of FindSounds.com and FindSounds Palette”, Proceedings of the International Computer Music Conference, 2004, pp. 215-218.

(38) EditoSoftware, Definiciones de estándares, http://www.editiosoftware.com/es/company/standards.html, accesado en Octubre del 2007

(39) N Komaki, M Haseyama, T Yamamoto, “A Simple Word Spotting Method Based on Template Matching For Speech Retrieval”, IEIC Technical Report, University of Hokkaido, Japan, 2005.

(40) B. Tate, “Bitter Java”, Manning 2002, pp 151.

(41) Aidan Hogan, RDF/XML Podcast Specification, 2005, http://sw.deri.org/2005/07/podcast/

(42) Justin Zobel, Alistair Moffat, “Inverted files for text search engines”, ACM Computing Surveys (CSUR), 2006

(43) Hansen, J.H.L. Huang, R. Zhou, B. Seadle, M. Deller, J.R. Gurijala, A.R. Kurimo, M. Angkititrakul, P., “SpeechFind: Advances in Spoken Document Retrieval for a National Gallery of the Spoken Word”, Speech and Audio Processing, IEEE Transactions, 2005.

(44) Steve Cronen-Townsend, Yun Zhou, W. Bruce Croft, “A framework for selective query expansion”, Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004, pp. 236-237.

(45) Alexander Maedche, “Ontology Learning for the Semantic Web”, Springer, 2002, pp. 174.

(46) Liwen Vaughan, “New measurements for search engine evaluation proposed and tested”, Information Processing & Management , 2003

(47) Keen, E. Michael, “Presenting Results of Experimental Retrieval Comparisons”, Information Processing and Management, v28 n4 p491-502 Jul-Aug 1992.

(48) T. Ayres, B. Nolan, “JSAPI Speech Recognition with Sphinx4 and SAPI5”, Proceedings of the 4th international symposium on Information and communication technologies, pp 181 – 183

(49) P. Zaitsev, V. Tkachenko, “High Performance Full Text Search for Database Content”, EuroOSCON 2006.

(50) C. Chelba, T. Hazen, M. Saraclar, “Retrieval and Browsing of Spoken Content”, A discussion of the technical issues involved in developing information retrieval systems for the spoken world. IEEE Signal Processing Magazine, Mayo 2008, pp 39.

(51) M. Witbrock, A. Hauptmann, “Speech Recognition and Information Retrieval: Experiments in Retrieving Spoken Documents”, Carnegie Mellon University, Pittsburg.