Un sistema de recuperación de información como mecanismo de interacción con un robot humanoide

BUAP

Orlando Ramos Flores

Asesor: Dr. David Eduardo Pinto Avendaño

Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla

Tesis para obtener el título de
Maestro en Ciencias de la Computación

Puebla, Pue. Julio 2017

Agradecimientos

Agradezco a mi madre y hermanos por su apoyo y compresión que me brindaron para poder concluir esta meta personal y profesional.

A mis profesores por su enseñanza y dedicación. A mi asesor el Dr. David Pinto Avendaño por compartir sus conocimientos y apoyarme durante mis estudios.

Un agradecimiento especial al Laboratorio Nacional de Supercómputo del Sureste de México (LNS), por habernos brindado la infraestructura para la realización de este trabajo de tesis.

También agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo brindado.

Resumen

En la actualidad el crecimiento de nuevas tecnologías ha incrementado de forma considerable, así como el número de usuarios que las utilizan, esto conlleva a un aumento de información, se estima que actualmente existen 4.46 billones de páginas web a nivel mundial. En cuanto a los dominios se refiere, el Centro de Información de la Red México (NIC México) se encarga de la administración de los dominios .mx desde el año 1989, a partir de 1992 se contaba con 45 dominios .mx y actualmente existen 849,036 dominios. Con este crecimiento de datos en Internet, cuando se realiza una búsqueda en la web por parte de un usuario, su objetivo principal es obtener la información más relevante basada en su consulta. Por tal motivo en este trabajo se investigaron e implementaron algoritmos y herramientas para la creación de un sistema de recuperación de información, tomando como base de conocimiento páginas web en idioma Español. Se diseño e implemento un crawler para la extracción de páginas web, con el objetivo de recopilar un corpus de gran volumen. La implementación contó con los recursos de hardware del Laboratorio Nacional de Supercómputo del Sureste de México (LNS). El Laboratorio de Ingeniería de Lenguaje y del Conocimiento (LKE) adquirió un robot humanoide para diferentes propósitos de investigación, cuenta con diversos sistemas de adquisición de señales, tales como cámaras, electroencefalogramas (EEG) y micrófonos, además cuenta con un sistema de voz en Inglés basado en un modulo de lenguaje de marcado de inteligencia artificial (AIML), que consiste en una base de preguntas y respuestas predefinidas. Por ello el sistema de recuperación de información sirvío como mecanismo de interacción con el robot humanoide, de esta forma su base de conocimiento contempla los documentos recuperados por el crawler.

Durante el preprocesamiento de los documentos HTML se extrajo el texto para formar el corpus, al mismo tiempo que se utilizó un modelo Naïve Bayes para la clasificación del idioma del documento HTML analizado, ya que únicamente se indexaron los documentos en idioma Español. Se diseño e implemento un motor de recuperación de información, con algoritmos de Procesamiento del Lenguaje Natural relacionados a la recuperación de información, se propuso un modelo de espacio vectorial con ranking de documentos, este modelo contempla

el uso de consultas en lenguaje natural, puesto que calcula el puntaje de aquellos documentos que contengan las palabras más relevantes, es decir, son las palabras que aparecen en menor cantidad dentro del corpus, después se crea un top de documentos relevantes basados en el puntaje obtenido (en orden descendente), esta lista de documentos es la que se regresa al usuario. Además se desarrolló un modelo para obtener el resumen automático del documento por el método de extracción llamado TextRank. Se utilizó una maquina virtual proporcionada por el LNS como servidor, que soporta al crawler y al motor de recuperación de información. Para el acceso al servidor se desarrolló un API RESTful con autenticación básica de usuarios, proporcionando un servicio web por medio de HTTP, de esta forma los usuarios pueden realizar consultas al sistema de recuperación de información desde cualquier lugar.

Puesto que el robot humanoide cuenta con micrófonos se experimentó por medio de consultas orales, para ello se aplicaron algoritmos y herramientas para el reconocimiento de voz, convirtiendo estos datos en información de tipo texto, cuando se tiene la consulta en texto se envía por medio de la API RESTful al servidor, el motor de recuperación de información regresa un resumen del documento más relevante. El robot humanoide cuenta con diferentes motores en el rostro para mostrar expresiones y simular el habla moviendo los motores de los labios, por este motivo se investigaron las herramientas para utilizar un sintetizador de voz en idioma Español, el cual recibe una cadena de texto y la convierte en un archivo de audio, además del sintetizador de voz con el que cuenta el LKE. Cuando se tiene el resumen del documento más relevante, se pasa a través del sintetizador de voz del LKE, así el robot humanoide reproduce el archivo de audio al mismo tiempo que mueve los motores faciales, de esta forma regresa la consulta solicitada por el usuario de una forma visual y natural.

En este trabajo también se implemento una interfaz web para el sistema de recuperación de información, puede ser consultado de forma tradicional, es decir escribiendo en un formulario la consulta, y por medio de una consulta de voz activando el micrófono. La interfaz web solo regresa los primeros diez documentos relevantes, donde se puede visualizar el resumen, todo el documento o una versión breve del documento, además se resaltan en negrita las palabras de la consulta en el documento.

Abstract

Currently the growth of new technologies has increased considerably, as well as the number of users who uses them, this leads to an increase of information, it's estimated that currently there are 4.46 billions web pages worldwide. As far as the domains are concerned, the Network Information Center Mexico (NIC Mexico) has been in charge of the administration of .mx domains since 1989, since 1992 there were 45 domains .mx and there are currently 849,036 domains with this growth of data on the Internet, when a web search is performed by a user, its main goal is to obtain the most relevant information based on your query. For this reason, in this work, algorithms and tools for the creation of an information retrieval system were investigated and implemented, based on Spanish-language websites. We designed and implemented a crawler for the extraction of web pages, with the aim of compiling a large volume corpus. The implementation was based on the hardware resources of the LNS (Laboratorio Nacional de Supercómputo del Sureste de México). The laboratory of Language & Knowledge Engineering (LKE) acquired a humanoid robot for different research purposes, has several systems of signal acquisition, such as cameras, electroencephalograms (EEG), microphones, and has a voice system in English-language based on an Artificial Intelligence Mark-up Language (AIML), which consists of a base of predefined questions and answers. This is why the information retrieval system serves as a mechanism for interaction with the humanoid robot, so its knowledge base includes the documents retrieved by the crawler.

During the preprocessing of the HTML documents, the text was extracted to build corpus, while a Naïve Bayes model was used to classify the language of the analyzed HTML document, since only the spanish language documents are indexed. An information retrieval engine was designed and implemented, with algorithms of Natural Language Processing related to information retrieval, we proposed a vector space model with documents ranking, this model considers the use of queries in natural language, since it calculates the score of those documents that contain the most relevant words, i.e., they are the words that appear in less quantity within the corpus, then a top list of relevant documents is created based on the obtained score (in

descending order), this list of documents is the one that is returned to the user. In addition, a model was developed to obtain the automatic summary of the document by the extraction method called TextRank. We used a virtual machine provided by the LNS as a server, which supports the crawler and the information retrieval engine. For access to the server a API RESTful was developed with basic user authentication, providing a web service through HTTP, in this way users can query the information retrieval system from anywhere.

As the humanoid robot has microphones, we experimented by means of oral queries. For this purpose, algorithms and tools for voice recognition were applied, converting this data to text information, then the text query is sent through the API RESTful to the server, the information retrieval engine returns to summary of the most relevant document. The humanoid robot has different motors in the face to show expressions and simulate speech by moving the motors of the lips, for this reason we investigated the tools to use a voice synthesizer in Spanish language, which receives the string of text and the converts it into an audio file, in addition to the voice synthesizer that the LKE has. When you have the summary of the most relevant document, it is passed through the voice synthesizer of the LKE, thus the humanoid robot reproduces the audio file while moving the facial motors, in this way returns the query requested by the user of a visual and natural form.

In this work also implemented a web interface for the information retrieval system, it can be consulted in a traditional way, i.e. by writing the query in a text box, and by means of a voice query activating the microphone. The web interface only returns the first ten relevant documents, where you can view the summary, the entire document or a short version of the document, and bold the words of the query in the document.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Capítulo 1

Introducción

La gran cantidad de información que se genera en la web ha aumentado de forma considerable en pocos años, esto gracias al avance de la tecnología y a la penetración del internet en más sectores de la población. La información de la web consiste de millones de documentos. Un documento visto desde la óptica de la informática, es un archivo con determinados atributos ya que contiene datos textuales y no textuales. Los documentos textuales contienen información escrita, y gráfica en algunos casos. Los documentos no textuales contienen datos multimedia: gráficos (imágenes), audio y vídeo. Dentro de los documentos textuales se pueden encontrar documentos estructurados, semi-estructurados y no estructurados. Los documentos estructurados contienen una estructura predefinida, tienen etiquetas definidas con el objetivo de mostrar información relevante más allá de la información presentada, es decir tienen un contexto semántico de los datos que lo conforman. Los documentos semi-estructurados son aquellos que en su mayoría de contexto tienen elementos de un documento estructurado, dejando algunas partes del documento sin estructura y sus datos están sometidos a constantes cambios. Los documentos no estructurados pueden contar con un título y párrafos, etc., pero no contemplan ninguna estructura bien definida.

Según el sitio World Wide Web Size[1] basado en el trabajo de [1], la cantidad actual de páginas web es de por lo menos 4.46 billones. Con está enorme colección de documentos disponibles, es necesario utilizar mecanismos de búsqueda de tal manera que cualquier persona pueda acceder a ellos, realizando búsquedas sencillas con palabras clave (keywords) sobre un determinado tema. Para cubrir está necesidad se hace uso de técnicas de IR.

[1] http://www.worldwidewebsize.com/

Aunque actualmente existen diversos sitios web de recuperación de información, como son: Google, Bing, Yahoo, Ask, etc., que proporcionan un servicio de búsqueda web a cualquier usuario, la motivación principal de este trabajo es crear un Sistema de Recuperación de Información (IRS: Information Retrieval System), para realizar una HRI con el robot humanoide (Arthur) del Laboratorio de Ingeniería del Lenguaje y del Conocimiento (LKE: Language & Knowledge Engineering). La base de conocimiento está conformada por páginas web en idioma español, que se extraen mediante un crawler. Además se implementa un ASR (Automatic Speech Recognition) en español, que sirve como mecanismo de entrada para las consultas por voz del usuario, y como mecanismo de salida se utiliza un Sintetizador de Voz en español (TTS: Text to Speech), dónde Arthur, por medio de audio, expresa el resultado de la consulta. Además se cuenta con una sistema web para interactuar con el IRS de la forma tradicional, es decir, escribiendo la consulta en un formulario y realizando peticiones de consulta al servidor.

El IRS hace uso de herramientas y técnicas del NLP para crear los mecanismos de búsqueda, clasificando los documentos por idioma, indexando los documentos en la estructura de datos posting list, utilizando algoritmos para la corrección automática de palabras, para generar conectores de búsqueda (AND, OR, NOT), ordenamiento por ranking de documentos para obtener el conjunto más relevante, resúmenes automáticos, etc.

1.1 Antecedentes

En el mercado actual, existen varios tipos de robots que interaccionan con los humanos de diferentes maneras. Sin duda, el uso de robots humanoides es un tópico de sumo interés en la actualidad. Se pueden encontrar diversos robots en la literatura que proveen de mecanismos básicos para el diálogo entre el robot y los seres humanos, que sin embargo, ya presentan características muy interesantes. Tal es el caso del robot humanoide Alpha 2[2], el cual interactúa con los seres humanos, conversando, contestando a preguntas, siendo tutor, intérprete, además de que puede realizar llamadas y comprobar los correos de voz; también puede leer y enviar mensajes de texto y correos electrónicos. Este robot es capaz de controlar el equipo de una oficina y el fax vía wifi. Contiene una base de conocimientos general, además de la función de búsqueda en la web; inclusive puede en las redes sociales compartir fotografías tomadas por él, utilizando la detección de rostros.

[2]http://www.ubtrobot.com/product/detail1.html

Otro robot que ha llamado la atención es el denominado Qbo[3], el cual es un robot de código abierto con software en Linux que cuenta con una visión estereoscópica, sistema de reconocimiento de voz, sistema para la síntesis de voz, una API y panel de control para personalizar la configuración. En cuanto a las conexiones, cuenta con Bluetooth y Wifi. También incluye sensores para sortear obstáculos cuando realiza movimientos.

1.2 Planteamiento del problema

El LKE ha adquirido un robot humanoide llamado Arthur, desarrollado por la empresa Hanson Robotics. El dispositivo tiene una multitud de actuadores y sensores controlados por diversos microprocesadores. Su potencial se encuentra principalmente en su capacidad de expresión emocional, pues con los motores que posee bajo la piel, éste es capaz de mover sus mejillas, ojos, nariz, labios, generando un número sumamente amplio de expresiones faciales que son de lo más natural posible. Además tiene un sistema de reconocimiento y sintetizador de voz en Inglés, y expresiones de su rostro con los que puede interactuar con los humanos de manera eficaz.

En el LKE se ha trabajado por más de 10 años el tratamiento automático de la información, y por tanto, se apetece llevar a cabo investigaciones tendientes al desarrollo de aplicaciones inteligentes fruto de la unión entre el robot humanoide y sus capacidades tecnológicas, y las técnicas de procesamiento automático del lenguaje natural que se cultivan en este laboratorio.

La necesidad de IR es algo común hoy en día, y está presente en empresas, centros de investigación, dependencias gubernamentales, instituciones educativas, etc., aunque el mayor porcentaje de IR se realiza en motores de búsqueda web. Es por tal motivo que para este trabajo se crea un corpus por medio de un crawler, para indexar documentos de la web en idioma Español, y utilizar está base de conocimiento para construir un IRS con una HRI con el robot humanoide. Además se necesitan implementar algoritmos para el reconocimiento y sintetizador de voz en idioma español.

[3] http://openqbo.org/wiki/doku.php

1.3 Objetivos

En esta sección se presenta el objetivo general y específicos de este trabajo.

1.3.1 Objetivo general

Creación de un sistema de recuperación de información que sirva como mecanismo de interacción entre los seres humanos y el robot humanoide y que permita llevar a cabo una serie de consultas avanzadas basadas en entrada de datos multimodal.

1.3.2 Objetivos particulares

- Diseñar un modelo de recuperación de información que sirva como mecanismo interactivo entre los seres humanos y el robot humanoide.
- Construir un crawler para el indexado de páginas web en idioma español.
- Procesar todos los documentos obtenidos del crawler, y crear un corpus con los datos resultantes.
- Construir los algoritmos para la recuperación de información de las consultas solicitadas por el usuario.

1.3.3 Preguntas de investigación

- ¿En que medida se podrá construir un índice de documentos de páginas web en idioma español?
- ¿El IRS será lo suficientemente eficiente para responder a las consultas planteadas por el usuario?
- ¿Qué tipo de preguntas son aquellas que podrán responderse mediante el modelo a desarrollar en este trabajo de tesis?

1.4 Organización de la Tesis

Este documento se organiza de la siguiente forma: en el Capítulo 1 se presenta la Introducción, así como los antecedentes, planteamiento del problema y los objetivos. Los trabajos relacionados

se describen en el Capítulo 2. En el Capítulo 3, como marco teórico se muestran las definiciones de conceptos, como son crawler, modelos de IR que se utilizan en este trabajo, métricas de evaluación, y resumen automático que se emplea para la selección de las oraciones más relevantes de los documentos. En el Capítulo 4, se presenta el modelo propuesto para la creación del IRS con retroalimentación HRI. Los experimentos y resultados se describen en el Capítulo 5. Las conclusiones se mencionan en el Capítulo 6. Después de este Capítulo se listan las referencias utilizadas. Además se cuenta con los siguientes anexos: Crawler, Reconocimiento de voz automático y Sintetizador de voz.

Capítulo 2

Trabajos Relacionados

En esta sección se introducen algunos trabajos que se encuentran de alguna manera relacionados con la recuperación de información utilizando robots, además se mencionan trabajos relacionados con el procesamiento de imágenes y sonido.

En [2] expresan que todo crawler se encarga fundamentalmente de automatizar el proceso de seguir los enlaces, pues se limita a enviar peticiones HTTP para documentos a otras máquinas en internet, al igual que un navegador web hace cuando el usuario hace clic en los enlaces, e implica la interacción con cientos de miles de servidores web. Para la construcción del crawler se enfocan en descargar únicamente páginas web, que son relevantes para un tema o conjunto de temas predefinidos. Aplicaron técnicas de clasificación para construir clasificadores por instancias positivas y negativas, usando un algoritmo llamado 1 (M-C) logrando una alta precisión de la clasificación, tan alta como la de una máquina de soporte vectorial (SVM). La estrategias que aplicaron fueron: la búsqueda por amplitud (Breadth-First), rastreo repetitivo de páginas web, crawler dirigido a sitios específicos, caminos aleatorios y muestreos para estimar el tamaño de los documentos en línea, y rastreo profundo web que son los datos accesibles y que están actualmente contenidos en bases de datos y sólo puede ser descargado a través de medios de solicitudes o formularios correspondientes.

En [3], la WWW es vista estructuralmente como un grafo dirigido o dígrafo, donde cada página web es un vértice y cada enlace es una arista. La conectividad del dígrafo además de permitir la navegación ayuda a caracterizarlo mediante el procesamiento de su conectividad en matrices de adyacencia, indicando con un 1 la existencia de una arista y 0 en caso contrario. Cuando se consideran dos vínculos se presentan patrones básicos formados por estos enlaces: una página vincula a otra página (aval), dos páginas apuntándose entre sí (mutua relevancia), una página referenciando dos páginas distintas (co-citación), dos documentos vinculados a un

tercero (apareo/coupling), cuando la p1 vincula a p2, la cual a su vez enlaza a p3, entonces transitivamente, p1 es aval de p3 (aval transitivo). Al combinarse estas estructuras, se forman patrones muy complejos, interesándose principalmente en la generalización de la co-citación con la estructura arbórea de salida (out degree) y la generalización de apareo con la estructura arbórea de entrada (in degree). También presentan un análisis de algoritmos de ordenación por relevancia basados en la conectividad, divididos en: (1) algoritmos dependientes de la consulta del usuario: HITS (Hyperlink Induced Topic Search), ARC (Automátic Resource Compiler), SALSA (Stochastic Approach for Link structure Analysis). (2) Algoritmos cuasi-dependientes de la consulta del usuario: Co-citacion, TOPIC (TOronto Page Influence Computation). (3) Algoritmos independientes de la consulta del usuario: Page Rank (PR). Además se presentan a los algoritmos Page Rank, WiseNut (WiseRank) y Teom como los nuevos motores de búsqueda de acceso público y gratuito a partir de la década del 2000.

En [4] se presentan modelos de lenguaje (LM) enfocados a la recuperación de información (IR), combinando y evaluando el proceso de suavizado de Pitman-Yor, la ponderación de características TF-IDF y el modelo basado en retroalimentación. El framework que utiliza es el de consulta de verosimilitud (QL) de LMs para IR, que es equivalente a usar un modelo bayesiano multinomial para la clasificación, con las clases correspondientes a los documentos y una probabilidad a priori uniforme sobre los LMs. La retroalimentación es un método tradicional utilizado en la IR que puede tener un gran impacto en el rendimiento de la recuperación. Los documentos mejor clasificados se pueden utilizar para construir un modelo de consulta para un segundo pase de recuperación. Con los modelos de lenguaje se tienen dos maneras de formalizar esto: Kullback-Leibler divergence (KL-divergence) [5] o también conocida como entropía relativa y Modelos de Relevancia [6], ambos métodos permiten la sustitución del vector de consulta con un modelo.

Un sistema QA (Question Answer) que se limita a responder sólo a las preguntas sobre previsiones meteorológicas, es presentado en [7]. El sistema contempla dos motores, un motor de QA y uno de IE (Information Extractor). El motor IE está formado por un crawler, para la descarga de páginas seleccionadas de un sitio web con información meteorológica, además utiliza un wrapper que extrae información sobre el clima de las páginas web semi-estructuradas, y se almacena en una base de datos relacional. El motor QA analiza las preguntas en lenguaje natural, para traducirlas a sentencias SQL, después se realizan las consultas en la base de datos y se recuperan las respuestas, y estas respuestas se convierten a oraciones en lenguaje natural para devolver la respuesta a la pregunta planteada, además se desarrolló una ontología manual con codificación de términos de tiempo específicos y ciudades, para el análisis de preguntas en

el proceso de inferencia dentro del motor QA. En este trabajo se lograron buenos resultados en la precisión (90.9%) y recuperación (75.0%), sobre un dominio restringido. Sin embargo, las cifras no son suficientes para un sistema práctico real.

En [8] desarrollaron un sistema llamado bot de discusión, que proporciona respuestas a preguntas de un foro de discusión de estudiantes, utilizando técnicas de procesamiento de lenguaje natural y recuperación de información. Primeramente crearon un corpus anotado de 1236 foros de discusión archivados de semestres anteriores, y 279 documentos de cursos. Cuando se publica una pregunta en el sistema, se extraen las características (palabras incluidas, frecuencia, etc.), después el sistema intenta coincidir el interés del estudiante (usuario) en todos los datos archivados, posteriormente, una lista de los mensajes relacionados con la pregunta se clasifican con base a métricas predefinidas. Se extrae la mejor respuesta de los principales candidatos de la lista. Para responder a los interés del estudiante, se calcula directamente una similitud coseno entre la pregunta de los post en los foros y los datos archivados con la técnica TF*IDF [9]. Para la lista de resultados se hizo un ranking, se toma el primer resultado y se aplica el algoritmo forward-backward, que consiste en recorrer el grafo de hilos (post en los foros) hacia adelante (forward) con una DFS, seguida por un recorrido DFS hacia atrás (backward). Si se cumple con la prueba límite, se detiene en ese nivel, además durante el recorrido DFS se aplican algunas reglas de extracción. El mensaje de respuesta se une con una referencia a la fuente original, ya sea una discusión (post) o un documento (curso), y se presenta al usuario.

Un sistema prototipo que consiste en un juego de preguntas y respuestas, entre un robot y un niño es presentado en [10], el escenario es una interacción niño-robot, donde ambos observan en un televisor un video de dibujos animados (de 10-15 minutos), después de ver el video, el robot genera preguntas y respuestas en relación a la historia del video. Realiza preguntas al niño y en caso de ser correcta, continúa con las preguntas. Además el niño puede hacer preguntas al robot y este las contestará. Para el proceso general de QA del video, primeramente se realiza el pre procesamiento para la generación de características, que consiste de redes neuronales convolucionales (CNNs) y redes neuronales recurrentes (RNNs). Lo que sigue es formar jerarquías de conceptos profundos para el aprendizaje de conceptos lingüístico-visual y almacenar el conocimiento del video en una estructura de red. Después se hace la conversión de visual-pregunta (convierte las escenas de vídeo observados a las preguntas), y por último un componente de microcódigo es candidato para usar RNNs (codifica las palabras en la respuesta y genera las próximas palabras de respuesta), el método es similar a las técnicas de Q&A de imagen usadas en [11]. Para los experimentos utilizaron el robot NAO Evolution - V5

de RobotsLAB y los vídeos son de Pororo, un famoso dibujo animado, que en su contenido presenta material para el aprendizaje temprano de idiomas para los niños.

La creación de un robot, mostrado en [12], es capaz de navegar en entornos urbanos desconocidos sin usar datos de GPS o conocimientos previos de mapas. El sistema se encarga de la comunicación con los transeúntes, la adquisición de la información pertinente de los seres humanos, los controles de verosimilitud, y la representación interna de la información en forma de gráficos de rutas topológicas. La técnica que usan para la comunicación es una "estructura de diálogos para preguntar por direcciones", que está compuesta por cuatro fases: (1) Introducción, donde el autor de la pregunta se dirige a un entrevistado y define la tarea, es decir, dar direcciones específicas de un objetivo de ubicación. (2) Proporcionando direcciones, el entrevistado proporciona la información necesaria. (3) Confirmación, cualquiera de los socios confirma la información, en esta fase se pueden hacer más consultas. (4) El autor de la pregunta agradece al entrevistado. Además cuentan con una estrategia para alinear el sistema de referencia personal, estos son incluidos en una máquina de estados finitos que controla el dialogo.

En [13] se presenta la creación de un sistema informático de inteligencia artificial, que es capaz de responder a preguntas formuladas en lenguaje natural, desarrollado por IBM y nombrado como Watson, esto se debe a una base de datos almacenada localmente, la información proviene de multitud de fuentes, incluyendo enciclopedias, diccionarios, tesauros, artículos de noticias, y obras literarias, al igual que bases de datos externos, taxonomías, y ontologías (específicamente DBpedia, WordNet, y Yago). Watson utiliza DeepQA [14], que es una arquitectura con una metodología de acompañamiento. Los principios generales en DeepQA son paralelismo masivo (en la consideración de múltiples interpretaciones e hipótesis), muchos expertos (facilitar la integración, aplicación y evaluación del contexto de una amplia gama de interrogación y de contenido de análisis probabilísticos de forma flexible), la estimación de la confianza generalizada (un sustrato de procesamiento de la confianza subyacente aprende cómo apilar y combinar las puntuaciones), y la integración de los conocimientos superficiales y profundos (equilibrar el uso de la semántica y la semántica estricta de poca profundidad, aprovechando muchas ontologías formadas de manera libre). El desafío de este sistema fue el de competir con humanos en el concurso Jeopardy[1] obteniendo una precisión inicial de 70%, con el continuo desarrollo de algoritmos, llegaron a un 85% de precisión y respondiendo a

[1]Jeopardy! es un concurso de televisión estadounidense de conocimientos con preguntas sobre numerosos temas como historia, idiomas, literatura, cultura popular, bellas artes, ciencia, geografía, y deportes.

las preguntas en 5 segundos o menos, utilizando una plataforma de computo masivamente en paralelo de alto rendimiento.

Utilizando el procesamiento de lenguaje natural y la visión por computadora, en [15] presentan un método para contestar preguntas basadas en escenas del mundo real (imágenes), que es una reminiscencia de una prueba de Turing visual, recibiendo como entrada lenguaje natural y como salida un análisis de la escena visual en un framework probabilístico (Bayesiano), la combinación de este enfoque es llamada multi-mundo. El sistema de búsqueda de respuestas está entrenado exclusivamente en los pares de preguntas y respuestas (Q, A). En el enfoque de un solo mundo, se genera el mundo mediante la ejecución de un algoritmo de segmentación semántica [16] sobre la imagen y se colecta la información de los objetos reconocidos, la posición 3D y el color (una única interpretación de una escena visual). En cambio para un multi-mundo, el enfoque se integra a lo largo de muchos mundos latentes, y por lo tanto se tienen diferentes interpretaciones de la escena visual al igual que preguntas. Para las predicciones se utilizó una distribución lineal logarítmica sobre las formas lógicas (árboles semánticos).

En [17] presentan un software de reconocimiento de voz llamado Julius, de alto rendimiento, de código abierto y que posee un gran vocabulario, se utiliza tanto para la investigación académica y para aplicaciones industriales. Soporta modelos de lenguaje estándar como el modelo estadístico de N-gramas, reglas basadas en gramáticas, una lista simple de palabras para el reconocimiento de palabras aisladas, así como modelos ocultos de Markov (HMM) como un modelo acústico. Además soporta el procesamiento de archivos de audio y streaming de audio, también soporta auto-splitting en la entrada con pausas largas. La estructura interna del sistema consiste de un motor de instancias, que contiene todos los módulos requeridos para un sistema de reconocimiento: entrada de audio, detección de voz, extracción de características, modelo de lenguaje, modelo acústico y proceso de búsqueda. El sistema está escrito en lenguaje C puro, y corre en las plataformas de Windows, Linux, Mac OS X, Solaris y otras variantes de Unix.

Capítulo 3

Marco Teórico

En esta sección se listan los fundamentos teóricos utilizados en este trabajo de tesis, describiendo de forma general cada uno de los enfoques.

3.1 Crawler

El Crawler es un bot (programa) que funciona de forma periódica y automática, tomando como base una lista de semillas (URL: Uniform Resource Locator), visita una semilla y analiza la página web en busca de nuevas URL y las añade a su lista. Por lo general se copian y almacenan las páginas, de tal manera que se puedan volver a ver y navegar, es decir se toma una instantánea de la página visitada. Además se implementan técnicas de selección de páginas web, así como la calendarización para visitar nuevamente a la página en busca de actualizaciones. Este procedimiento se repite para todas las URL de la lista.

En la Figura 3.1 se observa el modelo estándar del crawler, dónde se puede apreciar una lista de URL pendientes (semillas) que al ser visitadas, se descarga, almacena el documento en un directorio y se analiza para extraer las URL contenidas, para enviarlas a la lista de pendientes como nuevas, y por último se agrega la URL a la lista de visitadas. Sin embargo existen otros modelos más avanzados, como es el motor de búsqueda Google.

En la Figura 3.2, el crawler web (descarga de páginas web) se realiza mediante varios crawler distribuidos. Hay un servidor de URL (URL server) que envía listas de direcciones URL para ser buscadas en los crawler. Las páginas web que se recuperan son enviadas a un servidor de almacenamiento (Store server). El servidor de almacenamiento a continuación, comprime y almacena las páginas web en un repositorio (repository). Cada página web tiene un

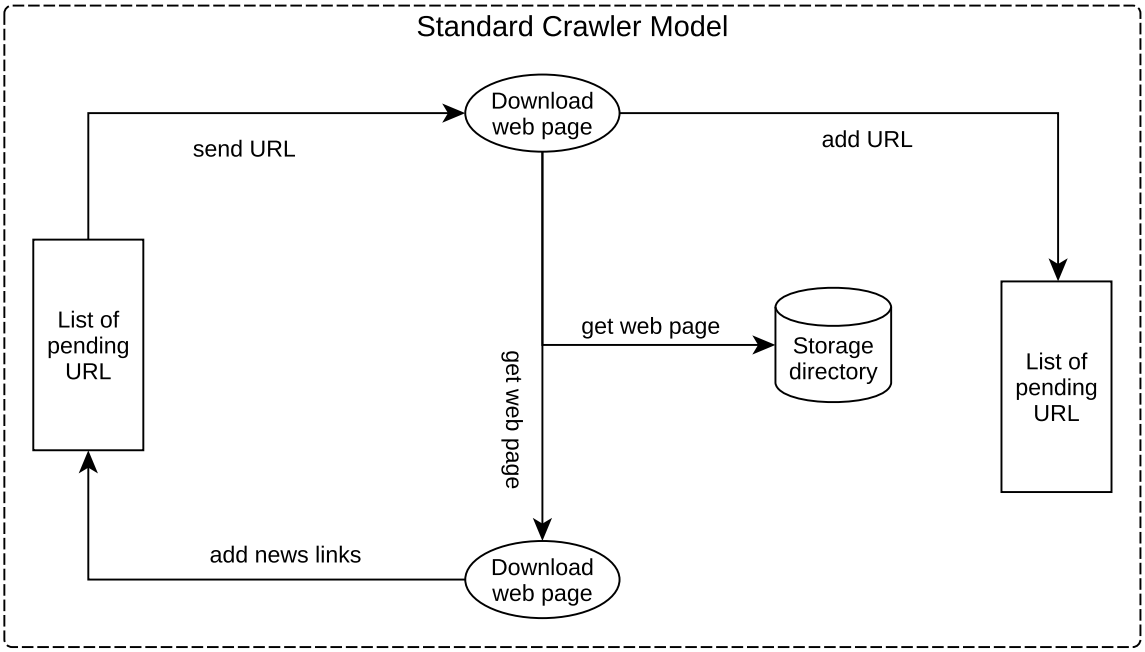

Fig. 3.1 Modelo de Crawler Estándar.

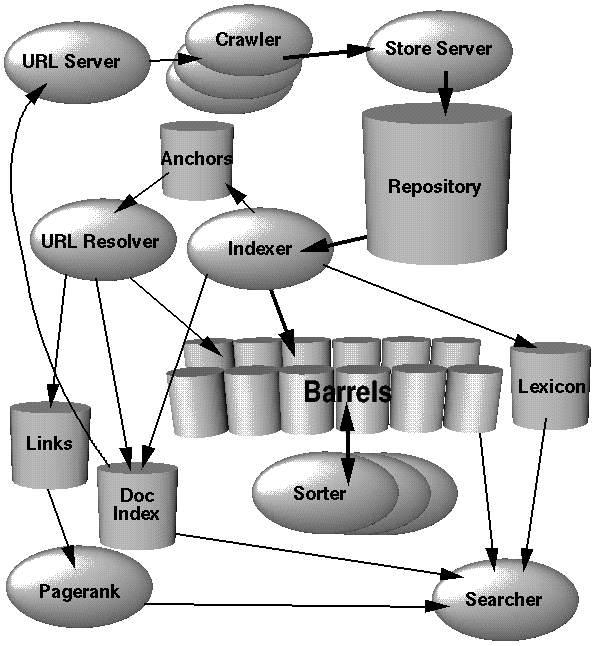

Fig. 3.2 Arquitectura de alto nivel Google.

número de identificación asociada llamada docID que se asigna cada vez que una nueva URL se analiza fuera de una página web. La función de indexación se realiza por el indexador (indexer) y el clasificador (sorter). El indexador realiza una serie de funciones. Se lee el repositorio, descomprime los documentos y los analiza. Cada documento se convierte en un conjunto de ocurrencias de palabras llamado hits (éxitos). Los hits registran la palabra, la posición en el documento, una aproximación del tamaño de la fuente, y mayúsculas. El indexador distribuye estos hits en un conjunto de "barrels" (barriles), creando un índice avanzado parcialmente ordenado. El indexador realiza otra función importante. Se analizan todos los links (enlaces) en cada página web y almacenes de información importante acerca de ellos en un archivo de anclas (anchors file). Este archivo contiene información suficiente para determinar desde donde y hacia que punto apunta el link, y el texto del enlace. El resolucionador de URL (URL resolver) lee el archivo de anclas y convierte las URL relativas en URL absolutas y a su vez en docIDs. Pone el texto de anclas adelante del índice, asociado con el docID al que apunta el ancla. También genera una base de datos de enlaces que son pares de docIDs. La base de datos de links se usa para calcular PageRanks para todos los documentos. El clasificador toma los barriles, lor requiere por wordID y se ordenan por docID, para generar el índice invertido. Esto se realiza en el mismo lugar, por lo que se necesita poco espacio temporal para esta operación. El clasificador también produce una lista de wordIDs y desplazamientos en el índice invertido. Un programa llamado DumpLexicon toma esta lista junto con el léxico producido por el indexador y genera un nuevo léxico para ser utilizado por el buscador. El buscador está gestionado por un servidor web y utiliza el léxico construido por DumpLexicon junto con el índice invertido y los PageRanks para responder a consultas [18].

3.2 Recuperación de Información

La Recuperación de Información consiste en buscar material (generalmente documentos) de naturaleza no estructurada (normalmente texto) que satisface una necesidad de información de grandes colecciones normalmente almacenadas en computadoras [9]. IR es la investigación que se ocupa de desarrollar algoritmos y modelos para recuperar información de repositorios de documentos. La investigación de IR tradicional se ocupa del texto, aunque la recuperación del habla, las imágenes y el video se están volviendo cada vez más comunes [19].

El objetivo de la IR dada una consulta, es devolver la lista de documentos más relevantes. La evaluación IR hace uso frecuente de las nociones de *precision* y *recall* como se observa en la Figura 3.3, su uso se ha traspasado al trabajo de evaluación de modelos de PLN estadísticos.

Precision es el cociente de la intersección del número de documentos relevantes y el número de documentos recuperados, sobre el número de documentos recuperados, como se ilustra en la Ecuación 3.1.

$$Precision = \frac{A \cap B}{B} \tag{3.1}$$

Recall es el cociente que expresa la intersección de documentos relevantes y los documentos recuperados, sobre el número de documentos que son relevantes (observe la Ecuación 3.2).

$$Recall = \frac{A \cap B}{A} \tag{3.2}$$

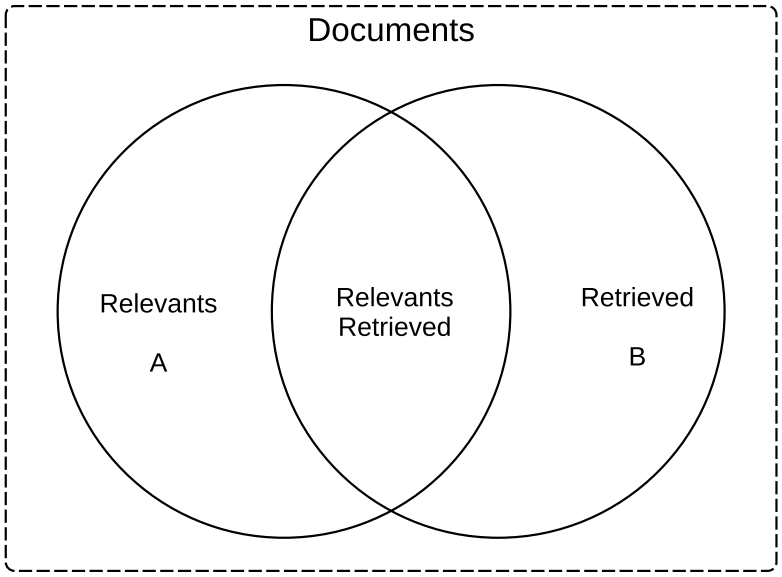

Fig. 3.3 Descripción gráfica de precision y recall para evaluar la IR.

3.2.1 Modelos de Recuperación de Información

Existen diversos modelos que se emplean en la IR; a continuación se describen los modelos que se utilizan en este IRS, el Modelo Booleano y el Modelo de Espacio Vectorial.

Modelo Booleano

Es un modelo de IR en el que se puede plantear cualquier consulta que esté en forma de una expresión booleana de términos, es decir, en la que los términos se combinan con los operadores AND, OR y NOT. El modelo ve cada documento como un conjunto de palabras. Además este modelo se considera como un sistema de coincidencia exacta, que devuelve documentos que satisfacen con precisión las expresiones mencionadas anteriormente, y siguen siendo ampliamente utilizadas en sistemas de información comercial. Sin embargo, para conjuntos de documentos grandes y heterogéneos, los conjuntos de resultados de sistemas de concordancia exacta suelen ser vacíos o enormes y difíciles de manejar.

Modelo de Ranking de Documentos

En este modelo se utilizó la técnica de Recuperación de Información por Ranking (o Rankeo) de Documentos. A diferencia del Modelo Booleano en el que se obtiene un conjunto de documentos que satisfacen una expresión de consulta, en los modelos de Recuperación Rankeados devuelven un orden sobre los documentos (superiores) de la colección con respecto a una consulta. Una de las ventajas de este modelo, son las consultas de texto libre, en lugar de una consulta de operadores y expresiones, la consulta del usuario esta dada en una o más palabras en lenguaje humano. Para llevar a cabo esta tarea se describen las siguientes definiciones, que son de vital importancia para este modelo:

- **Frecuencia de Término**. La Frecuencia de Termino ($TF_{t,d}$: Term Frequency), es el número de veces que el término t aparece en el documento d.
- **Frecuencia de Documento**. La Frecuencia de Documento (DF_t: Document Frequency), es el número de documentos que contienen al término t.
- **Frecuencia Invertida del Documento**. Los términos raros son más informativos que los términos frecuentes. Si se considera un término en la consulta que es raro en la colección de documentos, el documento que contenga a este término es muy probable que sea más relevante para la consulta, pero no es un indicador de relevancia seguro. Para términos

frecuentes, se desean **pesos** positivos altos, pero **pesos** bajos para los términos raros, se usa DF_t para calcular el **Peso** de la Frecuencia Inversa del Documento (IDF: Inverse Frequency Document) se deben contemplar dos puntos:

- DF_t es una medida inversa de la informatividad de t
- $DF_t <= N$, dónde N: Número de documentos

Entonces, IDF_t se define como:

$$IDF_t = log_{10}(N/DF_t) \tag{3.3}$$

Se utiliza el cálculo de log_{10} para amortiguar el efecto del peso de IDF_t. Además cuando se realizan consultas de un único término el IDF_t no tiene efecto en el ranking de consultas, por lo que el IDF_t afecta la clasificación de documentos para consultas con al menos dos términos en la consulta.

- **Frecuencia de Colección**. La Frecuencia de Colección (CF_t: Collection Frequency) del término t, es el número de ocurrencias de t en la Colección.
- **TF-IDF**. TF-$IDF_{t,d}$ (Term Frequency - Inverse Frequency Document) es una medida para producir un peso compuesto para cada término t en cada documento d. Este peso es una medida estadística utilizada para evaluar la importancia de una palabra para un documento en una colección o corpus. TF-$IDF_{t,d}$ asigna un peso t en un documento d que es:
  - Más alto cuando t ocurre muchas veces dentro de un pequeño número de documentos (otorgando de esta forma un alto poder discriminatorio para esos documentos).
  - Menor cuando el término t ocurre menos veces en un documento d, u ocurre en muchos documentos (ofreciendo así una señal de relevancia menos pronunciada).
  - Más bajo cuando el término t se produce en prácticamente todos los documentos.

  El peso de TF-$IDF_{t,d}$ esta dado por la formula siguiente:

$$TF - IDF_{t,d} = (1 + log(TF_{t,d})) * (log_{10}(N/DF_t)) \tag{3.4}$$

  Al peso TF-$IDF_{t,d}$ alternativamente también se le nombra como: TF.IDF, TF*IDF. Aunque es más conocido en la IR como Esquema de Ponderación.

- **Puntuación (Score)**. Cada documento se puede ver como un vector con un componente correspondiente a cada término en el diccionario, junto con un peso (TF-IDF$_{t,d}$) para cada componente, para los términos del diccionario que no se encuentran en un documento, su peso es cero. Está forma de vector es crucial para calcular la puntuación (score) y el rankeo de documentos. Así se tiene la siguiente fórmula que es la medida de puntuación para el traslape (Overlap Measure Score) de documentos:

$$Score(q,d) = \sum_{t \varepsilon q \cap d} TF.IDF_{t,d} \tag{3.5}$$

  El Score de un documento d es la suma sobre todos los términos de q, de la cantidad de veces que cada uno de los términos de la consulta se produce en d. De esta forma sólo se suma el peso TF-IDF$_{t,d}$ de cada término t en d.

Modelo de Espacio Vectorial

El modelo de espacio vectorial es uno de los modelos más utilizados para la IR, principalmente debido a su simplicidad conceptual, y la metáfora subyacente de usar la proximidad espacial para la simplitud semántica. Los documentos y las consultas se representan en un espacio de alta dimensión, en el que cada dimensión del espacio corresponde a una palabra en la colección de documentos. Se espera que los documentos más relevantes para una consulta sean los representados por los vectores más cercanos a la consulta, es decir, los documentos que usan palabras similares a la consulta.

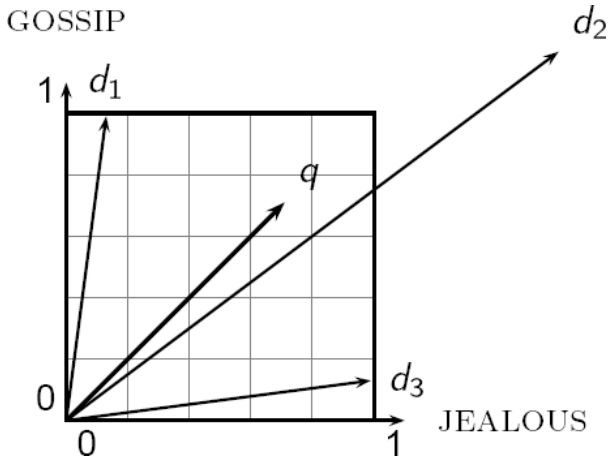

Fig. 3.4 Distancia Euclidiana entre consulta y documentos.

En lugar de considerar la magnitud de los vectores utilizando la distancia Euclidiana 3.4, la cercanía a menudo se calcula simplemente observando los ángulos véase la Figura 3.5, de esta forma los documentos de magnitud larga y corta ahora tienen pesos comparables, eligiendo documentos que encierran el ángulo más pequeño con el vector de consulta, cuando el ángulo entre dos documentos es cero, se tiene una similitud máxima. Para calcular esto se utiliza:

$$CosineSimilarity(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}} \tag{3.6}$$

De la Ecuación 3.6 se tiene:
q_i es el peso TF-IDF del término i en la consulta.
d_i es el peso TF-IDF del término i en el documento.
$|V|$ es la cardinalidad del Vocabulario

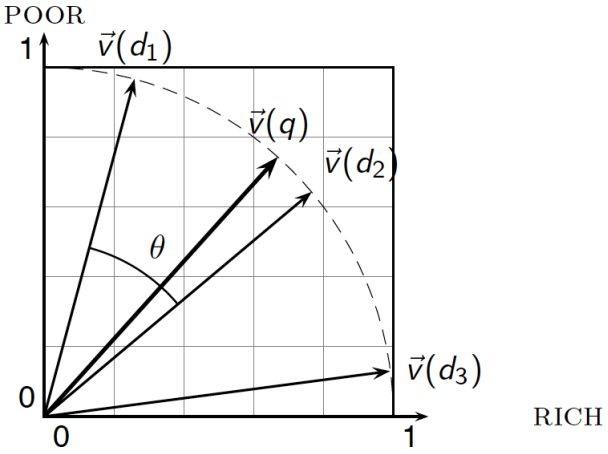

Fig. 3.5 Similitud Coseno utilizando ángulos de vectores.

Sin embargo lo que se desea es normalizar la longitud de los vectores, esto se realiza de una forma simple calculando el producto punto (o producto escalar), véase la Ecuación 3.7.

$$SmilarityCosine(\vec{q},\vec{d}) = \vec{q}\bullet\vec{d} = \sum_{i=1}^{|V|} q_i d_i \tag{3.7}$$

Un elemento más que es utilizado en el cálculo de la Similitud Coseno, consiste en la Norma L_2, que consiste en sumar cada elemento al cuadrado del vector y obtener la raíz cuadrada de esta suma, como se puede observar en la Ecuación 3.8.

$$||\vec{x}||_2 = \sqrt{\sum_i x_i^2} \tag{3.8}$$

Con todas estas técnicas lo que prosigue es calcular el Puntaje y Ranking de Documentos, con la ayuda del Algoritmo 1.

Algoritmo 1 Algoritmo para calcular el Score Coseno.

```
1:  procedure CosineScore(q)
2:      float Scores[N] = 0
3:      float Length[N]
4:      for each query term t do
5:          calculate w_{t,q} and fetch postings list for t
6:          for each pair(d, tf_{t,d}) in posting list do
7:              Scores[d] += w_{t,d} * w_{t,q}
8:          end for
9:      end for
10:     Read the array Length
11:     for each d do
12:         Scores[d] = Scores[d] / Length[d]
13:     end for
14:     return Top K components of Scores[]
15: end procedure
```

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Fig. 3.6 Variantes del Peso TF-IDF.

El peso TF-IDF tiene muchas variantes para calcular el TF, DF y la normalización, como se ilustra en la Figura 3.6. Las letras al inicio de las columnas son llamadas acrónimos y se usan para denotar combinaciones de estas variantes, la forma para denotarlas esta dada por ***ddd.qqq***, la letra *d* corresponde a los documentos y la *q* a la consulta. Por ejemplo lnc.ltc, al observarse la Figura 3.6 la primera letra *"l"* denota logarithm para el TF, *"n"* indica que no se contempla IDF y la tercera letra *"c"* es la Normalización Coseno, esto para los documentos. Para la consulta se utilizara *"l"* logarithm, *"t"* IDF y *"c"* Normalización Coseno. De este modo se pueden construir diferentes combinaciones entre las tres columnas.

3.3 TextRank

TextRank es un modelo basado en grafos de ranking para el procesamiento de texto. Estos algoritmos son esencialmente una forma de decidir la importancia de un vértice dentro de un grafo, basado en la información global extraída recursivamente de todo el grafo. La idea básica de utilizar un modelo de este tipo es la de "votar" o "recomendar", es decir; cuando un vértice se vincula a otro, básicamente le está asignando un voto a ese otro vértice. Además, la importancia del vértice que emite la votación determina la importancia del voto en sí mismo, y esta información también se tiene en cuenta en el modelo. Por lo tanto, la puntuación asociada con un vértice se determina sobre la base de los votos que se emiten para él, y la puntuación de los vértices emitiendo esos votos [20].

Una aplicación muy utilizada de este modelo, es para la obtención de un resumen automático sobre un texto determinado. En la Figura 3.7 se muestra el funcionamiento general. A partir de un texto se extraen las oraciones y sus respectivas palabras. Se crea un grafo, para cada palabra de la oración se crea un vértice, y se crea un arista (vincula) con cada una de las demás palabras (vértices) de la misma oración. Cuando el vértice (palabra) ya existe en el grafo, únicamente se vincula a ese vértice. Estos pasos se repiten para todas las oraciones del texto.

Una vez que se tiene el grafo completo, se aplica el algoritmo PageRank, que se utiliza para medir la importancia relativa de las páginas web, con un método para calcular un ranking para cada página del grafo web [21]. Como resultado de aplicar PageRank se obtiene una lista de palabras relevantes. Se aplica un algoritmo de Correlación entre las palabras de la oración, con cada una de las palabras relevantes, esto para todas las oraciones del texto. El resultado es una lista de oraciones en orden descendente, con respecto a su valor de correlación por oración.

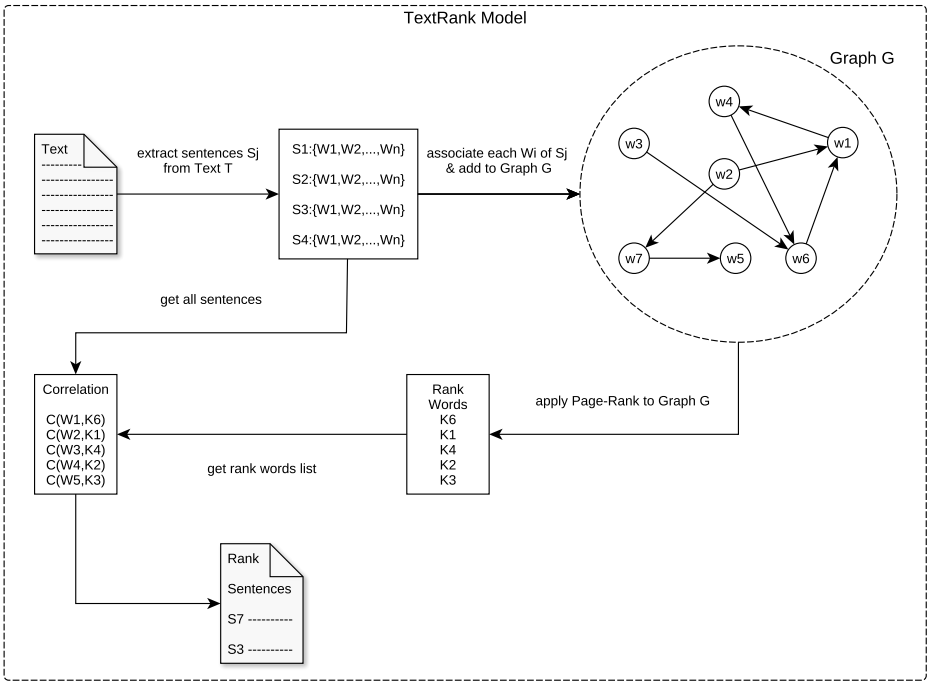

Fig. 3.7 Modelo general de TextRank.

3.4 Punto de Transición

Es una regularidad estadística que proviene de la tensión entre dos fuerzas inherentes a los lenguajes naturales: unificación y diversificación. La primera conduce a emplear términos de índole general, mientras que la segunda al uso de términos específicos. Los términos ligados a la primera fuerza establecen nexos con el entorno del texto, y los de la segunda detallan su contenido. Esto sugiere que las palabras que caracterizan un texto no sean ni las mas frecuentes ni las menos frecuentes, sino las que se encuentran en una frecuencia media de ocurrencia dentro del texto. La fórmula 3.9 se derivó para localizar la frecuencia que divide en dos al vocabulario de un texto: las palabras de baja, y alta frecuencia; justamente, el llamado punto de transición [22]. Dónde I, es la cantidad de palabras con frecuencia 1.

$$PT = \frac{\sqrt{1+8\times I_1}}{2} \tag{3.9}$$

3.5 Reconocimiento de Voz Automático

Un Reconocedor de Voz Automático (ASR: Automatic Speech Recognition) es una aplicación de la inteligencia artificial, cuyo objetivo principal es servir como herramienta computacional que permite la comunicación entre humanos y máquinas, es decir, es capaz de procesar la voz emitida por el humano y reconocer la información que se emite, convirtiéndola en datos de tipo texto. Se utilizan modelos de lenguaje como fuente de conocimiento sintáctico, semántico y pragmático que dan lugar al modelo de lenguaje del sistema. Actualmente existen diversos ASR, aunque la mayoría son comerciales y sólo contemplan acceso libre por un tiempo limitado a sus respectivas API, existen otros como CMU Sphinx y Julius que son de código abierto. A continuación se listan algunos ASR:

- Google Speech API
  https://cloud.google.com/speech/
- IBM Speech to Text
  http://www.ibm.com/watson/developercloud/speech-to-text.html
- Microsoft Bing Voice Recognition
  https://www.microsoft.com/cognitive-services/en-us/speech-api
- Houndify API
  https://www.houndify.com/
- Wit.ai
  https://wit.ai/
- CMU Sphinx
  http://cmusphinx.sourceforge.net/wiki/
- Julius
  https://github.com/julius-speech/julius
- Speech Recognition de la Web Speech API
  https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

3.6 Sintetizador de Voz

A diferencia del ASR la función del Text to Speech (TTS) es la producción artificial del habla. El motor TTS está compuesto de un back-end y un front-end. El front-end se hace cargo del preprocesamiento y normalización de texto, para después hacer una transcripción fonética a cada palabra, marca y divide el texto en unidades prosódicas, como frases, cláusulas y oraciones. El back-end es el sintetizador que convierte la representación simbólica lingüística del front-end, además se encarga de la intención prosódica (tono del perfil, duración de los fonemas, etc.) implementando la voz de salida. Los principales TTS en el mercado son:

- Síntesis de voz de Google incluida en Google Now
- Siri de iOS
- Sintetizador de Voz del LKE
- The Festival Speech Synthesis System
  http://www.cstr.ed.ac.uk/projects/festival/
- Speech synthesis de la Web Speech API
  https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis

3.7 Sistema Operativo de Robot

El Sistema Operativo de Robot (ROS: Robot Operating System) es un framework (entorno de trabajo o marco de trabajo) flexible para escribir software de robot. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto a través de una amplia variedad de plataformas robóticas. Los componentes principales de ROS son:

- Infraestructura de comunicaciones. ROS ofrece una interfaz de paso de mensajes que proporciona comunicación entre procesos y se conoce comúnmente como un middleware. El middleware ROS ofrece estas instalaciones:
  - Publicación / Suscripción de paso de mensajes anónimos
  - Grabación / Reproducción de mensajes
  - Petición / Respuesta de llamadas a procedimientos remotos

  - Sistema de parámetros distribuidos

- Características específicas de robot. Además de los componentes básicos de middleware, ROS proporciona bibliotecas comunes específicas de robot y herramientas que harán que el funcionamiento del robot sea más rápido. Algunas capacidades son:
  - Definiciones de mensajes estándar para robots
  - Biblioteca Geométrica de Robot
  - Lenguaje de Descripción de Robot
  - Anticipar el llamado a procedimientos remotos
  - Diagnóstico
  - Hacer estimaciones
  - Localización
  - Cartografía
  - Navegación
- Herramientas. Uno de los rasgos más característicos de ROS es el potente conjunto de herramientas de desarrollo (de línea de comandos, rviz y rqt). Estas herramientas soportan la introspección, depuración, trazado, y visualizar el estado del sistema en desarrollo.

3.8 API RESTful

Un API RESTful es una interfaz de aplicación de Software que utiliza solicitudes HTTP GET, PUT, POST y DELETE. También es conocida como un *servicio web* se basa en la tecnología de Transferencia de Estado Representacional (RES: Representational State Transfer), un estilo arquitectónico y el enfoque de comunicaciones de uso frecuente en los servicios de desarrollo web. La tecnología REST es generalmente preferida en lugar de SOAP (Simple Object Access Protocol) porque REST aprovecha menos ancho de banda, esto la hace más adecuada para su uso en Internet.

Una API RESTful para un sitio web es el código que permite que dos programas de Software puedan comunicarse entre sí, son utilizadas por sitios tales como Amazon, Google, Linkedin y Twitter. Un API RESTful explícitamente toma ventaja de metodologías HTTP definidos por la

RFC-2016[1]. Utilizan GET para recuperar un recurso, PUT para cambiar el estado de un recurso o actualizarlo, puede ser un objeto, archivo o bloque. POST para crear un recurso y DELETE para eliminarlo.

[1] https://tools.ietf.org/html/rfc2616

Capítulo 4

IRS como mecanismo de HRI

En el Sistema de Recuperación de Información (IRS) como mecanismo de Interacción Humano-Robot (HRI) se definen los modelos propuestos para cada una de las secciones. La primera etapa consiste en la descarga de documentos con el Crawler desarrollado para este propósito. En la siguiente etapa, se define el Motor de Recuperación de Información (IRE), donde de los documentos recuperados en HTML (HyperText Markup Language) se extrae el texto. Además se crean los algoritmos para identificar el idioma, preprocesamiento y normalización de los documentos. Se presenta el Modelo de Recuperación, que es un Modelo de Espacio Vectorial que utiliza el Ranking de Documentos, además se definen las herramientas y algoritmos que serán utilizados. En esta etapa también se crea un API RESTful para proporcionar un servicio web a los usuarios accediendo al IRS desde cualquier lugar, utilizando autenticación de usuarios y se crea un Sistema Web para el IRS. Para finalizar en las etapas de Reconocimiento de Voz Automático (ASR) y del Sintetizador de Voz (TTS), se describen las herramientas y algoritmos usados en su implementación. Además se identifican los mecanismo del Sistema Operativo de Robot (ROS: Robot Operating System) para la creación de algoritmos para Speech Recognition y TTS.

El modelo general propuesto para este trabajo de tesis se muestra en la Figura 4.1. Se resalta de color verde las solicitudes de consulta del usuario, y de azul la respuesta a dichas solicitudes. Este modelo cuenta con los siguientes módulos, *Crawler*: encargado de la recopilación de documentos del la web. *Motor de Recuperación de Información* (IRE): se encarga del preprocesamiento e indexado de los documentos en un Corpus de índices invertidos, y de la implementación de los algoritmos recuperación de documentos utilizando un Modelo Booleano con Ranking de documentos. *Reconocedor de Voz Automático* (ASR) reconoce el audio y lo

convierte a texto. *Sintetizador de Voz* (TTS): convierte el texto de un documento recuperado y lo transforma en audio.

4.1 Crawler

El Crawler utilizado en este trabajo se creó de forma similar a los tradicionales, tomando como base un conjunto de URL semillas almacenadas en una Base de Datos (DB: Data Base). Después se toma una URL, al visitarla lo primero es analizar el archivo robots.txt del dominio de la URL, para verificar si la URL se puede visitar, si es así, se visita la URL. Posteriormente, se descarga y analiza el tipo de documento verificando su mime-type, entonces se actualizan los datos de la URL: visitada, estatus, su tipo de documento y su nombre de archivo y ruta de almacenamiento físico en la DB. El almacenamiento físico esta organizado por directorios, cada directorio es etiquetado por fecha, y a su vez cada directorio contiene los documentos descargados durante ese día. Sí el documento descargado es HTML, se extraen los links contenidos en el documento, los links relativos se transforman en absolutos como se observa en la Tabla 4.1, para después añadir los nuevos links a la DB. En caso de no tener permitido visitar la URL, se actualizan los datos en la DB y continúa con la siguiente URL. Este procedimiento se repite para cada URL en la DB. La Figura 4.2 muestra de forma gráfica su funcionamiento.

Tabla 4.1 Ejemplo de conversión de links relativos a absolutos.

Links relativos	**Links absolutos**
index.html	http://www.mysite.com/
index.php	http://www.mysite.com/
home/logo.png	http://www.mysite.com/home/logo.png
media/graphics/image.jpg	http://www.mysite.com/media/graphics/image.jpg
info.php?query=2323	http://www.mysite.com/info.php?query=2323

Para el funcionamiento del Crawler se usa el Algoritmo 2 que se encarga de obtener links de la DB, realiza peticiones para la visita de la página web, de ser positiva la respuesta se descarga el documento, para esto se utiliza el Algoritmo 3 de Web Scraping. Después de descargar el documento y almacenarlo físicamente, se verifica si es un documento HTML, de ser así con el Algoritmo 4 se extraen todos los links que contenga, convirtiendo los links relativos en absolutos (si los hay). Estas lista de links es insertada a la DB para su posterior análisis. Este proceso se realiza de forma concurrente, de este modo se pueden definir el número de procesos (hilos en este caso) para el funcionamiento del Crawler.

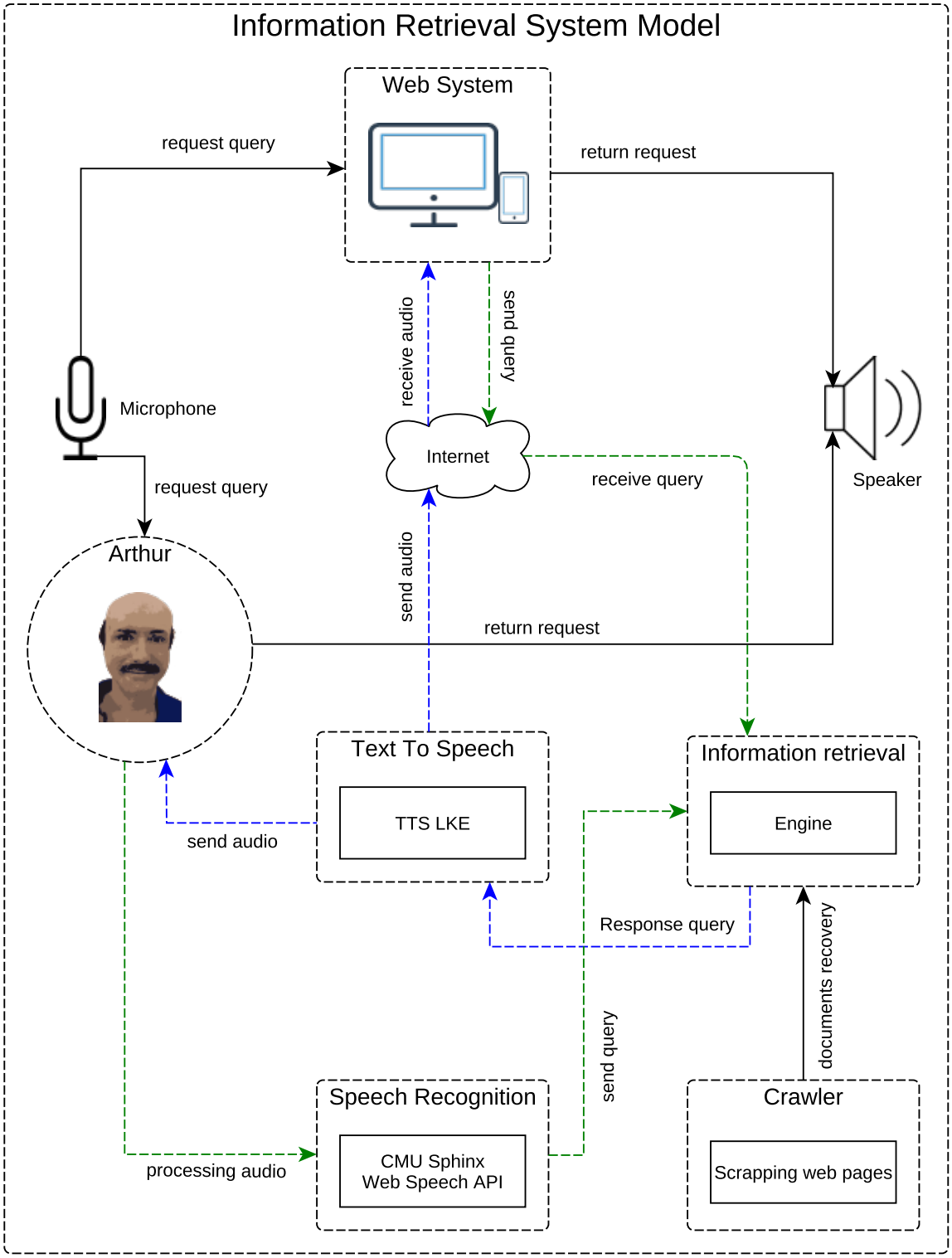

Fig. 4.1 Modelo propuesto.

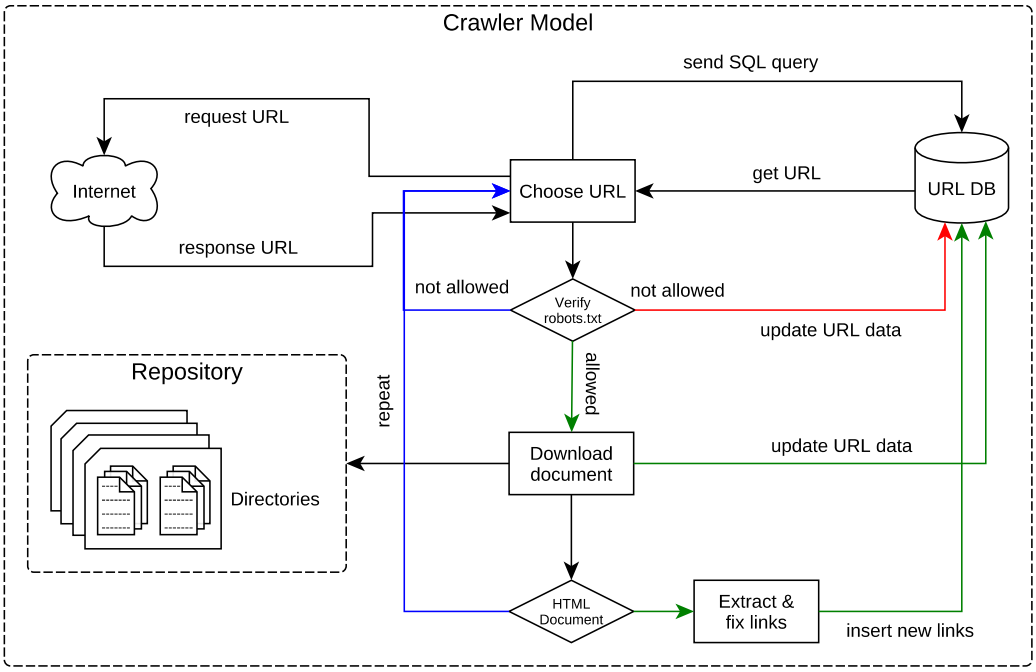

Fig. 4.2 Modelo del Crawler.

4.2 Motor de recuperación de información

El motor de recuperación de información (IRE: Information Retrieval Engine) se encarga de utilizar los modelos y las técnicas de la Recuperación de información (IR). Para la Recuperación de Documentos se usó un Modelo de Ranking como base del Modelo de Espacio Vectorial.

El IRE se muestra en la Figura 4.3 dónde se encuentra el Ranking de Documentos utilizando el corpus de Índices Invertidos y el corpus original, además de las entradas (Speech Recognition) y salidas (Text To Speech) al Sistema. De manera inicial se carga en la memoria RAM (Random Access Memory) al corpus de índices invertidos, en una estructura de datos tipo diccionario como se muestra en la Tabla 4.4. Se procesan las consultas del usuario utilizando reconocimiento de voz, de Sphinx (CMU Sphinx) o desde el navegador (Web Speech API), se realiza un preprocesamiento de la consulta y se buscan los términos (palabras) en los postings que previamente se han cargado en memoria. Después se procesa la consulta en el Modelo de Espacio Vectorial para obtener un Ranking de documentos, como resultado de la búsqueda se obtienen tuplas de los documentos recuperados, estas tuplas se conforman del DocID, seguido de una lista de posiciones del término buscado dentro del documento y el Score (puntaje) del

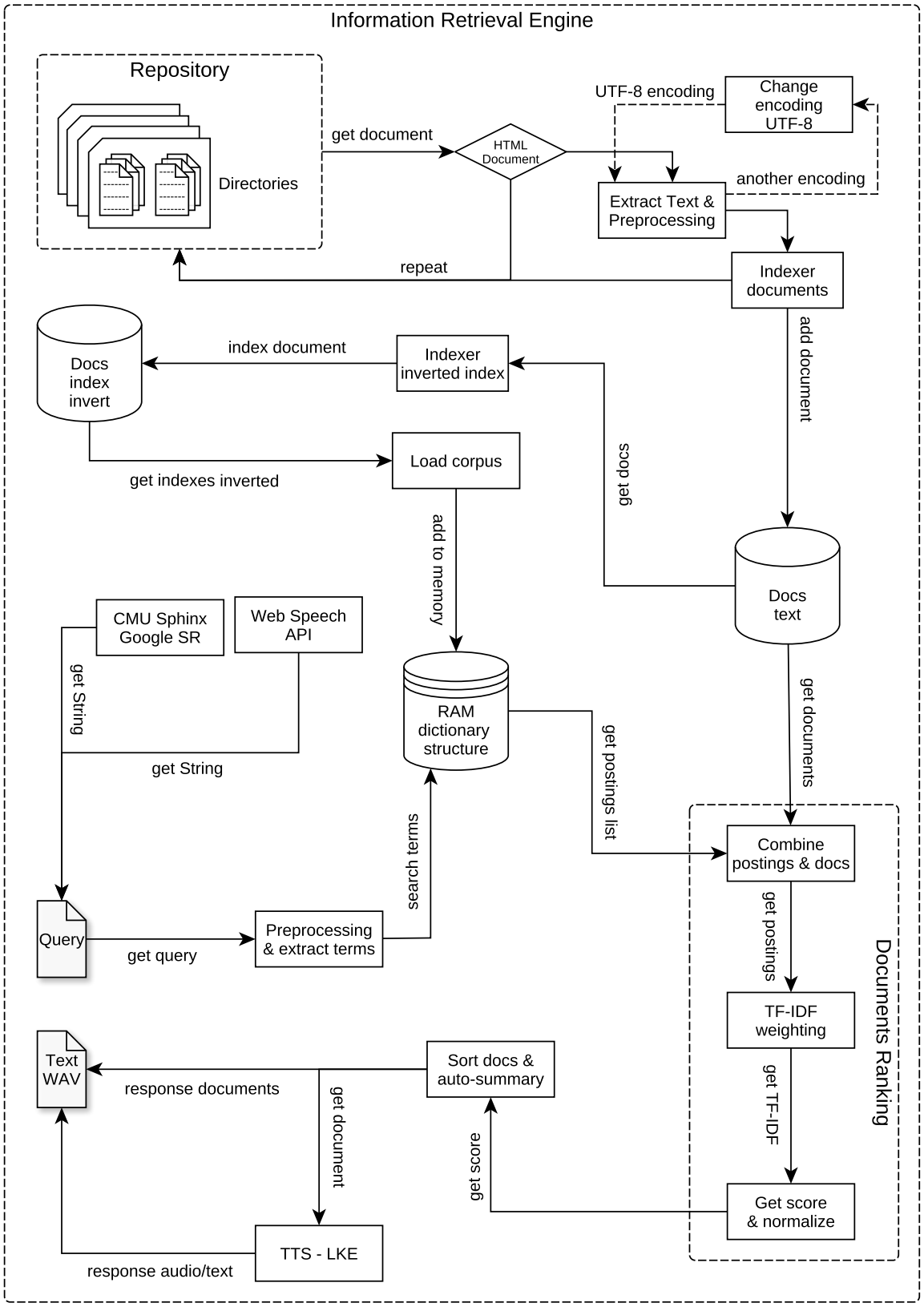

Fig. 4.3 IRE: Motor de Recuperación de Información.

documento. Del top de la lista de resultados se toma el docID del documento y se extrae el texto del corpus Original, para posteriormente extraer un breve resumen del documento recuperado, se pasa al TTS-LKE o Festival para convertirlo en audio y se devuelve al usuario junto con el resumen de texto.

4.2.1 Preprocesamiento

Del repositorio de directorios, se seleccionan los documentos HTML. En primera instancia se verifica la codificación del documento a procesar. De tener una codificación distinta a la UTF-8[1] (8-bit Unicode Transformation Format), se convierte a esta codificación. Se prosigue extrayendo únicamente el texto del documento HTML, después se procede a la limpieza de caracteres y conversión a minúsculas del texto.

4.2.2 Identificación de idioma

El texto limpio se utiliza como test en un modelo Naïve Bayes para identificar el lenguaje del documento, el modelo cuenta con cuatro lenguajes como entrenamiento, véase la Tabla 4.2.

Tabla 4.2 Lenguajes para la detección de idiomas

Lenguaje	**Líneas**
english	2980
italian	2821
french	2980
spanish	2980

Se toma la clase del lenguaje con el valor más alto y se asigna al documento analizado. De esta forma se identifican los documentos recuperados en lenguaje Español.

4.2.3 Indexado de documentos

El texto esta listo para ser indexado en un corpus de documentos, el nombre físico de cada documento corresponde con el ID en la DB, de esta manera se tiene claro el origen del documento. Para indexar el documento se coloca su ID seguido del texto en el mismo renglón. Esta estructura se puede apreciar en la Tabla 4.3. Así para todos los documentos HTML.

[1] Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes web.

Tabla 4.3 Estructura del corpus de documentos html.

Doc. Id	**Texto del documento html**
1567	aptitud física y ejercicio medlineplus en español usted esta...
4546	twitter es lo que esta pasando que esta pasando mas twitter...
2197554	como mejorar la atencion al cliente y que no sea un problema...
2197561	los mejores articulos de mi posicionamiento web en 2016...

Una vez que se ha limpiado el texto, se ha obtenido el idioma y se ha indexado el documento al corpus, se actualiza la base de datos, marcando estas tareas como hechas.

4.2.4 Indexado de índices invertidos posicional

A partir del corpus de documentos se crea un índice invertido posicional. La idea de un índice invertido se muestra en la Tabla 4.4. Se tiene un diccionario (a veces también denominado vocabulario o léxico) de términos (palabras). Así, cada término contiene el total de documentos (DF:Document Frequency) donde se ha encontrado dicho término, además de una lista que registra los documentos en los que se produce el término, a está lista se le denomina convencionalmente como **posting list**. Cada elemento del posting esta formado por el DocID del documento, seguido de la(s) posición(es) que ocupa dentro del mismo documento, los DocID y las posiciones del documento están ordenados de forma ascendente.

Tabla 4.4 Índice invertido posicional.

Diccionario	**Postings**	
Término	DF	DocId:Posición
accidentado	4	**20650**:61,107;**22313**:5,12;**24089**:137;**24105**:539
canek	1	**22622**:13,39,116
simulacros	2	**20514**:54;**20838**:54
gente	5	**10263**:189;**10695**:101;**10722**:62;**10779**:52;**10804**:29
helado	4	**12570**:4802;**12657**:1691,1695,1707,1981;**13686**:929;**14162**:996,1020
subtotal	1	**26846**:203,226,248

4.2.5 Modelo de Espacio Vectorial

En este modelo se utiliza el corpus de documentos original y el corpus de índices invertidos (que se han creado en etapas anteriores). Se crean los algoritmos necesarios para calcular

$Score(q,d)$, dónde q son los términos de la consulta y d los documentos dónde se encuentran esos términos.

En este trabajo se utiliza un motor con la combinación de acrónimos ***lnc.ltc*** de la Figura 3.6. Los tres primeros acrónimos corresponden al Score de documentos, donde se le aplica *logaritmo* al TF, *no* se aplica ninguna operación al DF se toma como 1 y se utiliza una *normalización* coseno. Para los acrónimos de la consulta, se utiliza *logaritmo* para el TF, para el DF se utiliza *IDF* y una *normalización* coseno.

Cuando se recibe la consulta, se calcula el *Score* de cada término de la consulta y de los documentos donde aparecen estos terminos utilizando la Ecuación 3.5 y almacenando los datos en vectores. La representación de un conjunto de documentos como vectores en un espacio de vector común se conoce como modelo de espacio vectorial.

Tabla 4.5 Score de los términos de la consulta de ejemplo.

Term	**TF-RAW**	**TF-W**	**DF**	**IDF**	**WT**	**Normalize**
Hugo	1	1.0	27	2.568	2.568	0.650
Chávez	1	1.0	10	3.0	3.0	0.759

Para describir de forma más amplia este procedimiento, se presenta una consulta de ejemplo: *"Hugo Chávez"*, lo primero es verificar que los términos se encuentren en el corpus. Después se calcula el Score de la consulta y de los documentos donde también aparecen estos terminos. El cálculo del Score para los terminos de la consulta, se muestra en la Tabla 4.5. En esta Tabla la primera columna hace referencia al término *TF-RAW*, que es la frecuencia cruda del término, es decir cuántas veces se repite este termino en la consulta esto es igual a 1, la segunda columna *TF-W* hace referencia al cálculo del TF como se observa en la Ecuación 3.4 y en este caso es $1 + log(TF - RAW) = 1 + 0.0 = 1.0$, la tercera *DF* es el número de veces que aparece el término en los documentos: $\text{DF}_{Hugo} = 27$ y $\text{DF}_{Chávez} = 10$, para *IDF* se calcula con la Ecuación 3.4 que es la Frecuencia Inversa del Documento, es decir el peso del término en los documentos, para este ejemplo el número de documentos es de $N = 10000$, entonces $\text{IDF}_{Hugo} = log_{10}(10000/27) = 2.568$ e $\text{IDF}_{Chávez} = log_{10}(10000/10) = 3.0$. La columna *WT* corresponde a la multiplicación de *TF-W* por *IDF*, y la última columna *Normalize* es la normalización de cada término con la Norma L_2 que se realiza con la división de $IDF/NormaL_2$, para el cálculo de la Norma L_2 se utiliza la Ecuación 3.8 que sería $\sqrt{2.568^2 + 3.0^2} = 3.949$, de este modo $Normalize_{Hugo} = 2.568/3.949 = 0.650$ y para $Normalize_{Chávez} = 3.0/3.949 = 0.759$.

En la Tabla 4.6 se muestra el Score de Documentos para el término *Hugo*, abajo en la misma Tabla se ilustra el Score de Documentos para el término *Chávez*. Las operaciones son similares

Tabla 4.6 Score de Documentos de la consulta ejemplo.

Term	docID	TF-Raw	TF-W	WT	Normalize
Hugo	2370746	1	1.0	1.0	1.0
Hugo	2385677	1	1.0	1.0	1.0
Hugo	2397722	1	1.0	1.0	1.0
Hugo	2398033	3	1.477	1.477	1.0
Hugo	2399093	1	1.0	1.0	1.0
Hugo	2401101	1	1.0	1.0	1.0
Hugo	2401938	2	1.301	1.301	1.0
Hugo	2402069	2	1.301	1.301	1.0
Hugo	2402126	2	1.301	1.301	1.0
Hugo	2402252	1	1.0	1.0	1.0
Hugo	2402266	2	1.301	1.301	1.0
Hugo	2402294	1	1.0	1.0	1.0
Hugo	2406955	1	1.0	1.0	1.0
Hugo	2410809	1	1.0	1.0	1.0
Hugo	2431629	3	1.477	1.477	1.0
Hugo	2606374	1	1.0	1.0	1.0
Hugo	2606977	1	1.0	1.0	1.0
Hugo	2693678	1	1.0	1.0	1.0
Hugo	3445898	1	1.0	1.0	1.0
Hugo	3446983	1	1.0	1.0	1.0
Hugo	3806869	1	1.0	1.0	1.0
Hugo	4084692	1	1.0	1.0	1.0
Hugo	4088737	1	1.0	1.0	1.0
Hugo	4779911	2	1.301	1.301	0.707
Hugo	5060170	1	1.0	1.0	0.560
Hugo	5064281	1	1.0	1.0	1.0
Hugo	5347284	1	1.0	1.0	1.0
Chávez	2391664	1	1.0	1.0	1.0
Chávez	2399733	2	1.301	1.301	1.0
Chávez	2399786	1	1.0	1.0	1.0
Chávez	2399816	1	1.0	1.0	1.0
Chávez	2761614	2	1.301	1.301	1.0
Chávez	3442389	2	1.301	1.301	1.0
Chávez	4261075	2	1.301	1.301	1.0
Chávez	4779911	2	1.301	1.301	0.707
Chávez	5060170	1	1.0	1.0	0.560
Chávez	5349324	1	1.0	1.0	1.0

a las realizadas en la Tabla 4.5, las variaciones en este caso tienen que ver con el los acrónimos ***lnc***, es decir el valor de la columna con el encabezado *WT* es la multiplicación de *TF-W* por *IDF* que según la Figura 3.6 de Variantes de Pesos TF-IDF, el valor de *IDF* para este caso es de 1, por lo que $WT = TF - W * 1$.

La última etapa consiste en realizar el producto punto de la Ecuación 3.7 de cada elemento *Normalize* de la consulta y el vector *Normalize* de los documentos. Del ejemplo se realiza el producto de $Normalize_{query,Hugo} = 0.650$ con todos los elementos de $Normalize_{document,Hugo}$ y con $Normalize_{document,Chávez}$. Los resultados se almacenan en una lista, omitiendo aquellos resultados duplicados, es decir los resultados que se hayan calculado y almacenado de una operación previa.

Al finalizar la operación de producto punto se ordena la lista en forma descendente con respecto al Score, véase la Tabla 4.7. Los elementos en el top de la lista resultan ser los más relevantes, según el método propuesto y están listos para devolverse al usuario en forma de documentos, para ello se utiliza el corpus de documentos original, este corpus contiene el docID de cada documento, por lo que de una manera simple se pueden recuperar los documentos mejor rankeados por el algoritmo y se presenten al usuario como resultado final.

De los encabezados de la Tabla 4.7, el primero hace referencia al *docID* del documento, la columna *Positions* se refiere a la posición dentro del texto de los terminos de la consulta *"Hugo"* y *"Chávez"*, la tercera columna muestra el *Score* (puntaje) alcanzado por cada documento para la consulta solicitada. La forma de proceder para todas las consultas sigue este procedimiento basado en un Modelo de Espacio Vectorial con Ranking de Documentos utilizando Similitud Coseno.

Tabla 4.7 Resultado de Ejemplo utilizando Similitud Coseno.

docID	**Positions**	**Score**
4779911	5,6,61,62	0.9970131701443772
5060170	160,161,656,678	0.993625421266906
2391664	766	0.759605846405135
2399733	11,72	0.759605846405135
2399786	181	0.759605846405135
2399816	111	0.759605846405135
2761614	276,362	0.759605846405135
3442389	6,62	0.759605846405135
4261075	257,393	0.759605846405135
5349324	157	0.759605846405135
2370746	132	0.6503837006776374
2385677	1,60	0.6503837006776374
2397722	370	0.6503837006776374
2398033	1,67,159,184	0.6503837006776374
2399093	881	0.6503837006776374
2401101	424	0.6503837006776374
2401938	175,195	0.6503837006776374
2402069	182,201	0.6503837006776374
2402126	186,205	0.6503837006776374
2402252	166	0.6503837006776374
2402266	199,218	0.6503837006776374
2402294	190	0.6503837006776374
2406955	854	0.6503837006776374
2410809	368	0.6503837006776374
2431629	1,71,182,247	0.6503837006776374
2606374	216	0.6503837006776374
2606977	1,57	0.6503837006776374
2693678	173	0.6503837006776374
3445898	196	0.6503837006776374
3446983	156	0.6503837006776374
3806869	442	0.6503837006776374
4084692	185	0.6503837006776374
4088737	132	0.6503837006776374
5064281	168	0.6503837006776374
5347284	193	0.6503837006776374

4.3 Reconocimiento de Voz Automático

Las aplicaciones para el reconocimiento de voz (ASR: Automatic Speech Recognition) que se utilizaron en este trabajo son CMU Sphinx, Google Speech Recognition y Web Speech API. Las primeras dos ASR se implementaron en lenguaje Python, y la tercera como su nombre lo dice se utilizó desde el navegador Web en lenguaje JavaScript.

Para la versión de escritorio ASR, se utilizó **SpeechRecognition**[2] que es una librería de Python para realizar reconocimiento de voz, con soporte para varios motores y APIs en línea y fuera de línea. A continuación se describen CMU Sphinx y Google Speech Recognition utilizadas en esta librería.

En la Figura 4.4 se muestra la forma en la que se realiza el proceso de reconocimiento de voz, se inicia cuando el usuario transmite una consulta al micrófono, antes de ser procesado el audio, se establecen las características de dicho reconocimiento de audio, como establecer la frecuencia en 16000 hz, después se escucha el audio durante 1 segundo para calibrar el umbral de energía para los niveles de ruido ambiental, así como colocar el número de segundos (5 en este caso) antes de que se interrumpa la señal del micrófono. Una vez que se ha obtenido una muestra de audio, se puede optar por grabar un archivo de audio en formato wav, para después procesarlo o enviar directamente la muestra para ser procesada. La librería Speech Recognition puede procesar una muestra de audio en dos diferentes APIs una después de otra. En este trabajo se utilizaron ambas de forma separada para obtener el texto.

4.3.1 CMU Sphinx

CMU Sphinx (CMU: Carnegie Mellon University) es un conjunto de bibliotecas de desarrollo para el reconocimiento de voz, y herramientas que se pueden vincular a aplicaciones de habilitación de voz. Las bibliotecas y las aplicaciones desarrolladas pueden usarse tanto para fines de investigación como para fines comerciales. Para este trabajo utilizamos un toolkit (conjunto de herramientas) llamado **Pocketsphinx** se trata de Sphinx para plataformas embebidas.

Pocketsphinx es una biblioteca que depende de otra biblioteca llamada **SphinxBase**, que proporciona una funcionalidad común a través de todos los proyectos CMU Sphinx. Para instalar Pocketsphinx, es necesario instalar tanto Pocketsphinx y Sphinxbase. Es posible utilizar Pocketsphinx tanto en Linux, Windows, MacOS, iPhone y Android.

[2]https://pypi.python.org/pypi/SpeechRecognition/

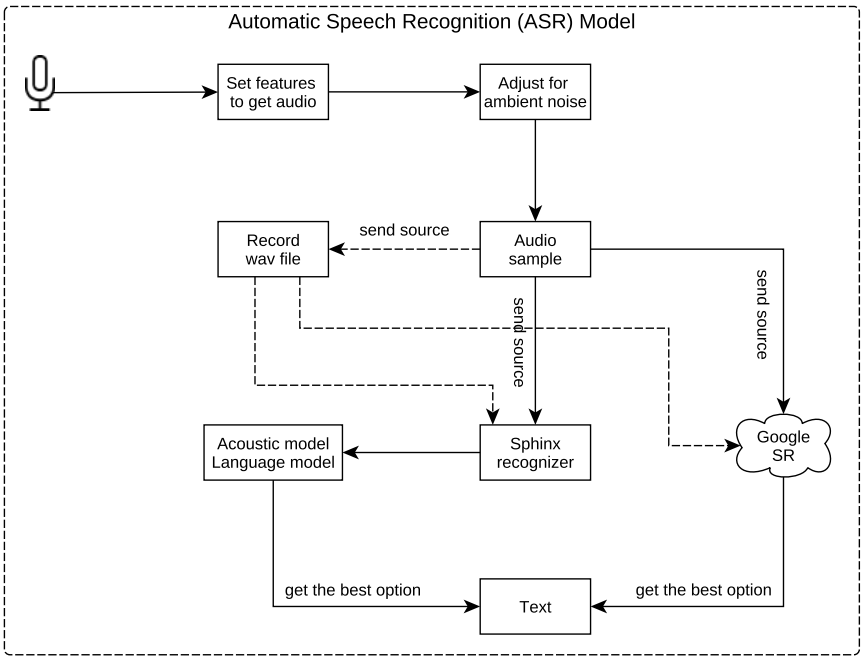

Fig. 4.4 Modelo ASR de la librería Speech Recognition.

4.3.2 Google Speech Recognition

La API de Google Speech Recognition pertenece a la Plataforma de Google Cloud, está permite a los desarrolladores convertir audio en texto mediante la aplicación de modelos de redes neuronales de gran alcance en un formato fácil para usar la API. La API reconoce más de 80 idiomas y variantes, para apoyar a su base de usuarios global. Puede transcribir el texto de dictado por micrófono de una aplicación de los usuarios, activar el control de comandos a través de la voz, o transcribir archivos de audio, entre muchos otras aplicaciones. Reconocer el audio cargado a través de una solicitud web, y se integran con el almacenamiento de audio en Google Cloud Storage, mediante el uso de la misma tecnología que Google utiliza para alimentar sus propios productos.

La API puede transmitir los resultados de texto en tiempo real, devolviendo los resultados del reconocimiento parciales a medida que estén disponibles, con el texto reconocido que aparece inmediatamente mientras se habla. Como alternativa, la API de voz puede devolver texto reconocido mediante un archivo de audio almacenado.

4.3.3 Web Speech Recognition

Esta aplicación se encuentra dentro de la Web Speech API que proporciona la capacidad de reconocer contexto de voz desde la entrada de audio (normalmente a través del servicio de reconocimiento de voz por defecto del dispositivo) y responder apropiadamente. En general el constructor de la interfaz se usa para crear un nuevo objeto *SpeechRecognition*, que tiene una serie de controladores de eventos disponibles para detectar cuando se habla a través del micrófono del dispositivo. Además se cuenta con una aplicación extra, que es la interfaz *SpeechGrammar* que representa un contenedor para un conjunto particular que debe reconocer una aplicación.

4.4 Sintetizador de Voz

EL Sintetizador de Voz (TTS: Text To Speech) del LKE es el que se utilizó, además del TTS de Festival y como tercera opción la interfaz de Síntesis de voz (Speech Synthesis) de la API Web Speech. Los dos primeros TTS funcionan de una manera muy similar, lo único que los distingue es el tono de voz. Actualmente existen diversos TTS, la mayoría son comerciales y algunos como Festival de de código libre. El modelo general de funcionamiento de un TTS se muestra en la Figura 4.5.

4.4.1 TTS LKE y Festival

Festival se ocupa de la investigación en todas las áreas de la tecnología de voz, incluyendo el reconocimiento de voz, síntesis de voz, procesamiento de señal de voz, acceso a la información, interfaces multimodales y sistemas de diálogo. En su framework Festival ofrece el texto completo de la voz (speech) a través de una serie de APIs: a nivel de shell, a través de un interprete de comandos de esquema, como una librería de C++, Java y Emacs. Además es multilingüe, incluyendo español (castellano) aunque inglés es el más avanzado. Festival es de software libre y las herramientas del habla se distribuyen para su uso comercial y no comercial sin restricciones.

Después de instalar Festival, se elige la voz en Español Castellano del catálogo de voces disponibles (véase la Tabla C.1) y se descarga, de este modo se puede utilizar la voz en Español, ya que por default el TTS tiene una voz en Inglés.

De la misma forma que se ejecuta Festival se puede utilizar el TTS LKE, ya que esta basado en Festival con una voz propia, neutra y de tipo masculino. Esta voz fue desarrollada para

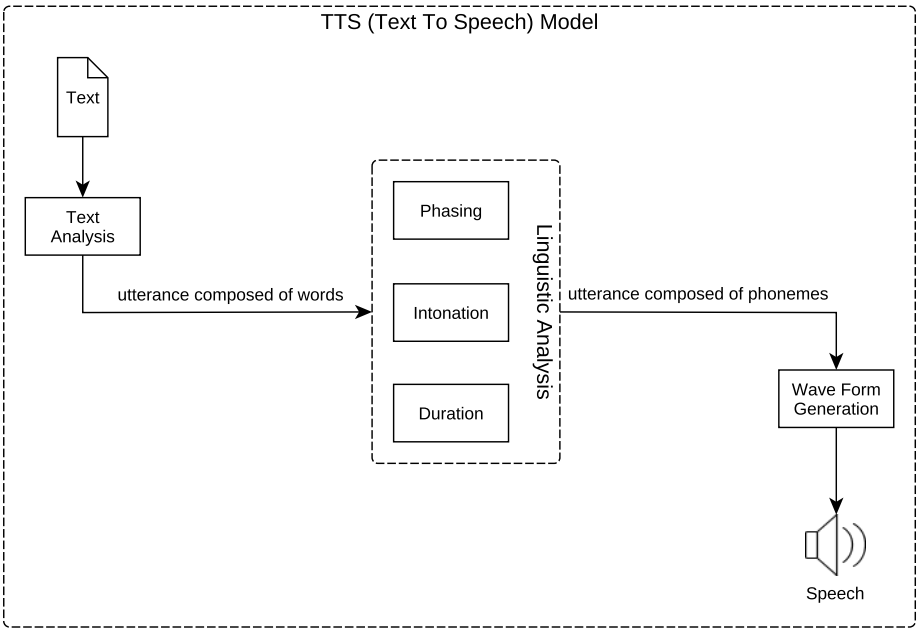

Fig. 4.5 Modelo del Sintetizador de Voz (TTS).

expresarse en Español Mexicano. Los dos TTS contemplan una forma de asignar acentos de una manera simple, anteponiendo la letra acentuada con un apóstrofo simple (e.g. coraz'on) de una forma similar es para la tilde (e.g ni no). En el caso de Festival respeta las pausas utilizadas con signos de puntuación (puntos y comas).

4.4.2 Web Speech Synthesis

La aplicación de síntesis de voz también forma parte de la Web Speech API, se accede a través de la interfaz *SpeechSynthesis*, es un componente (TTS) de conversión de texto a voz que permite leer a los programas su contenido de texto (normalmente a través de un sintetizador de voz por defecto del dispositivo). Los diferentes tipos de voz están representados por objetos *SpeechSynthesisVoice*, y las diferentes partes del texto que se desea que se hablen son representados por objetos *SpeechSynthesisUtterance*. Y se puede obtener estos objetos del habla pasándolos al método *SpeechSynthesis.speak()*.

La Web Speech API es una tecnología experimental, por tal motivo las especificaciones de esta tecnología no se han estabilizado. Las Tablas 4.8 y 4.9 muestran la compatibilidad para el uso en varios navegadores. Está tecnología esta sujeta a cambios en futuras versiones.

Tabla 4.8 Compatibilidad Web Speech API en Navegadores de Escritorio.

Característica	Chrome	Edge	Firefox	IE	Opera	Safari
Soporte Básico	33	Si	49	No soportado		

Tabla 4.9 Compatibilidad Web Speech API en Dispositivos Móviles.

Característica	Android	Chrome	Edge	Firefox	IE	Opera	Safari
Soporte Básico	Desconocido	Si	Si	Desconocido	No soportado		

4.5 Sistema de Recuperación

Al unir las secciones anteriores se forma el Sistema de Recuperación de Información como el que se muestra en la Figura 4.1, para el funcionamiento de esta aplicación se utilizó un Micro-Framework llamado *Flask* para crear la aplicación principal y una *API RESTful* con autenticación de usuario, un servidor de aplicaciones web llamado *uWSGI* y un servidor web llamado *Nginx*, como se puede observar en la Figura 4.6.

La aplicación del lado del servidor esta escrita en Python 3.5 utilizando el paquete *Flask*, cuando se inicia el servidor se carga el Corpus de índices invertidos en memoria. Una vez que se ha completado la carga, el servidor de aplicaciones *uWSGI* crea n-procesos (un número de procesos previamente asignados) de la aplicación, a su vez el servidor web *Nginx* crea el mismo numero de procesos y de esta forma el servidor IRS esta listo para recibir peticiones por medio de la API RESTful.

Para realizar peticiones a la API RESTful previamente se proporcionan nombres de usuario, passwords y la URL de la API a los clientes (usuarios). Se puede acceder a través de diferentes entornos de aplicación por parte del cliente (Python, PHP, Javascript, etc), la petición se realiza en formato JSON[3] (JavaScript Object Notation - Notación de Objetos de JavaScript) al igual que la respuesta. El acceso a la API RESTful esta dentro de la red LAN de la Universidad y puede darse a través de internet habilitando los puertos HTTP para accesar desde cualquier lugar.

En el lado del cliente se puede acceder desde el navegador web o desde un entorno de escritorio, es decir, accesando a la API RESTful desde un programa escrito en un lenguaje de programación (e.g. Python, PHP, Javascript) que soporte peticiones, respuestas y autenticación HTTP. Cuando el usuario acceda desde el navegador web únicamente tiene que accesar al sitio del IRS. En cambio si el usuario desea utilizar el servicio web de la API RESTful en un

[3]http://www.json.org/json-es.html

lenguaje de programación, debe contar con las herramientas y los mecanismos para conectarse y autentificarse, así como realizar peticiones a la API RESTful.

En la Figura 4.6 se puede observar como un cliente (usuario) utiliza el servicio de la API RESTful para realizar peticiones y recibir respuestas del IRS. En el caso de Arthur (el robot humanoide) su funcionamiento esta basado en ROS (Robot Operating System), así de este modo el robot recibe la consulta del usuario por medio del micrófono, después utiliza el módulo *Speech Recognition* para devolver la consulta en texto, que es enviada a través de la *API RESTful* para recuperar la información solicitada en formato JSON, una vez que se tiene el resultado la información se publica a través del modulo *Publisher* de ROS. El módulo *Subscriber* está a la espera de la publicación de información por parte de algún *Publisher*, así cuando el módulo publique información, este la obtiene y utliza el módulo *TTS* para convertir una fracción del texto en voz y esta sea enviada a la salida de audio designada. Además se tiene el módulo *Move Mouth Motors* (Movimiento de Motores de la Boca) de Arthur, este se encarga de activar el movimiento y sincronización de los motores de la boca dependiendo de las letras que contiene cada palabra.

Las herramientas, algoritmos, paquetes y configuraciones necesarias para el funcionamiento de la aplicación se describen en el Anexo X.

4.5.1 Recursos de Hardware utilizados

Para este trabajo se utilizó un máquina virtual proporcionada por el Laboratorio Nacional de Supercómputo del Sureste de México (LNS). Las características de esta máquina de describen en la Tabla 4.10. En una segunda etapa se solicitó que duplicarán los recursos de procesadores y memoria RAM.

Tabla 4.10 Descripción de características de la Máquina Virtual proporcionada por el LNS.

No.	**Característica**	**Descripción**
1	Arquitectura	x86_64
2	Núcleos por socket	4
3	Socket	1 inicial / 2 final
4	CPU(s)	4 inicial / 8 final
5	Nombre Procesador	Intel(R) Xeon(R) CPU E5-2650 v3 2.30GHz
6	Fabricante	Genuine Intel
7	Memoria RAM	32 GB inicial / 64 GB final
8	Capacidad HDD	2 TB
9	Sistema Operativo	Ubuntu 16.04 Xenial Xerus

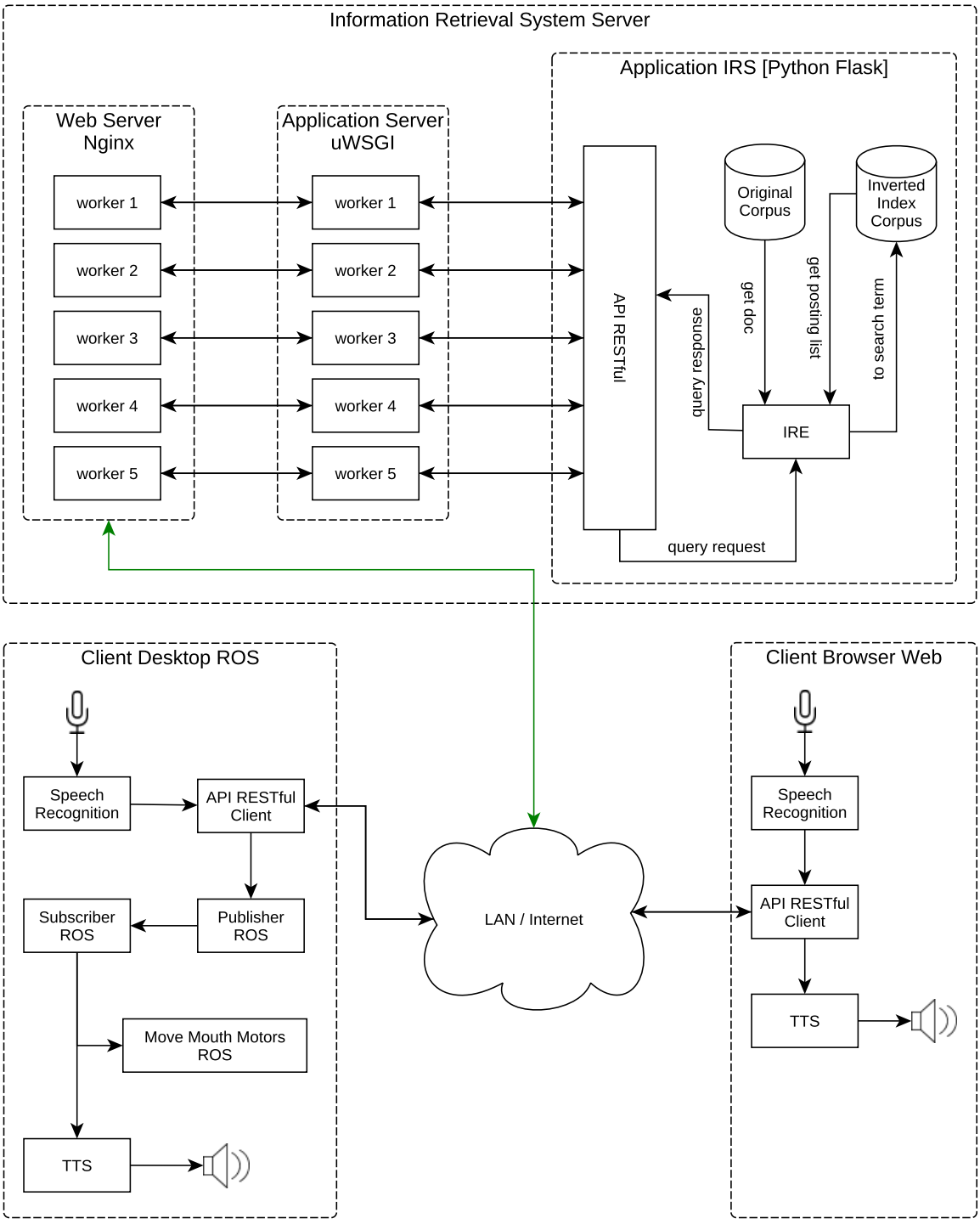

Fig. 4.6 Modelo de la Aplicación del IRS.

Inicialmente la máquina contaba con 4 CPUs y 32 GB en RAM, al estar aumentando de forma gradual los documentos descargados por el Crawler, se solicito al LNS que duplicarán los CPU y RAM, ya que son requeridos para el preprocesamiento, la creación del corpus de índices invertidos y para la carga en memoria RAM de este corpus.

Capítulo 5

Resultados

En esta sección se describen los resultados obtenidos, del crawler realizado se muestran las estadísticas de los documentos recuperados de la web. Del Sistema de Recuperación de Información (IRS) se muestran las consultas hechas a través del robot humanoide Arthur, utilizando una Reconocedor de Voz Automático (ASR) para la entrada por medio del micrófono, cuando se ha obtenido el texto del ASR se envía a través de una API RESTful al servidor, al llegar al servidor, el motor de recuperación de información (IRE) se encarga de obtener los documentos relevantes utilizando ranking de documentos, el resultado es enviado a través de una API RESTful (vía http) desde el servidor al cliente (en este caso a Arthur), después con un Sintetizador de Voz (TTS) se convierte una porción del documento en audio, para que Arthur mueva los motores de su boca al mismo tiempo que se escucha el audio. Además se presenta el IRS vía web de forma similar, utilizando un ASR para la entrada de las consultas (el teclado se puede utilizar como entrada), y el TTS para mostrar los resultados (también se muestran los resultados en forma escrita).

5.1 Crawler

Después de diseñar el Crawler y realizar las pruebas necesarias para detectar los errores y depurarlos, se inició su funcionamiento a finales del mes de Noviembre de 2016. En esta primera etapa la recuperación de documentos de la web se realizó con sólo un proceso dedicado a esta tarea, en una computadora de escritorio. A partir del mes de Enero de 2017 se migró el Crawler a un servidor, funcionando con diez procesos (hilos) de manera concurrente. Finalmente a mediados del mes de Marzo de 2017 el Laboratorio Nacional de Supercómputo del Sureste de

México (LNS) nos proporciono una máquina virtual (sus características se muestran en la Tabla 4.10) bajo el proyecto *"Procesamiento masivo de páginas web y redes sociales de México"*. En está máquina virtual el Crawler esta en funcionamiento con cien hilos de manera concurrente.

Tabla 5.1 URLs registradas por el Crawler durante el período del 24-Nov-16 al 31-May-2017

No.	Cantidad	Descripción
1	62,732,076	URLs pendientes por visitar
2	4,525,485	URLs visitadas
3	3,662,009	URLs de documentos descargados
4	863,476	URLs no descargadas
	67,257,561	URLs totales

En la Tabla 5.1 se listan las URLs que se han registrado en la DB y en la Figura 5.1 se muestra las estadísticas de URLs visitadas para descargar su contenido, se han accedido a 4,525,485 URLs, de las cuales 3,662,009 documentos se han descargado y 863,476 URLs no se ha podido acceder por diferentes códigos de error devueltos, como los que se muestran en la Tabla 5.6.

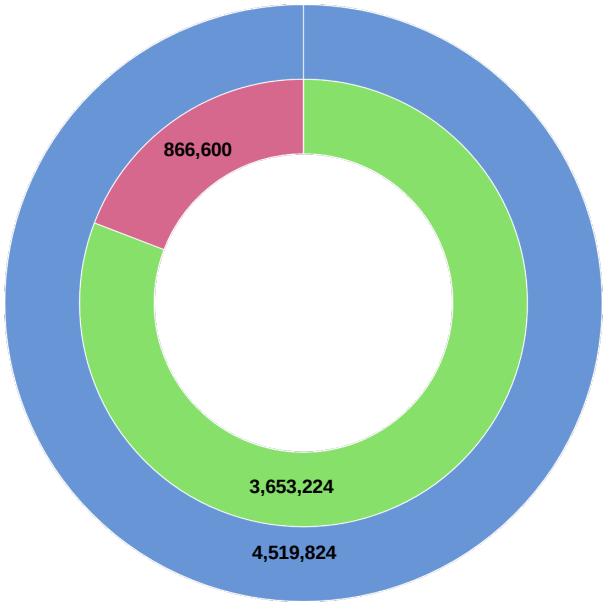

Fig. 5.1 Gráfica de URLs accedidas por el Crawler para la descarga de documentos.

El comportamiento del acceso a las URL y descarga de documentos por el Crawler se muestra en las Figuras 5.2, 5.3 y 5.4. En la Figura 5.2 se observa una gráfica de barras, las barras de color azul representan el total de URLs accedidas por día, las barras de color verde indican las URL accedidas de manera exitosa, lo que conlleva a la descarga del documento. Las barras de color rojo indican URLs que no se tiene permitido visitar o han devuelto un error como respuesta. Al migrar el Crawler de la PC a un servidor en Enero de 2017, muestra un alza en las visitas de URL que no se ha repetido hasta la fecha.

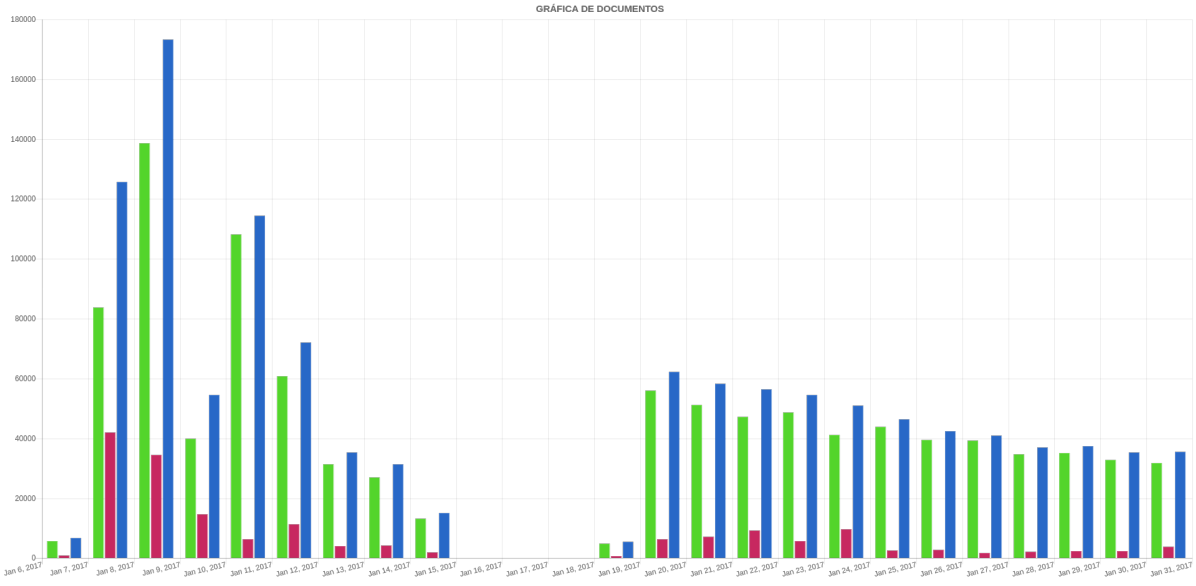

Fig. 5.2 Gráfica de URLs accedidas por el Crawler en el mes de Enero de 2017.

Para el mes de Marzo de 2017 se migró nuevamente, esta vez en la máquina virtual del LNS, se puede observar en la Figura 5.3 que a partir de la segunda mitad del mes los accesos diarios fueron de alrededor de 40 mil URLs visitadas por día, pero en los días siguientes el número de URLs visitadas por día ronda las 20 mil. Para el mes de Mayo continua decrementando el número de URLs visitadas diariamente situándose entre 15 a 20 mil URLs descargadas. Un análisis sobre los dominios de las URL visitadas en el mes de Mayo arroja a los sitios de *play.google.com*, *wikipedia.org* en sus múltiples lenguajes pero principalmente en idioma español, *www.stitcher.com*, *accounts.google.com* y en menor medidad *itunes.apple.com* y *twitter.com*. Este puede ser un mótivo por el bajo rendimiento en la visita de URLs, ya que estos sitios están preparados para detectar robots como es el caso del Crawler, por ello los principales dominios que rechazaron la

visita son *es.wikipedia.org*, *www.google.com.mx*, *news.google.com.mx*, *books.google.com.mx*, *play.google.com*, *www.facebook.com* y *www.google.com*.

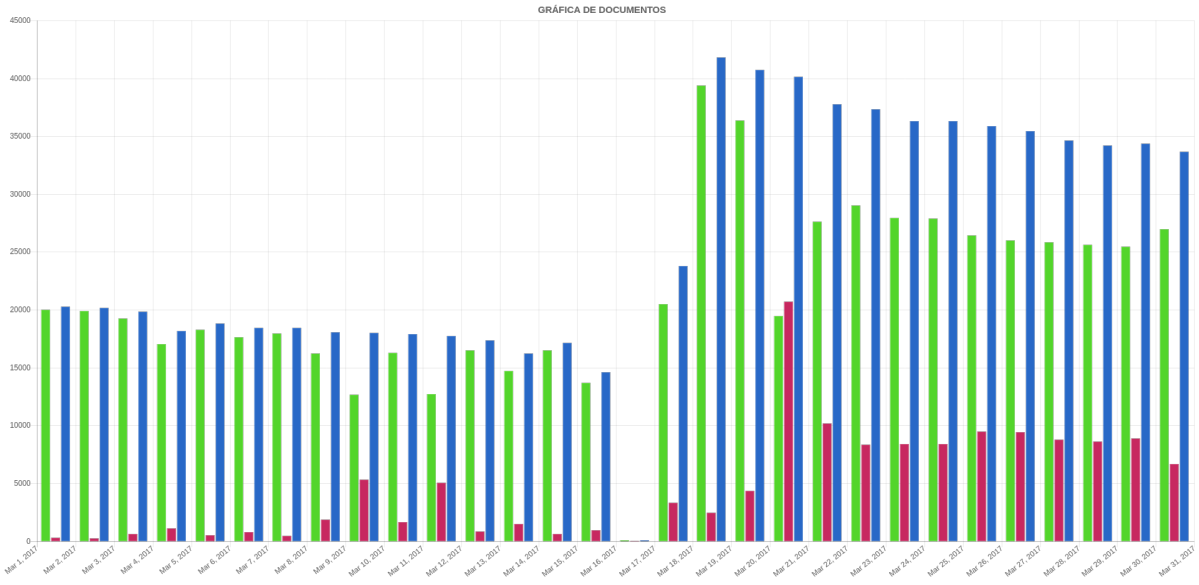

Fig. 5.3 Gráfica de URLs accedidas por el Crawler en el mes de Marzo de 2017.

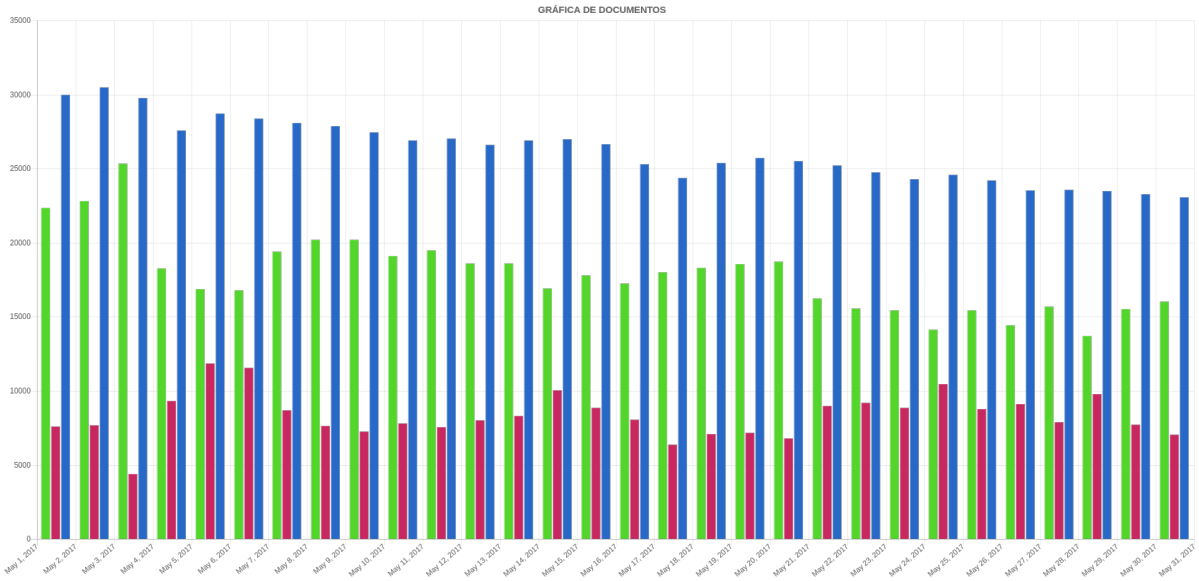

Fig. 5.4 Gráfica de URLs accedidas por el Crawler en el mes de Mayo de 2017.

Una de las tareas que contempla la creación del Crawler, es recopilar e identificar todos los sitios web que contengan en su dominio el sufijo **.mx**, la organización NIC (Network Information Center) México [23] se encarga de la administración del nombre de dominio territorial **.mx**, el código de dos letras asignado a cada país según el ISO 3166. Entre sus funciones están el proveer los servicios de información y registro para **.mx**, así como la asignación de direcciones de IP, y el mantenimiento de las bases de datos respectivas a cada recurso. La Tabla 5.2 indica las estadísticas actuales de estos dominios.

Tabla 5.2 Cantidad de nombres de dominio registrados bajo .mx por NIC México

No.	Dominio	Descripción	No. dominios
1	.com.mx	Usado para entidades comerciales	512,541
2	.net.mx	Proveedores de redes	329
3	.org.mx	Organizaciones no lucrativas	17,796
4	.edu.mx	Instituciones educativas	11,317
5	.gob.mx	Entidades gubernamentales	8,760
6	.mx	Indica un dominio territorial	297,213
Total			847,956

La recuperación de documentos por medio del Crawler bajo el sufijo .mx se observa en la Figura 5.5, el dominio con el mayor número es *.gob.mx* con 2,715 sitios web, 1,859 sitios con el dominio *.mx*, 984 del dominio *.com.mx*, del dominio *.edu.mx* y con 6 sitios web del dominio *.net.mx*. En la Tabla 5.3 se muestran el número de dominios de sitios web, así como el total de páginas web por dominio. El total de dominios recopilados hasta el momento es de 6,509 y el total de páginas web es de 199,885.

Tabla 5.3 Dominios .mx y páginas web de documentos recuperados por el Crawler.

No.	Dominio	No. Dominios	No. Páginas Web
1	.com.mx	984	41,992
2	.net.mx	6	94
3	.org.mx	561	8,973
4	.edu.mx	381	5,666
5	.gob.mx	2,715	112,356
6	.mx	1,859	30,804
Total		6,506	199,885

La Figura 5.6 muestra el comportamiento de los dominios .mx recopilados por el Crawler por mes. Los colores mostrados en está Figura corresponden al los de la Figura 5.5. En la

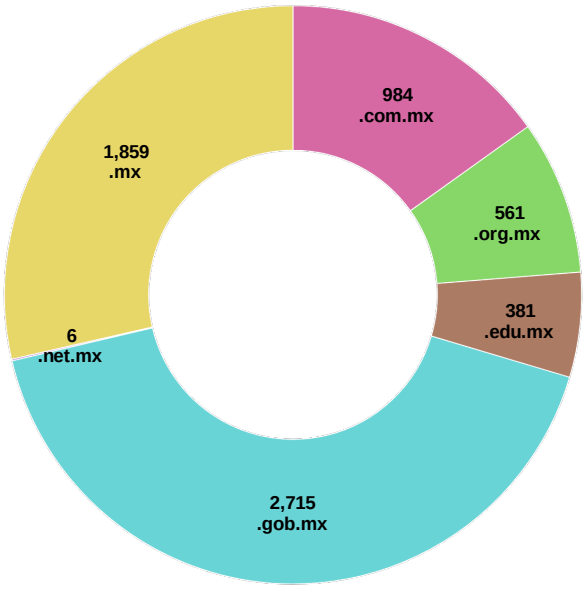

Fig. 5.5 Gráfica de dominios .mx recuperados por el Crawler.

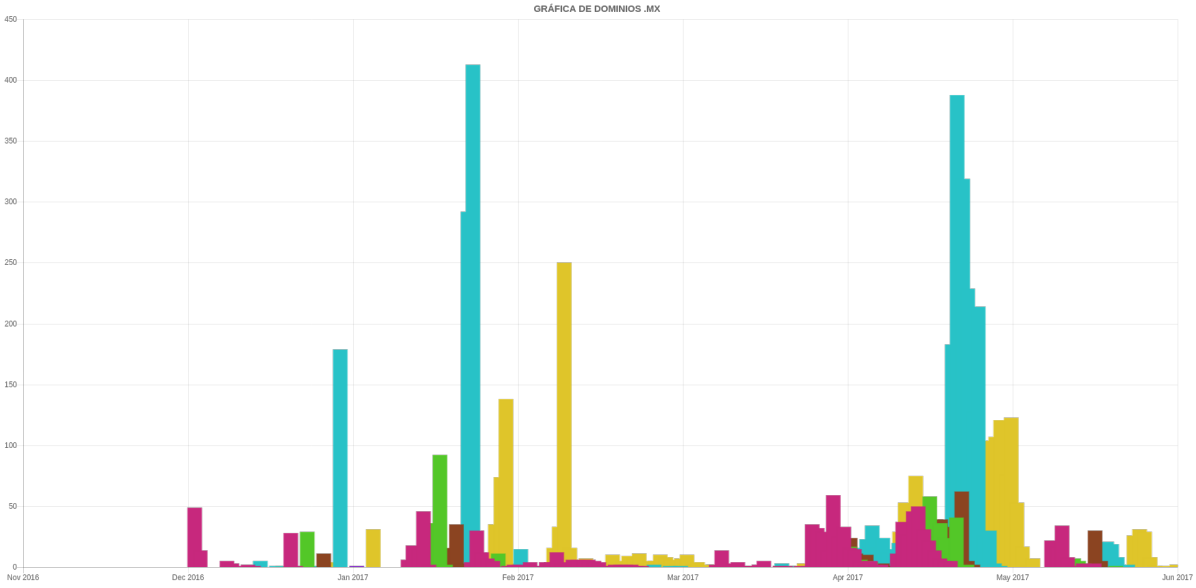

Fig. 5.6 Gráfica del comportamiento de dominios .mx recopilados por el Crawler.

Figura 5.6 se muestra un mayor número de dominios ocurrido en los meses de Febrero, Abril y Mayo de 2017. Existen varios períodos de tiempo en los que no se consigue descargar este tipo de dominios, esto se debe a las URL que se han visitado y no incluyen páginas web con estos dominios.

Los tipo de documentos recuperados se determinan a través de su Mime-Type, este se obtiene una vez que se ha descargado el documento usando el comando *file*. En la Tabla 5.4 se presentan los tipos de documentos recuperados, como se puede observar el mayor número corresponde a documentos *HTML*, seguido de documentos *XML*, *PDF* y *JEPG* y demás de la Tabla.

Cuando se visita una URL devuelve un código de estado, el número devuelto indica principalmente los siguientes tipos:

- 2xx para peticiones correctas, la petición fue recibida correctamente, entendida y aceptada.
- 3xx redireccion de URL, el cliente deberá tomar una acción adicional para completar la petición.
- 4xx errores por parte del cliente, la solicitud contiene sintaxis incorrecta o no puede procesarse.
- 5xx errores en del servidor, falló al completar una solicitud aparentemente válida.

Dentro de la Tabla 5.6 se muestran todos los códigos devueltos por peticiones HTTP, los códigos propios que se definieron para el Crawler son el 601, asignado cuando se verifica dentro del archivo robots.txt que no se tiene permitido visitar la URL, y el 666: que se utiliza para describir un error desconocido a los códigos antes mencionados por lo que no se visita la URL. Además existen algunos códigos que especifican un error, definidos por los sitios web a los que se intenta visitar como el código 999 y 810.

Tabla 5.4 Principales Mime-Type de los documentos recuperados.

No.	Mime-Type	Documentos
1	text/html	3,419,758
2	application/xml	158,228
3	application/pdf	41,739
4	image/jpeg	15,626
5	application/msword	3,965
6	inode/x-empty	2,151
7	text/plain	2,046
8	application/octet-stream	1,786
9	application/zip	1,737
10	image/png	1,450
11	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	745
12	application/vnd.ms-excel	502
13	application/x-shockwave-flash	464
14	application/vnd.openxmlformats-officedocument.wordprocessingml.document	456
15	video/mp4	384
16	audio/mpeg	339
17	application/x-dosexec	308
18	application/x-bzip2	280
19	application/gzip	243
20	application/vnd.ms-office	184
21	image/gif	165
22	video/quicktime	150

Tabla 5.6 Lista de Códigos devueltos en las peticiones de URL hechas por el Crawler.

No.	Código	Descripción	No. URLs
1	200	OK	3,655,131
1	601	Acceso no permitido por robots.txt	578,221
2	404	Not Found	138,939
3	666	Unknown Error	79,684
4	400	Bad Request	29,117
5	429	Too Many Requests	17,605
6	403	Forbidden	9,153
7	500	Internal Server Error	3,571
8	503	Service Unavailable	1,124
9	401	Unauthorized	1,022
10	302	Found	604
11	999	Solicitud Denegada LinkedIn	338
12	410	Gone	298
13	505	HTTP Version Not Supported	284
14	301	Moved Permanently	135
15	415	Unsupported Media Type	111
16	303	See Other (desde HTTP/1.1)	97
17	502	Bad Gateway	83
18	406	Not Acceptable	39
19	456	Autodiscover MSDN - Microsoft	17
20	308	Permanent Redirect	16
21	300	Multiple Choices	15
22	530	Not logged - remote FTP	14
23	504	Gateway Timeout	12
24	409	Conflict	9
25	521	Refused connection - Cloudflare	5
26	402	Payment Required	3
27	810	Incorrect Certificate for VPN authentication	3
28	405	Method Not Allowed	2
29	512	Not updated	2
30	523	Origin is unreachable - Cloudflare	1
31	408	Request Timeout	1
32	461	Don’t retry	1
33	307	Temporary Redirect (desde HTTP/1.1)	1
34	501	Not Implemented	1

5.2 Sistema de Recuperación de Información

El Sistema de Recuperación de Información (IRS) construido recibe como entrada el texto obtenido del ASR (ya sea a través de el Robot Humanoide Arthur o de la página web), después se envía la consulta en formato texto a través de la API RESTful hacia el servidor, que contiene el Motor de Recuperación de Información (IRE) para recuperar los documentos relevantes de la consulta del usuario, la respuesta es devuelta a través de la API RESTful, después en el caso de Arthur, se configuró el TTS en idioma Español así la respuesta puede escucharse a través del robot. En el caso del TTS de la página web se utiliza las herramientas de la Web Speech API.

5.2.1 Experimento 1

A continuación se listan algunos experimentos realizados con el IRS. El primero contempla la consulta: *"Enfermedad de la Esquizofrenia"* hecha por el usuario *orlando*, al enviar la consulta el IRE descarta las stopwords, la consulta es procesada y se regresa en formato JSON como se observa en el Código 5.1.

Código 5.1 Respuesta de la API RESTful en formato JSON al cliente

```
task: {
  id: 1
  documents: [{
    text: "NIMH Trastorno Bipolar (facil de leer)",
    astts: "department of health and human services",
    score: 1935973.9404076675,
    astext: "Department of Health and Human Services",
    docID: 12319
  }],
  length: 1021,
  query: "enfermedad esquizofrenia",
  retrieved: True,
  time: 0.02,
  user: "orlando",
}
```

La respuesta recibida a través del Código 5.1 muestra un arreglo *documents* (se pueden devolver más de un documento), la etiqueta *astex* contiene el resumen automático calculado por el algoritmo *TextRank* implementado en el IRE, *astts* es el mismo resumen automático con la diferencia que se ha preprocesado y configurado para ser interpretado por el TTS, el *docID*

es el Id del documento, *text* que contiene el texto tal y como se encuentra en el documento original, y *score* el puntaje alcanzado por el Modelo de Ranking de documentos. Además la etiqueta *retrieved* indica *True/False* para indicar si se recuperaron documentos, *query* define la consulta preprocesada regularmente diferente de la original, *user* es el nombre del usuario que solicitó la consulta y *length* contiene el total de documentos relacionados a la consulta. El siguiente párrafo muestra el texto original del documento, las palabras de la consulta se resaltan en negrita, para ilustrar este documento únicamente se muestran las oraciones que contienen las palabras de la consulta.

NIMH Trastorno Bipolar (fácil de leer) ... Algunas personas que experimentan estos síntomas sufren del trastorno bipolar, una grave ***enfermedad*** *mental. Lea este folleto para obtener más información. ¿Qué es el trastorno bipolar? El trastorno bipolar es una grave* ***enfermedad*** *del cerebro. También se llama* ***enfermedad*** *maníaco-depresiva. Los que sufren del trastorno bipolar experimentan cambios de ánimo inusuales ... Generalmente la* ***enfermedad*** *dura toda la vida. cuáles son los síntomas del trastorno bipolar? Los cambios de estado de ánimo bipolares se llaman "episodios anímicos" ... Los genes, porque la* ***enfermedad*** *es hereditaria La anormalidad en la estructura y función del cerebro ... Un médico puede creer que la persona tiene una* ***enfermedad*** *distinta como, por ejemplo,* ***esquizofrenia*** *o depresión. Además, los que sufren del trastorno bipolar a menudo tienen otros problemas de salud. Esto puede hacer que a los médicos les sea difícil diagnosticar el trastorno bipolar. Ejemplos de estos otros problemas incluyen el abuso de sustancias, los trastornos de ansiedad, la* ***enfermedad*** *de la tiroides, las enfermedades cardíacas, y la obesidad. ¿Cómo se trata el trastorno bipolar? Por ahora, el trastorno bipolar no tiene cura ... Sabía que es una* ***enfermedad*** *grave, pero se sintió aliviado al descubrirlo, pues tuvo síntomas durante años pero nadie sabía porque. Ahora está recibiendo tratamiento y se siente mejor ...*

En algunos casos el algorimto TextRank no genera oraciones que incluyan las palabras de la consulta, esto se debe a la evaluación del algoritmo Page Rank que determina que ciertas palabras en una oración son más relevantes que otras, como en este caso. Al analizar el resumen generado se observa una palabra que se repite prácticamente en todas las oraciones: *"mejor"*, es por tal motivo que estas oraciones tienen mayor peso que las palabras de la consulta y por el tipo de algoritmo utilizado. El siguiente párrafo muestra el resumen generado de forma automática.

Department of Health and Human Services. Ahora está recibiendo tratamiento y se siente mejor. Pero el tratamiento es la mejor manera de comenzar a sentirse mejor. También puede ayudar a los pacientes a llevarse mejor con familiares y amigos. Un tratamiento funciona mejor cuando es continuo y no es interrumpido de vez en cuando. A veces una persona debe probar distintos medicamentos para descubrir cuáles dan mejor resultado. Si mantiene su nivel de estrés bajo podrá desempeñarse mejor y podrá ayudar a su ser querido a seguir el tratamiento. STR 09-3679Modificado en julio del 2009 The National Institute of Mental Health (NIMH) is part of the National Institutes of Health (NIH), a component of the U. NIMH Trastorno Bipolar (fácil de leer) Health & Education Outreach Research Priorities Funding Labs at NIMH News About Us Transforming the understanding and treatment of mental illnesses

En cuanto al resumen automático generado para el TTS, consiste de limpiar el texto de caracteres especiales y convertir a minúsculas el texto, además de cambiar el formato de *acentos* y la *ñ*, las comas (,) y puntos (.) permanecen. La siguiente lista muestra los cambios de formato:

- reemplazar **á** por **'a**
- reemplazar **é** por **'e**
- reemplazar **í** por **'i**
- reemplazar **ó** por **'o**
- reemplazar **ú** por **'u**
- reemplazar **ñ** por **~n**

De esta forma el resumen automático en formato TTS esta listo para ser interpretado y convertido en audio, para transmitir la respuesta a la búsqueda solicitada. El párrafo es el siguiente:

department of health and human services. ahora est'a recibiendo tratamiento y se siente mejor. pero el tratamiento es la mejor manera de comenzar a sentirse mejor. tambi'en puede ayudar a los pacientes a llevarse mejor con familiares y amigos. un tratamiento funciona mejor cuando es continuo y no es interrumpido de vez en cuando. a veces una persona debe probar distintos medicamentos para descubrir cu'ales dan mejor resultado. si mantiene su nivel de estr'es bajo podr'a desempe narse mejor y podr'a ayudar a su ser querido a seguir el

tratamiento. str 09 3679modificado en julio del 2009 the national institute of mental health nimh is part of the national institutes of health nih , a component of the u. nimh trastorno bipolar f'acil de leer health education outreach research priorities funding labs at nimh news about us transforming the understanding and treatment of mental illnesses

5.2.2 Experimento 2

Aquí se muestra el IRS Web que se realizó, la interfaz muestra un menu *Inicio* y *Estadísticas*. En *Estadísticas* se muestran las gráficas del Crawler, y en *Inicio* se encuentra el formulario para realizar las búsquedas en el IRS, se cuenta con una caja de texto para escribir la consulta y para realizar la búsqueda se presiona el botón con el ícono de lupa o presionando la tecla *enter*, también se puede utilizar el Reconocedor de Voz presionando el botón con el ícono de micrófono como se observa en la Figura 5.7. En este experimento se realiza la consulta *"rutina de ejercicio para quemar la grasa abdominal"*, únicamentte se muestran los primeros 10 documentos (los mejores rankeados según el algoritmo).

Fig. 5.7 Interfaz Web del Sistema de Recuperación de Información.

En la interfaz se observa, la cantidad de documentos encontrados con la información solicitada, en este caso es de *1,876 documentos*, en una colección de *100,000 documentos* con

un tiempo de respuesta de *4.06 segundos*. Además se resaltan en negrita las palabras clave de la consulta, cada párrafo tiene una breve porción del texto, para mostrar el resumen obtenido por el algoritmo TextRank basta con dar clic en el segundo ícono que se encuentra sobre cada párrafo, como se observa en la Figura 5.8 para los dos primeros resultados.

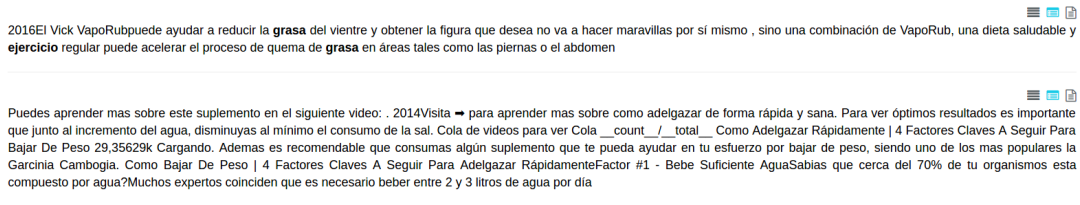

Fig. 5.8 Resumen de la consulta para el experimento 2.

Para visualizar el texto completo del documento recuperado se puede dar clic al último ícono de cada párrafo, como se observa en la Figura 5.9. Además se puede visualizar el resumen del texto y el texto en su versión breve presionando los íconos correspondientes.

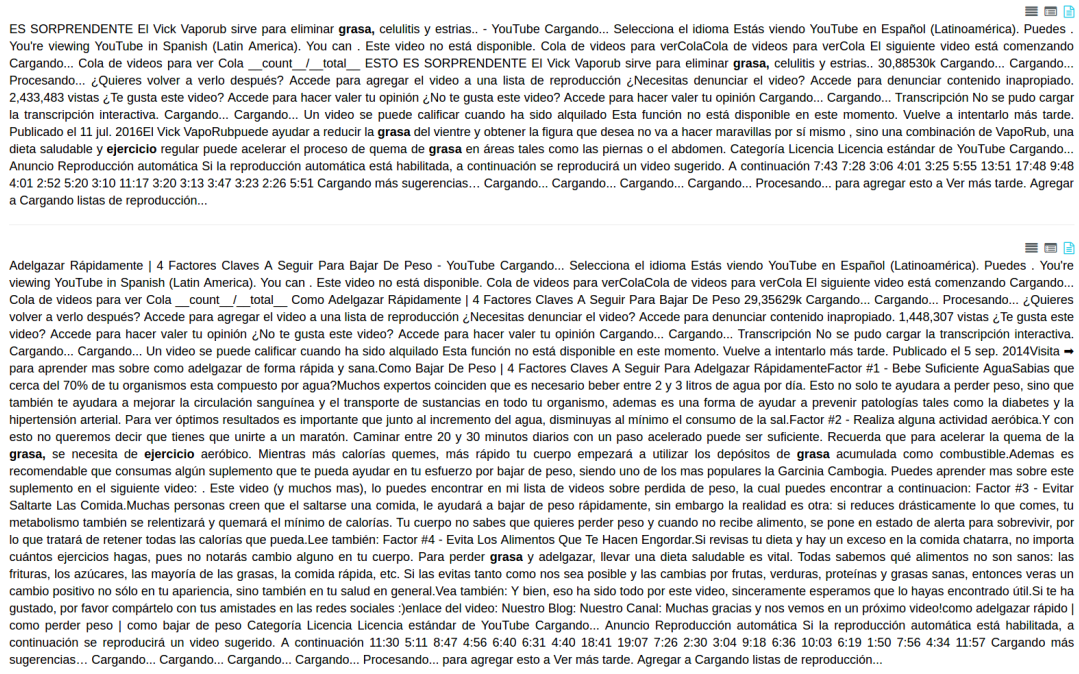

Fig. 5.9 Texto completo de la consulta para el experimento 2.

5.2.3 Experimento 3

Este experimento consiste en la Interacción Humano-Robot con Arthur. Se plantea una consulta por medio de la voz del usuario hacia el robot humanoide, quien la procesa utilizando Sphinx y la API de google speech para reconocer la consulta y convertirla en texto, posteriormente la consulta es enviada a través de la API RESTful al IRS, la respuesta se procesa con el TTS del LKE para convertir el texto en audio, de esta forma el robot por medio de expresiones faciales expresa la respuesta final al usuario.

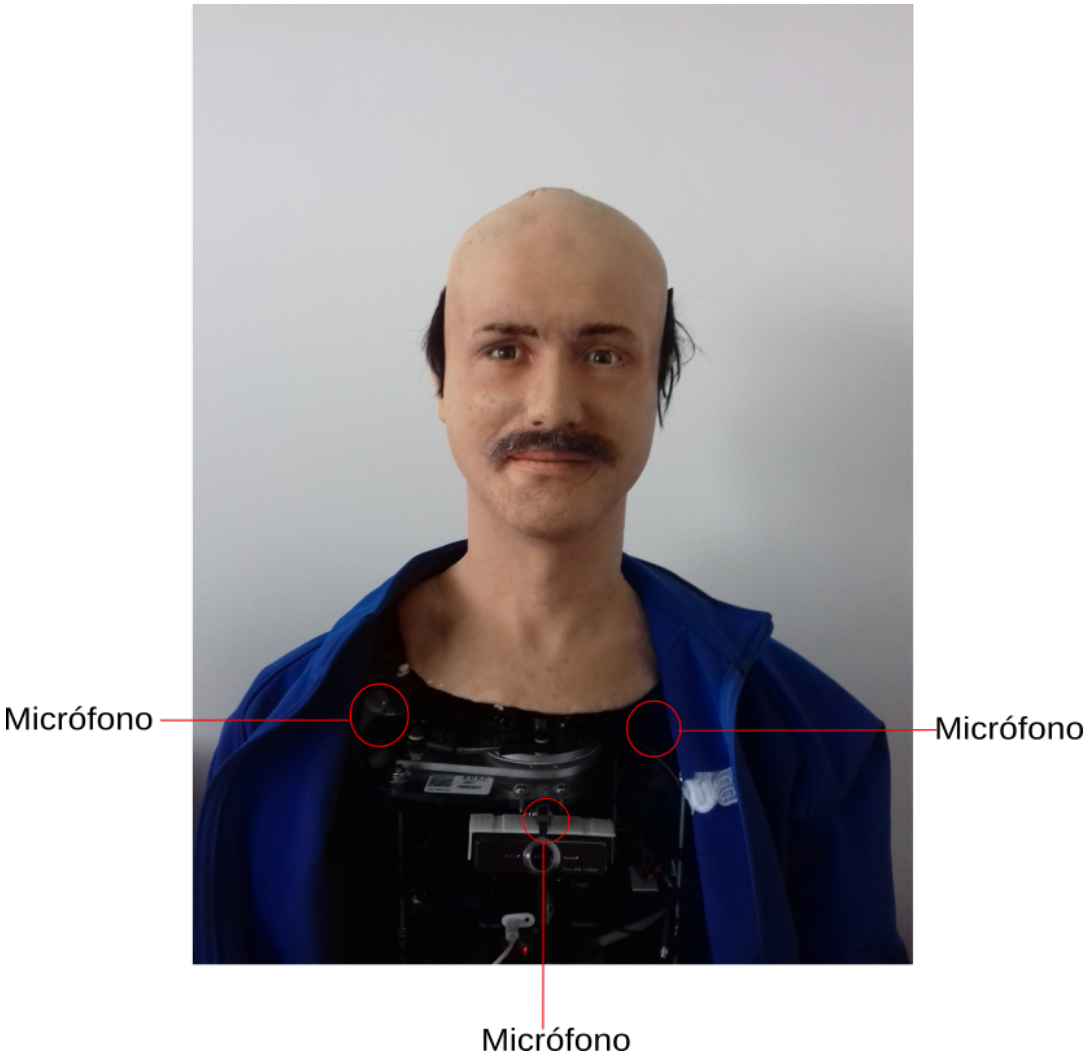

Fig. 5.10 Ubicación física de micrófonos del robot humanoide Arthur.

Fig. 5.11 Expresiones del rostro mientras habla el robot humanoide Arthur.

En la Figura 5.10 se observa la ubicación de los micrófonos del robot humanoide Arthur, para el Reconocimiento Automático de Voz se utilizó el micrófono central que captura la voz del usuario, se procesa utilizando Sphinx con la API de Google Speech, después de realizar el reconocimiento se envía el texto al servidor con el Motor de Recuperación de Información a través de la API RESTful, la consulta se procesa recuperando el documento con el mayor score, se obtiene el resumen automático del documento y se regresa el resultado por medio de la API RESTful.

Cuando se obtiene el resumen (en formato TTS) devuelto por la API RESTful se procesa el texto con el Sintetizador de Voz LKE para obtener un audio en formato wav, posteriormente el tópico de ROS chatbot, se encarga de reproducir el audio y mover los motores de la boca del robot, para expresar el resultado de la consulta al usuario de una forma natural. Algunas de las expresiones se capturaron a manera de ejemplo son mostradas en la Figura 5.11. Para cada consulta se repite el proceso.

Capítulo 6

Conclusiones

En este capítulo se describen las conclusiones sobre la creación en implementación, de un Sistema de Recuperación de Información (IRS) como Mecanismo de Interacción con un Robot Humaoide y cada uno de sus módulos (ASR, TTS, IRE y API RESTful).

6.1 Reconocimiento de Voz Automático

Los algoritmos desarrollados para el Reconocimiento de Voz Automático (ASR) utilizando la API de escritorio con los modelos de lenguaje y acústico de Sphinx, funcionan de forma correcta siempre y cuando las palabras a reconocer estén dentro de su modelo de lenguaje, además el audio que recibe debe ser lo más claro posible, ya que esto limita el reconocimiento de forma adecuada. Sin embargo cuando se utiliza el ASR con la API de Google Speech, el reconocimiento casi siempre es el esperado, y la desventaja de usar esta herramienta es el limite de peticiones (1 petición=15 seg, 60 minutos al mes), o de lo contrario se tiene que adquirir una licencia.

Para el ASR del Sistema Web, al utilizar el motor del navegador Chrome con la Web Speech API, se obtienen excelentes resultados, ya que se realiza el reconocimiento de voz con la ayuda de Google, aunque para el desarrollador la forma de llevar a cabo este procesamiento no es transparente, esto a que funciona dentro del motor del navegador. Una de las desventajas puede ser el hecho de que es una tecnología experimental, según la documentación en Mozilla Developer Network[1] (MDN) por lo que no se tiene una documentación detallada. Además solo funciona en los navegadores Chrome, Firefox y Edge.

[1] https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

6.2 Sintetizador de Voz

La utilización del Sintetizador de Voz (TTS) de la herramienta Festival[2] se implementó en ROS para el robot humanoide (Arthur), el nodo encargado del TTS recibe la ruta del archivo de audio en Español, lo reproduce al mismo tiempo que se activa el tópico de las expresiones faciales para mover los motores de la boca. Un punto a considera al usar el TTS de festival es precisamente la voz, ya que es usada en muchos proyectos (robots). Así que para que Arthur tenga una voz propia, se utiliza el TTS del LKE. Actualmente se trabaja en refinar este último TTS con Arthur.

En el IRS Web se utiliza la interfaz SpeechSynthesis de la Web Speech API, todo el procedimiento se realiza dentro del motor del navegador al igual que el Speech Recognition, los resultados son los esperados y pueden elegirse diferentes lenguajes de voz, para el Español la voz es de tipo femenino, aunque hay algunas que contemplan los dos tipos de voz, femenino y masculino (Inglés USA). Los inconvenientes son similares al Speech Recognition, ya que el procedimiento de TTS se oculta al desarrollador.

Una opción más de TTS para el Sistema Web consiste en utilizar el TTS LKE y obtener el archivo wav, enviarlo desde el servidor hacia el cliente en base64 convirtiendo el archivo de audio wav en una cadena de texto. El cliente recibe esta cadena, se convierte nuevamente de base64 a wav y se reproduce en la página web. Un posible inconveniente de usar este método, es para archivos grandes de audio, ya que la cadena aumenta en tamaño considerablemente, y el tiempo de respuesta puede tardar un poco más, ya que se tiene que procesar el archivo de audio.

6.3 Motor de Recuperación de Información

El Motor de Recuperación de Información (IRE) desarrollado permite realizar consultas para la extracción de documentos. La implementación de los algoritmos del Modelo de Espacio Vectorial utiliza un Ranking de Documentos, de esta forma el usuario obtiene un resultado relevante a la consulta, en cambio si utiliza un Modelo Booleano, recibiría documentos en el orden en que se indexaron. El IRE contempla los algoritmos para la extracción de texto de los documentos HTML, el preprocesamiento de documentos para la creación del corpus de índices invertidos, y la implementación del algoritmo TextRank.

Todos los algoritmos están escritos en lenguaje Pyhton 3.5, y para soportar el IRE se utilizó un Microframework llamado Flask en un servidor (máquina virtual) proporcionado por el

[2]http://www.cstr.ed.ac.uk/projects/festival/

LNS, con esta misma herramienta se crea la API RESTful para proporcionar el servicio a los clientes (usuarios), además se realizan las configuraciones necesarias para utilizar un servidor de aplicaciones entre el IRE y el servidor HTTP.

En los experimentos que se realizaron se utilizó un Corpus de índices invertidos de diferentes capacidades, de 10 mil documentos, 50 mil documentos, 100 mil documentos y 1 millón de documentos. Un inconveniente para el Corpus de 1 millón de documentos, es el espacio que ocupa en la memoria RAM del servidor, alrededor de 25.9 GB de los 64 GB disponibles. Por ello se trabajó con los Corpus de menor tamaño (de 50 mil y 100 mil). Al incrementarse el número de documentos, el tiempo de respuesta también incrementa.

En este trabajo de tesis la evaluación del IRS no se realizó, ya que no se cuenta con un conjunto de documentos definidos con un juicio (experto) de relevancia, es decir una evaluación binaria para documentos relevantes y otro par para documentos no relevantes. Además se necesitaría una colección de documentos específicos así como de consultas.

6.4 API RESTful

El motivo principal de crear una API para acceder al IRS, se debe a la posibilidad de que este Sistema pueda ser utilizado por múltiples usuarios, en diferentes lugares y al mismo tiempo. Por ello se implementó un servicio web a través de la API RESTful, además beneficia la búsqueda de documentos desde la interfaz (PC) que controla al robot (Arthur), sin tener la necesidad de realizar algún tipo de instalación extra para el funcionamiento del IRS.

Para el funcionamiento de la API RESTful del lado del cliente solo se necesita la URL, un usuario y password y una cadena de texto con la consulta, con estos elementos se puede realizar la conexión al servidor. La conexión y consulta así como la respuesta se presentan en formato JSON. En este trabajo se utilizó la API desde el cliente con lenguaje Python y para el IRS Web en lenguaje Javascript. La API solo contempla la operación básica de autenticación y no se cuenta con una interfaz para la administración de usuarios.

6.5 Trabajo a Futuro

Como trabajo a futuro se pretende recopilar un mayor volumen de páginas web, de sitios con dominios *.mx*, para proyectos académicos y de investigación. Además de incrementar el corpus existente de documentos recuperados. De manera paralela se plantea realizar un Crawler especializado en ciertos sitios y anexar los algoritmos al IRS para la búsqueda de definiciones analíticas y campos semánticos en la red, entre otras tareas.

En cuanto al ASR utilizando Sphinx se planea la mejora del reconocimiento, entrenando el modelo acústico y de lenguaje con más palabras del idioma Español, o inclusive con el vocabulario del Corpus de índices invertidos. En lo que se refiere al IRE, se planea reducir el espacio en memoria RAM, utilizando una técnica de compresión de datos en el corpus de índices invertidos.

Referencias

[1] Antal Bosch, Toine Bogers, and Maurice Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

[2] Vladislav Shkapenyuk and Torsten Suel. Design and implementation of a high-performance distributed web crawler. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 357–368. IEEE, 2002.

[3] Cristian Merlino-Santesteban. Análisis de conectividad en la recuperación de información web. *Ciência dóna Informação*, 32(3):113–119, 2003.

[4] Antti Puurula. Cumulative progress in language models for information retrieval. In *Proceedings of Australasian Language Technology Association Workshop*, pages 96–100. Citeseer, 2013.

[5] ChengXiang Zhai. Notes on the kl-divergence retrieval formula and dirichlet prior smoothing. *Class notes for introduction to text information systems at the Dept. of Computer Science*, 2007.

[6] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

[7] Hoojung Chung, Young-In Song, Kyoung-Soo Han, Do-Sang Yoon, Joo-Young Lee, Hae-Chang Rim, and Soo-Hong Kim. A practical qa system in restricted domains. In *Proceedings of the Workshop Question Answering in Restricted Domains, within ACL*, 2004.

[8] Donghui Feng, Erin Shaw, Jihie Kim, and Eduard Hovy. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 171–177. ACM, 2006.

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[10] Kyung-Min Kim, Chang-Jun Nan, Jung-Woo Ha, Yu-Jung Heo, and Byoung-Tak Zhang. Pororobot: A deep learning robot that plays video q&a games. In *AAAI 2015 Fall Symposium on AI for Human-Robot Interaction (AI-HRI 2015)*, 2015.

[11] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304, 2015.

[12] Andrea Bauer, Dirk Wollherr, and Martin Buss. Information retrieval system for human-robot communication-asking for directions. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4150–4155. IEEE, 2009.

[13] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[14] David A Ferrucci. Ibm's watson/deepqa. In *ACM SIGARCH Computer Architecture News*, volume 39. ACM, 2011.

[15] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690, 2014.

[16] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[17] Akinobu Lee and Tatsuya Kawahara. Recent development of open-source speech recognition engine julius. In *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pages 131–137. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, International Organizing Committee, 2009.

[18] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[19] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[20] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[22] Héctor Jiménez-Salazar, David Pinto, and Paolo Rosso. Uso del punto de transición en la selección de términos ındice para agrupamiento de textos cortos. *Procesamiento del Lenguaje Natural*, 35:383–390, 2005.

[23] Network Information Center México S.C. Indicadores de dominios, 2017.

Anexo A

Crawler

En esta sección se mencionan las herramientas, algoritmos y paquetes utilizadas para esta tarea. La sección contiene las subsecciones Base de Datos, Algoritmos, Paquetes e instalación, donde se describe el funcionamiento de cada una de ellas.

A.1 Base de Datos

En esta sección se describe el manejador de Base de Datos PostgreSQL, así como las tablas que contemplan las Base de Datos. PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto. Por tal motivo se eligió este administrador de bases de datos. La versión con la que se esta trabajando hasta el momento es PostgreSQL 9.5.

A.1.1 Instalación

Para instalar PostgreSQL, escribir los siguientes comandos en la terminal, se necesitan privilegios de administrador:

- **sudo apt-get update**
- **sudo apt-get install postgresql postgresql-client postgresql-contrib libpq-dev**

Para verificar que se ha instalado correctamente, utilizar el comando **psql** que da acceso a la consola de PostgreSQL, pero únicamente se usará para mostrar la versión de PostgreSQL con el siguiente comando:

psql -V

A.1.2 Accesando a la consola

Para accesar a la consola con el comando **psql** y el usuario **postgres** con **sudo** (privilegios de administrador) sin crear ni cambiar usuario, escribir en la terminal:

sudo -u postgres psql

A continuación se listan algunos comandos útiles para desplegar información dentro de la consola de PostgreSQL:

- **\q** Para salir de la consola.
- **\l** Mostrar las bases de datos.
- **\c nombreBD** Para conectarse (seleccionar) a una base de datos específica.
- **\conninfo** Muestra información de la conexión: base de datos, usuario, socket y puerto.
- **\d** Lista las relaciones que existen en la base de datos.
- **\dt** Lista las tablas contenidas de la base de datos.
- **\d nombreTabla** Muestra la estructura de la tabla.
- **\password** Cambiar el password de acceso a la base de datos del usuario.
- **\?** Para desplegar en pantalla más comandos.

A.1.3 Tablas de la Base de Datos

Las Tablas A.1 y A.2 describen la información de las únicas dos Tabla que contiene la Base de Datos, la descripción de cada uno de los campos de la Tabla A.1 se específica a continuación:

- **id** Es la llave primaría de la tabla, es de tipo numérico y autoincrementable.
- **rooturl** Campo asignado para almacenar las URLs. Además es un campo único, para evitar URLs repetidas.
- **mimetype** Almacena el mime-type del documento descargado.
- **status** Se específica el tipo de estatus de la URL, ejemplo: 200, 404, 303, 501, etc.
- **filename** Almacena la URL parcial dónde se guarda el documento, ejemplo: 2017-04-06/1000.html

Tabla A.1 Estructura para almacenar URLs en la Tabla del Crawler.

Column Name	Type	Modifiers
id	bigint	not null default autoincrement
rooturl	text	not null
mimetype	text	not null default ”::text
status	smallint	not null default ’0’::smallint
filename	text	not null default ”::text
language	text	not null default ”::text
visited	smallint	not null default ’0’::smallint
tries	smallint	not null default ’0’::smallint
parsed	smallint	not null default ’0’::smallint
indexed	smallint	not null default ’0’::smallint
startdate	timestamp with time zone	not null default ’1970-01-01 00:00:00’::timestamp
downloaddate	timestamp with time zone	not null default ’1970-01-01 00:00:00’::timestamp
Index:	PRIMARY KEY, btree (id)	
Index:	UNIQUE CONSTRAINT, btree (rooturl)	

- **language** Cuando se procesan los documentos, se específica su lenguaje, ejemplo: es, en, fr, etc.
- **visited** Bandera utilizada para indicar si se ha visitado una URL.
- **tries** Número de veces que se ha visitado esa URL.
- **parsed** Es un campo tipo bandera para indicar que el documento se ha procesado.
- **indexed** Bandera para indicar si el documento se encuentra indexado en el corpus.
- **startdate** Timestamp que indica la fecha y hora en que se inserto la URL a la base de datos.
- **downloaddate** Timestamp que indica la fecha y hora en que se descargo el documento.

La descripción de los campos de la Tabla A.2 almacena el total de las URL almacenadas en la Tabla del Crawler, se filtran los datos por fecha actual cada hora, para evitar congestionamiento en la DB.

- **fecha** Es la llave primaría de la tabla, es de tipo fecha.
- **descargado** Campo asignado para almacenar todas las URL descargadas.

Tabla A.2 Estructura de Tabla de Estadísticas.

Column Name	Type	Modifiers
fecha	date	not null
descargado	integer	not null default 0
nodescargado	integer	not null default 0
com_mx	integer	not null default 0
org_mx	integer	not null default 0
edu_mx	integer	not null default 0
gob_mx	integer	not null default 0
net_mx	integer	not null default 0
mx	integer	not null default 0
Index:	PRIMARY KEY, btree (fecha)	

- **nodescargado** Almacena las URL que no se descargaron.
- **com_mx** Almacena las URL con dominios .com.mx
- **org_mx** Almacena las URL con dominios org.mx
- **edu_mx** Almacena las URL con dominios edu.mx
- **gob_mx** Almacena las URL con dominios gob.mx
- **net_mx** Almacena las URL con dominios net.mx
- **mx** Almacena las URL con dominios con .mx

A.2 Algoritmos

En esta sección se presentan tres algoritmos esenciales para el funcionamiento del Crawler, consiste de un algoritmo de Web Scraping para la extracción de documentos en la web, un algoritmo para extraer los links de un documento HTML y un algoritmo general que hace uso de los dos primeros algoritmos.

A.2.1 Funcionamiento del Crawler

El algoritmo para descargar documentos, cuenta con hilos para accesar a la Base de Datos de forma síncrona, véase el Algoritmo 2. Lo primero que se hace es crear una conexión con la DB, después en un ciclo infinito se ejecuta el Crawler, se ejecutan varios hilos al mismo tiempo, se sincronizan al accesar al recurso de la DB. La idea principal de este Crawler es extraer todos los links que contenga un documento HTML y alimentar con ellos la DB, para así volver a utilizar dichos links en más búsquedas.

A.2.2 Web Scraping

Web Scraping es una técnica que se utiliza mediante programas de Software para extraer información de sitios web, simulando ser una persona que accede a través de un navegador web. Se utilizo un algoritmo de este tipo creado en lenguaje Python para extraer documentos de la web.

En primera instancia en el Algoritmo 3 se definen los encabezados, es decir la forma en la que el algoritmo se presenta cuando visita una página web, simulando ser un navegador (Chrome, Firefox, Safari, etc.), si la conexión con dicha URL es positiva devuelve un *status* (código numérico), el número 200 indica que la conexión es exitosa. En este punto se analiza el archivo *robots.txt* que debe encontrarse en la raíz del dominio, se verifica que se tenga permiso para visitar la url, de ser así se descarga el documento, una vez que se ha descargado se extrae su *Mime-Type* para saber el tipo de documento que se descargo. Como paso final se regresa un valor de verdadero, el status producido, el *Mime-Type* y la ruta de almacenamiento físico del archivo descargado.

Algoritmo 2 Funcionamiento del Crawler.

```
procedure CRAWLER
    connection ← to make connect to DB
    while True do
        thread.lock()
        connection.query('Select url not visited')
        thread.release()
        url ← connection.data.url
        documentName ← connection.data.id
        ws ← WEBSCRAPING(url, documentName)
        if ws.value = True then
            mimetype ← ws.mimetype
            filePath ← ws.filePath
            thread.lock()
            conection.update('mimetype, filePath, datetime, mark as visited ')
            thread.release()
            if mimetype = 'html' then
                linkList ← GETLINKS(filePath, url)
                if length(linkList) > 0 then
                    thread.lock()
                    for each link in linkList do
                        conection.insert('insert new link')
                    end for
                    thread.release()
                end if
            end if
        else
            thread.lock()
            conection.update('filePath, datetime, mark as visited')
            thread.release()
            ws.message                                  ▷ Print error message
        end if
    end while
end procedure

procedure RUNCRAWLER
    threads_number > 0
    for each i such that 0 ≤ i ≤ threads_number do
        Thread.run(CRAWLER, i)
    end for
end procedure
```

Algoritmo 3 Algoritmo de Web Scraping

```
procedure WEBSCRAPING(url, documentName)
    headers ← to define http headers                ▷ "User-Agent" : "Mozilla/5.0 ..."
    response ← request(headers, timeout=5)
    if response.status = 200 then
        if ALLOWVISTURL(url) = True then
            documentStored ← FILEWRITE(response.text, documentName)
            mimetype ← GETMIMETYPE(documentStored)
            filePath ← GETFILEPATH(documentStored)
            return (True, response.status, filePath, mimetype)
        else
            return (False, 601, "Not Allowed to visit URL")
        end if
    else
        return (False, response.status, "Error to processing URL")
    end if
end procedure
```

Algoritmo 4 Extracción de Links de un documento HTML.

```
procedure GETLINKS(htmlFilePath, url)
    resultList ← empty
    linkList ← GETANCHORSLINKS(htmlFilePath)
    if length(linkList) > 0 then
        for each link in linkList do
            if link is absolute then
                if link not in resultList then          ▷ Avoid repeated links
                    resultList.append(link)
                end if
            else                                        ▷ the link is relative
                domain ← GETURLDOMAIN(url)
                newLink ← CONVERTRELATIVETOABSOLUTE(link, domain)
                if newlink not in resultList then       ▷ Avoid repeated links
                    resultList.append(newlink)
                end if
            end if
        end for
    end if
    return resultList
end procedure
```

A.2.3 Extracción de links de un documento HTML

En el Algoritmo 4 se muestra la forma de extraer todos los links de un archivo HTML. Para exploración y extracción de etiquetas *"<a href>"* del archivo HTML se utiliza la librería *"lxml"* y *"beautifulSoup"* de Python. La importancia de este algoritmo yace en el hecho de obtener los links del documento y verificar que sean links absolutos, no relativos. De ser relativos, se identifica el dominio de la URL para después concatenar el dominio con el link relativo y formar un link absoluto. Como resultado devuelve una lista de links absolutos o una lista vacía en caso de no encontrar ninguno.

A.3 Paquetes de Python utilizados

La aplicación esta escrita en el lenguaje Python para la versión 3.5. La mayoría de distribuciones Linux lo tienen pre-instalado. Se necesita instalar el paquete python-dev, el cual contiene los archivos de encabezado para la API de Python C. Escribir el siguiente comando en la terminal:

sudo ap-get install python-dev

Para el funcionamiento de la aplicación (Crawler) se necesitan de otros paquetes de Python. Para ello se necesita un administrador de paquetes llamado **pip**[1], para instalarlo escribir el siguiente comando en la terminal:

sudo apt-get install python-pip

A.3.1 Pip

Este administrador de paquetes cuenta con varios comandos, en este manual solo se muestra el funcionamiento de los comandos para instalar y desinstalar paquetes. Se necesitan privilegios de administrador (sudo).

- Instalar paquetes:
  sudo pip install nombrePaquete
- Desinstalar paquete:
  sudo pip uninstall nombrePaquete

[1] https://pip.pypa.io/en/stable

- Generar una lista de paquetes instalados con formato para instalarse vía archivo de requerimientos:
  sudo pip freeze > paquetes.txt

- Instalar un archivo de requerimientos:
  sudo pip install -r paquetes.txt

A.3.2 Paquetes Python requeridos

Para el funcionamiento de la aplicación se necesitan de algunos paquetes. Los siguientes paquetes no necesitan instalarse, ya que forman parte de la biblioteca estándar de Python 3.5.

- **sys** Este módulo proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete, ya que estas funciones interactúan fuertemente con el intérprete. Siempre está disponible.

- **os** Este módulo proporciona una forma portátil de utilizar la funcionalidad dependiente del sistema operativo.

- **urllib2** Módulo que define las funciones y clases que ayudan en la apertura de URLs (HTTP) y la autenticación implícita, redirecciones, cookies y más.

- **datetime** Proporciona un módulo con clases para manipular fechas y horas en formas simples y complejas.

- **subprocess** Este módulo permite obtener nuevos procesos, conectarse a tuberías (pipe) de entrada / salida / error, y obtener sus códigos de retorno, entre otras funcionalidades.

- **urlparse** Este módulo define una interfaz estándar para romper las cadenas URL (Uniform Resource Locator) en componentes (esquema de direccionamiento, ubicación de red, ruta de acceso, etc.) para combinar los componentes de nuevo en una cadena de URL y para convertir una *URL relativa* a una *URL absoluta* dada una *URL base*.

- **robotparser** Este módulo responde a preguntas sobre si una gente puede o no recuperar una dirección URL en el sitio Web que establece el archivo robots.txt.

- **time** Este módulo proporciona varias funciones relacionadas con el tiempo.

- **threading** Este módulo construye interfaces de threading de alto nivel en la parte superior del nivel más bajo del módulo thread; es decir se encarga de la administración de hilos.

Los paquetes que deben de ser instalados vía **pip** o **apt** son los siguientes:

- **lxml** Es la biblioteca más completa y fácil de usar para procesar XML y HTML en el lenguaje Python. Para instalar este paquete escribir uno de los siguientes comandos:
  - **sudo pip install lxml**
  - **sudo apt-get install python-lxml**
- **bs4** Un módulo de Python para la extracción de datos de los archivos HTML y XML. Funciona con un analizador (lxml) y formas idiomáticas de navegar, buscar y modificar el árbol de análisis. Cualquiera de los siguientes comandos se pueden utilizar para instalarlo.
  - **sudo pip install bs4**
  - **sudo apt-get install python-bs4**
- **PyGreSQL** Es un módulo que se conecta a una base de datos PostgreSQL. Incorpora la biblioteca de consultas de PostgreSQL para permitir el fácil uso de las poderosas funciones de PostgreSQL desde una secuencia de comandos Python. Para instalar el módulo escribir cualquiera de los siguientes comandos en la terminal:
  - **sudo pip install pygresql**
  - **sudo apt-get install python-pygresql**

Anexo B

Reconocimiento de Voz Automático

En esta sección se describe el Reconocimiento de Voz Automático (ASR: Automatic Speech Recognition) utilizado en este trabajo. Como se ha mencionando con anterioridad se emplea una librería en lenguaje Python llamada *SpeechRecognition*, este paquete contempla el uso de APIs para esta tarea como son CMU Sphinx y Google Speech Recognition. Además en la aplicación de IRS web se utiliza la interfaz SpeechRecognition de la Web Speech API. A continuación se describen estas herramientas y como instalarlas. La instalación de todas las librerías es para Pyhon 3.5 en un Sistema Operativo Linux, específicamente en Ubuntu 16.04, se requieren privilegios de administrador (sudo).

B.1 Librería SpeechRecognition

El paquete o librería utilizada es la SpeechRecognition 3.6.5 para realizar el reconocimiento de voz, con soporte para varios motores y APIs. En este trabajo solo se utilizó PocketSphinx (es Sphinx para plataformas embebidas) de CMU Sphinx y Google Speech Recognition. La forma de instalar esta librería es con los mismos comandos mencionados en el Anexo A. Para esta librería se tienen que instalar otros paquetes que son requeridos para su correcto funcionamiento, se requieren permisos de administrador (sudo) son los siguientes:

B.1.1 PyAudio 0.2.9+

Es un paquete para la manipulación del micrófono conectado al dispositivo, además esta librería depende de otras que también deben ser instaladas. La siguiente lista muestra la instalación de estos paquetes.

- **sudo apt-get install portaudio19-dev**
- **sudo apt-get install flac**
- **sudo pip3 install pyaudio**

B.1.2 PocketSphinx

Es un motor de reconocimiento de voz ligero, ajustado específicamente para dispositivos móviles, aunque funciona de igual manera en dispositivos de escritorio. PocketSphinx necesita de algunos paquetes extra, algunos son necesarios para la instalación y el funcionamiento de compiladores. Además se necesita SphinxBase como requisito para el funcionamiento de PocketSphinx. Se tiene que descargar los códigos fuentes de SphinxBase y PocketSphinx, después se tienen que compilar e instalar. A continuación se listan los comandos de instalación.

- **sudo apt-get install gcc automake autoconf libtool bison swig python-dev libpulse-dev git**
- Para instalar SphinxBase se tienen que seguir los siguientes pasos:
  - Crear un directorio con el nombre de *sphinx-source* normalmente el directorio se crea el la carpeta del usuario **$: cd /home/user/** con el comando **mkdir sphinx-source**.
  - **cd sphinx-source** entrar en el directorio creado.
  - **git clone https://github.com/cmusphinx/sphinxbase.git** se clona el repositiorio utilizando el comando *git*. Automáticamente se descarga una carpeta con el nombre *sphinxbase*.
  - **cd sphinxbase** acceder a la carpeta descargada *sphinxbase*.
  - **./autogen.sh** ejecutar el t *autogen.sh* que generará los archivos *Makefile* y otros archivos importantes para compilar y configurar e instalar *SphixBase*.
  - **make** este comando crea automáticamente los programas ejecutables y librerías de código fuente mediante la lectura de archivos llamados *Makefile* que especifican las dependencias que existen entre los archivos que componen el código fuente de un programa.
  - **sudo make install** es importante ejecutar este comando con privilegios de administrador (sudo). Este es el último comando necesario para instalar SphinxBase.

Una forma de comprobar si realmente se ha instalado *SphinxBase* es escribiendo **sphinx_** en una *terminal* y presionar la tecla *tab* para ver todas las opciones de lo que se acaba de instalar, como se muestra a continuación:

```
user@mypc:~$ sphinx_
user@mypc:~$ sphinx_cepview sphinx_cont_seg sphinx_fe sphinx_jsgf2fsg
sphinx_lm_convert sphinx_lm_eval sphinx_pitch
```

Una vez que se ha instalado correctamente *SphixBase* es el turno de instalar PocketSphinx, la instalación es muy similar. Seguir cada uno de los siguientes pasos:

- **cd /home/user/sphinx-source** ubicarse dentro de la carpeta de trabajo previamente creada.
- **git clone https://github.com/cmusphinx/pocketsphinx.git** clonar el repositorio. Se descarga y crea automáticamente una carpeta con el nombre pocketsphinx.
- **cd pocketsphinx** entrar en la carpeta recién descargada.
- **./autogen.sh** ejecutar el script.
- **make** crear archivos Makefile con el comando *make*.
- **sudo make install** compilar e instalar los archivos fuentes de *pocketsphinx*.

Se verifica que se haya instalado correctamente escribiendo en una *terminal* pocketsphinx_ y presionando la tecla **tab** para que muestra las opciones que inician de la misma forma.

```
user@mypc:~$ pocketsphinx_
user@mypc:~$ pocketsphinx_batch pocketsphinx_continuous
pocketsphinx_mdef_convert
```

B.1.3 SpeechRecognition 3.6.5

Al instalar todas las dependencias necesarias, los motores y APIs que se ocuparán en la librería SpeechRecognition se procede a instalar. Google Speech Recognition es instalado junto con esta librería, por lo que no es necesario instalar algo más.

sudo pip3 install SpeechRecognition se necesitan privilegios de administrador (sudo)

B.2 Reconocimiento de Voz

Para el reconocimiento de voz utilizando la librería SpeechRecognition se puede hacer de dos formas, utilizando el micrófono como fuente de entrada (véase el Algoritmo 5) o desde un archivo de audio (Algoritmo 6) regularmente en formato wav.

Algoritmo 5 ASR utilizando Sphinx y el micrófono como fuente de entrada.

```
 1: procedure RECOGNIZEMICROPHONEVOICE
 2:     import SpeechRecognition library
 3:     with MICROPHONE as source:
 4:         ADJUST_FOR_AMBIENT_NOISE(source)
 5:         Message "Microphone ready, say something:"
 6:         audio ← LISTEN(source)
 7:     data ← GETWAVDATA(audio)
 8:     WRITEINTOFILE(data)
 9:     result ← RECOGNIZESPHINX(audio, 'es-es')
10:     return result
11: end procedure
```

En el Algoritmo 5 en la línea 4 se ajusta dinámicamente el umbral de energía utilizando el audio de la fuente para tener en cuenta el ruido ambiental, calibrando el nivel de energía del ambiente. Debe utilizarse en períodos de audio sin voz, se detendrá antes si se detecta algún sonido de voz. Se puede ajustar el parámetro *duration* en el método, que es el número máximo de segundos que ajustará dinámicamente el umbral de energía, el valor de este debe ser al menos de 0.5 (segundos) para obtener una muestra representativa del ruido del medio ambiente.

El método *Listen(source)* de la línea 6 del Algoritmo 5 se encarga de grabar una frase de la fuente, esto se hace esperando hasta que el audio tenga una energía por encima del umbral de energía previamente establecido (i.e cuando el usuario ha empezado a hablar), luego graba hasta encontrar el umbral de pausa (i.e después de algunos segundos de no hablar o no hay más entrada de audio por parte de la fuente). El silencio final no está incluido.

En la línea 9 el método *Sphinx* realiza el reconocimiento de voz usando CMU Sphinx, el lenguaje para el reconocimiento esta dado por *'es-es'*. De forma predeterminada Sphinx transcribe las palabras que reconoce con la probabilidad más alta. El modelo de lenguaje y acústico *'es-es'* se describe en la siguiente sección.

En el Algoritmo 6 en la línea 3 la fuente proviene de un archivo de audio, este archivo debe estar en formato *wav*, *aiff* o *flac*. Si el archivo es una cadena, se interpreta como una

ruta a un archivo de audio en el sistema de archivos. De lo contrario, se toma como un objeto perteneciente a la clase archivo.

Algoritmo 6 ASR con Google Speech utilizando un archivo de audio como entrada.

```
1: procedure RECOGNIZEMICROPHONEVOICE
2:     import SpeechRecognition library
3:     with AUDIOFILE(audiofile) as source:
4:         audio ← RECORD(source, duration=5)
5:     result ← RECOGNIZEGOOGLE(audio, 'es-mx')
6:     return result
7: end procedure
```

El método *record(source, duration=5)* graba 5 segundos de audio del archivo que comienza al inicio del archivo. Si la duración no se específica, entonces grabará hasta que no haya más entrada de audio. En la línea 5 del Algoritmo 6 se realiza el reconocimiento de voz utilizando la API de Google, se necesita una llave (key) para utilizarla, aunque si no se específica una, se específica una llave genérica, está generalmente debería usarse solo para fines personales o de prueba, ya que puede ser revocada por Google en cualquier momento. El lenguaje para el reconocimiento se especifica como *'es-mx'*. Al igual que Sphinx regresa la transcripción de palabras más probable.

B.2.1 Modelos de Lenguaje y Acústico en Español

Estos modelos están en Sphinx de manera predeterminada en idioma Inglés *en-us*, para este trabajo se utiliza el modelo en Español *es-es*. Lo primero es descargar los archivos necesarios y configurarlos para ASR de Sphinx, a continuación se listan los pasos a seguir:

- Descargar los archivos de la página:
  https://cmusphinx.github.io/2010/08/voxforge-spanish-model-released/
  - **cmusphinx-es-5.2.tar.gz** contiene un folder con archivos y carpetas para la configuración.
  - **es-20k.lm.gz** contiene el archivo de modelo de lenguaje.
  - **es.dict** es el archivo de modelo acústico.
- **tar -xvzf cmusphinx-es-5.2.tar.gz** descomprimir el archivo

- **mv cmusphinx-es-5.2.tar.gz es-es** renombrarlo como *es-es*
- **cd es-es** acceder al directorio
  ls listar y verificar que el contenido sea el siguiente:
  etc model_parameters *README* **result scripts**
- **mv model_parameters acoustic-model** renombrar la carpeta por *acoustic-model*
- **cd acoustic-model** acceder al folder
  - **ls** listar el contenido, solo debería contener una carpeta:
    voxforge_es_sphinx.cd_ptm_4000
  - **cd voxforge_es_sphinx.cd_ptm_4000** acceder a la carpeta
  - **ls** el contenido es de ocho archivos:
    feat.params mdef means mixture_weights noisedict sendump
    transition_matrices variances
  - **mv * ../.** mover los ocho archivos un nivel atrás
  - **rmdir voxforge_es_sphinx.cd_ptm_4000** ahora eliminar el directorio vacío
- **gunzip -k es-20k.lm.gz** descomprimir el archivo del modelo de lenguaje, devuelve el archivo **es-20k.lm**
- **mv es-20k.lm language-model.lm.bin** renombrar el archivo modelo de lenguaje como *language-model.lm.bin*
- **mv language-model.lm.bin es-es/.** mover el modelo de lenguaje dentro de la carpeta *es-es*
- **mv es.dict pronounciation-dictionary.dict** renombrar el archivo del modelo acústico como *pronounciation-dictionary.dict*
- **mv pronounciation-dictionary.dict es-es/.** mover el modelo acústico dentro de la carpeta *es-es*

La carpeta con los modelos de lenguaje y archivos de configuración en Español debe copiarse dentro de la librería *SpeechRecognition* de Python, se requieren privilegios de administrador (sudo) en la ruta:
sudo cp -r es-es /usr/local/lib/python3.5/dist-packages/speech_recognition/pocketsphinx-data/

B.2.2 Web Speech API

Otro ASR utilizado en este trabajo es la interfaz **SpeechRecognition** de la Web Speech API, está API ha existido desde hace unos años, la especificación fue escrita en 2014, sin cambios significativos hasta ahora. A finales de 2015, Firefox (44+ y Firefox OS 2.5+) ha implementado la Web Speech API con el apoyo de Chrome (33+ con prefijo webkit).

Algoritmo 7 Reconocimiento de Voz utilizando la Web Speech API

```
1: procedure WEBASR
2:     recognition ← create instance of SpeechRecognition
3:     recognition.lang ← 'es-MX'
4:     recognition.maxAlternatives ← 1
5:     RECOGNITION.ONAUDIOSTART(True)
6:     result ← RECOGNITION.ONRESULT(event)
7:     return result
8: end procedure
```

La escritura de un programa en lenguaje JavaScript[1] utilizando la Web Speech API es sumamente simple, en la línea 2 del Algoritmo 7 se crea una instancia de la clase SpeechRecognition, con ella se pueden acceder a los parámetros y métodos de está interfaz. En la línea 3 se define el lenguaje, y en la 4 se establece el número máximo de alternativas devuelto.

El método de la línea 5 activa el micrófono en la página web, de forma predeterminada no esta disponible en los navegadores, aparece un mensaje solicitando al usuario que permita el uso del micrófono. En la página web se establece un botón que al pulsarlo activa el micrófono para la captura de audio.

La línea 6 se encarga de obtener la transcripción del audio recogido por el micrófono. La forma en la que se realiza esta acción no es transparente para el programador, ya que dentro del motor de renderizado WebKit se realiza la operación de transcripción, utilizando el la API de Speech Recognition de Google de forma interna. Regresa una lista ordenada en forma ascendente (por confianza) de transcripciones, en formato JSON. Pero en este caso se ha definido que solo devuelva un valor.

[1] https://developer.mozilla.org/es/docs/Web/JavaScript

Anexo C

Sintetizador de Voz

El sintetizador de voz (TTS: Text To Speech) que se utilizó en este trabajo de tesis tiene como base a festival, que es un framework que ofrece una API con diferentes módulos para la síntesis del habla, además es multilingüe. El Sistema Operativo de Robot (ROS) tiene este módulo para que el robot humanoide pueda expresarse por medio de audio.

C.1 Festival

Este sintetizador de voz se puede instalar en varios sistemas operativos, en este documento únicamente se muestra la instalación en un Sistema Operativo Ubuntu 14.04 y ROS Indigo, estas son las características de software que controlan al robot humanoide.

C.1.1 Sound-play

El TTS festival se instala directamente en ROS instalando el paquete *soundplay*, se necesitan privilegios de administrador (sudo), escribir en una terminal el siguiente comando:

sudo apt-get install ros-indigo-sound-play

Una forma de corroborar que se ha instalado correctamente es con una prueba con ROS. Los nodos involucrados son **soundplay_node.py** y **say.py** del paquete *soundplay*. Ejecutar los siguientes comandos en terminales separadas:

- **rosocore** sirve para levantar el nodo maestro de ROS.

- **rosrun sound_play soundplay_node.py** este nodo esta a la espera de recibir un mensaje del nodo *say.py*.
- **rosrun sound_play say.py "Greetings Humans. Take me to your leader."** este nodo es acompañado de la cadena que se desea reproduzca el TTS. De manera predeterminada la voz es en Inglés.

Se pueden instalar diferentes voces, la voz predeterminada se denomina **voice_kal_diphone** en Inglés y la única voz en Español Castellano se llama **voice_el_diphone**[1].

C.1.2 Instalación de nuevas voces en Festival

Para instalar nuevas voces existe un comando para visualizar una lista de las voces disponibles, escribir en la terminal el comando:

sudo apt-cache search –names-only festvox-*

Este despliega una lista como la que se muestra en la Tabla C.1, de la lista la voz **festvox-ellpc11k** es la voz en Español Castellano. El comando para instalar cualquier voz para festival es el siguiente:

sudo apt-get install festvox-ellpc11k

Una forma de conocer el nombre del módulo de voz que se ha instalado es desplegando los archivos con en comando *ls* como sigue:

- **ls /usr/share/festival/voices/** desplega los directorios con las voces instaladas:
  - english
  - spanish
- **ls /usr/share/festival/voices/english/** desplega el directorio con el nombre **kal_diphone**.
- **ls /usr/share/festival/voices/spanish/** desplega el directorio con el nombre **el_diphone**

[1] http://www.cstr.ed.ac.uk/projects/festival/manual/festival_24.html

Para utilizar el TTS Festival con la voz es español *el_diphone* se agrega al inicio palabra *voice_* con el nodo de prueba *say.py* se ejecuta como sigue:

rosrun sound_play say.py "Saludos a todos, sean bienvenidos" voice_el_diphone

Tabla C.1 Voces Disponibles del TTS Festival.

Name Diphone	**Description**
festvox-kallpc16k	Voz masculina norteamericana para festival, con muestreo a 16 kHz.
festvox-kallpc8k	American English male speaker for festival, 8khz sample rate
festvox-kdlpc16k	Voz masculina norteamericana para festival, con muestreo a 16 kHz.
festvox-kdlpc8k	American English male speaker for festival, 8khz sample rate
festvox-ca-ona-hts	Catalan female speaker for festival, 16kHz HTS
festvox-czech-dita	Czech adult female speaker "dita" for Festival
festvox-czech-krb	Czech child male speaker "krb" for Festival
festvox-czech-machac	Czech adult male speaker "machac" for Festival
festvox-czech-ph	Czech male speaker for Festival
festvox-hi-nsk	Hindi male speaker for festival
festvox-italp16k	Italian female speaker for Festival
festvox-itapc16k	Italian male speaker for Festival
festvox-mr-nsk	Marathi male speaker for festival
festvox-ru	Russian male speaker for Festival
festvox-suopuhe-common	Common files for Festival Finnish speakers
festvox-suopuhe-lj	Finnish female speaker for Festival
festvox-suopuhe-mv	Finnish male speaker for festival
festvox-te-nsk	Telugu (te) male speaker for festival
festvox-don	voz masculina minimalista en inglés británico para festival
festvox-ellpc11k	Voz masculina de español-castellano para Festival
festvox-rablpc16k	voz masculina en inglés británico para festival, ratio 16khz
festvox-rablpc8k	voz masculina en inglés británico para festival, ratio 8khz
festvox-en1	mbrola-en1 voice support for festival
festvox-us1	mbrola-us1 voice support for festival
festvox-us2	mbrola-us2 voice support for festival
festvox-us3	mbrola-us3 voice support for festival

Festival cuenta con una consola para realizar diferentes operaciones con el Sintetizador, la forma de entrar es escribiendo en una terminal *festival*. Una vez dentro se puede establecer la voz y reproducir cadenas de texto como se muestra a continuación:

- *festival*> **(voice_el_diphone)** establecer la voz.

- *festival*> **(SayText "Hola saludos a todos, adios")**
- *festival*> **(help)** despliega una lista de comandos que se pueden utilizar en la consola.
- *festival*> **(quit)** para salir de la consola.

Existe una forma más para utilizar el TTS, es con el comando **text2wave** que recibe una cadena o un archivo de texto (con el formato TTS), además se especifíca un archivo wav de salida. Esta es la forma en la cuál funciona el TTS del LKE. Se muestra a continuación la sintaxis:

- **text2wave -o audio.wav file.txt > log.txt** proceso del TTS
- **aplay audio.wav** reproducir el archivo de audio

El comando **text2wave** recibe como entrada a **file.txt** y se transforma el texto en audio en el archivo **audio.wav**, el archivo **log.txt** guarda mensajes desplegados por el comando y agiliza su proceso.