Módulo de Recopilación y Almacenamiento de Noticias para SACAN

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Yunior Chacón Sorio

Tutores:

MSc. Dunnia Castillo Galán

Ing. Ramón Morales Álvarez

La Habana

Resumen

La gran cantidad de información que existe en internet crece a un ritmo increíble en los últimos años, donde diariamente se publican miles y miles de noticias en todo el mundo en diferentes diarios y revistas digitales. Los usuarios en su día a día gastan tiempo en consultarla, la Universidad de las Ciencias Informáticas hace algunos años desarrolló un sistema capaz de recuperar noticias de la web, almacenarlas y clasificarlas.

El presente trabajo tiene como fin desarrollar un sistema capaz de buscar estas noticias de manera automática lo que permitirá a los especialistas disminuir el tiempo que se le dedica a esta tarea, también permitirá su almacenamiento en base de dato para ser procesadas en lenguaje natural para su posterior clasificación.

Índice de contenido

Introducción

En la actualidad las noticias forman parte de la vida cotidiana de las personas al encender el televisor, leer el periódico, pero no se puede dejar de mencionar la internet. Google Trends o como su nombre indica búsqueda de tendencias de google es una herramienta de Google Labs que muestra los términos de búsqueda más populares. La misma permite mostrar con cuanta frecuencia se realiza una búsqueda de un término en particular en diferentes regiones del mundo y en varios idiomas. (1) En junio del 2016 en el *top* de categorías por tiempo pasado online en cualquier dispositivo indicó que mensualmente la categoría de *News/Information* en minutos mensual fue de 53412. (2) Statista es un portal de estadísticas que ofrece más de un millón de datos de más de 22500 fuentes. En la estadística de aplicaciones móviles con más rápido crecimiento por categoría en el 2017 la categoría de *News/Magazines* tuvo un 20% más de crecimiento.

Debido a esto se puede extraer la importancia que tiene en la actualidad la prensa digital, así como la gran variedad de noticias que son publicadas cada segundo en la red. Este gran volumen de información ha crecido exponencialmente en los últimos años con la popularización del uso de la internet, por lo que surge la necesidad de buscar una alternativa para organizar y recuperar la información.

Ante esta situación debido a la gran cantidad de noticias disponible en línea, surge el uso de técnicas de Recuperación de Información de sus siglas en inglés (IR) *Information Recovery* para evitar agobiar a los usuarios. Los principales objetivos de estos métodos son: reducir el tiempo invertido al leer los artículos, evitar la redundancia y proporcionar capacidad en la búsqueda por temáticas. Por lo cual, la recuperación y la extracción de artículos de noticias se han convertido en puntos clave.

La falta de estándares de publicación de noticias, el contenido heterogéneo de los sitios web y la gran cantidad de formatos de publicación hacen que la extracción de noticias de la web sea un problema abierto. Los enfoques de recuperación y extracción de noticias se enfrentan a dos problemas: primero la identificación de páginas de artículos de noticias dentro de una colección de documentos heterogéneos en la web donde hay muchos contenidos no deseados (por ejemplo, páginas de secciones, titulares y anuncios dados un documento identificado como una página de noticias), y segundo la extracción de los campos del artículo, es decir, el título, el cuerpo y la imagen, si está presente.

Una vez posible la obtención de los metadatos es necesario utilizar técnicas de Procesamiento de Lenguaje Natural (PLN).(3) Las tecnologías lingüísticas están ganando terreno en este mundo cada vez más digital, y poco a poco se extiende su uso en los sectores profesionales con el objetivo de descubrir, clasificar, organizar o buscar contenido de forma automática, lo cual puede redundar en un uso más eficiente del tiempo, una reducción de gastos y una toma de decisiones más ágil en las organizaciones.

Una de las aplicaciones del Procesamiento de Lenguaje Natural (PLN) son los *Named-entity recognitio*n (NER) reconocimiento de entidades nombradas. Como su nombre lo indica, NER detecta algunas entidades como son: personas, localizaciones, organizaciones o marcas. (4) El NER utiliza la tecnología de *Machine Learning*, reglas y corpus lingüísticos.

Los *Named Entity Recognition*, (NER) son herramientas automáticas de PLN que ayudan a entender el qué, quién y dónde de una serie de documentos. Emergieron como suplente de la extracción de información principal de los textos, se pensó: “si al menos tengo qué se ha hecho, quién lo ha hecho y dónde, tendré información apreciable de un documento”.

En Cuba existen varios centros y proyectos que utilizan herramientas para la recuperación de la información. Una de estas herramientas es el Sistema de Almacenamiento y Clasificación de Noticias (SACAN), el cual fue desarrollado en la Universidad de las Ciencias Informáticas, específicamente en el Centro de Ideoinformática con el objetivo de ayudar en la recuperación de noticias, y actualmente se encuentra operativo. SACAN es utilizado por el equipo de la Mesa Redonda para almacenar en una base de datos noticias extraídas por parte de los especialistas, pero presenta algunas deficiencias como: la búsqueda y clasificación manual de noticias, lo que resulta tedioso al tener que copiar cada información y definir detalles como: fecha, dónde se publicó el artículo, las palabras clave así como la clasificación por temática y descriptores, lo cual hace que se consuma gran cantidad de tiempo en esta actividad.

Por lo anteriormente planteado se propone como **problema de investigación**: ¿Cómo facilitar el proceso de obtención y almacenamiento de noticias para el sistema SACAN?

Como **objeto de estudio**: el proceso de recuperación de información.

Se plantea como **objetivo general**: Desarrollar un sistema que permita la obtención automática de noticias, así como su procesamiento y almacenamiento.

Para dar cumplimiento al objetivo general se proponen los siguientes **objetivos específicos**:

1. Realizar un estudio del estado del arte de diferentes sistemas homólogos para la recuperación de noticias.
2. Definir la tecnología a utilizar para la recuperación y extracción de noticias.
3. Definir la estructura para el almacenamiento de noticias en la base de datos.
4. Definir las funcionalidades del sistema de recuperación de noticias.
5. Implementar el sistema de recuperación de noticias.
6. Realizar pruebas de validación al sistema de recuperación de noticias.

Se selecciona como **campo de acción:** la recuperación automática de noticias en la web.

Para guiar la investigación se planteó la siguiente **idea a defender:** El desarrollo de un sistema automático permitirá minimizar el tiempo en la búsqueda y extracción de noticias de la web.

Para dar cumplimiento al objetivo general se propusieron las siguientes **tareas de investigación**:

1. Caracterización de las estructuras actuales en el diseño de páginas web.
2. Identificación de las tecnologías y herramientas actuales en la recuperación de la información en la web.
3. Identificación de algoritmos y métodos actuales que permitan la extracción del contenido de la noticia.
4. Evaluación de los métodos anteriormente identificados y la utilización de alguno de ellos o de desarrollo propio.
5. Caracterización de la estructura de noticias y sus elementos.
6. Definición de los requisitos funcionales y no funcionales de la aplicación.
7. Definición de una estructura para almacenar las noticias en las bases de datos.
8. Realización de pruebas a la propuesta de solución.

En pos de cumplir las tareas anteriores se utilizaron diferentes **métodos de investigación**:

Métodos Teóricos

Histórico – Lógico: Posibilitó un análisis profundo sobre la evolución y desarrollo de técnicas y algoritmos en la recuperación de noticias.

Analítico – Sintético: Aportó la posibilidad de analizar las diferentes fuentes bibliográficas buscando la esencia y rasgos que caracterizan el proceso de extracción y clasificación de noticias.

Inductivo – Deductivo: Posibilitó un desarrollo del conocimiento más generalizado del análisis de las diferentes formas del proceso de extracción y clasificación de noticias para determinar cuál es la solución más viable al problema.

Modelación: Permitió confeccionar los diferentes artefactos de ingeniería de software para establecer la base para el desarrollo de la aplicación.

Métodos Empíricos

Entrevista y Observación: Permitieron identificar las principales funcionalidades del sistema para la propuesta de solución mediante los análisis de los datos obtenidos.

Estructura del documento

Capítulo 1: Fundamentación Teórica: Durante este capítulo se mencionan los diferentes conceptos asociados a la investigación, lo cual posibilita un mejor entendimiento de la problemática planteada, así como las principales herramientas y metodologías en el proceso de extracción y clasificación de noticias.

Capítulo 2: Análisis y presentación de la propuesta de solución: Durante este capítulo se realizan los levantamientos de requisitos del software, así como se describe la propuesta de solución mediante los artefactos de la ingeniería de software.

Capítulo 3: Construcción y evaluación de la solución: Durante este capítulo se describe el proceso de desarrollo del software, arquitectura, implementación de la solución y validación de las funcionalidades de la propuesta de solución.

Capítulo 1 Fundamentación Teórica

En este capítulo se aprecian los aspectos teóricos que fundamentan el desarrollo del sistema propuesto. Se explican las tecnologías y herramientas usadas para su proceso. Así como las técnicas actuales en la recuperación de información y los mecanismos para la extracción y procesamiento de texto. También se da una panorámica de herramientas existentes en la recuperación de información y se explican conceptos básicos para obtener una mayor apreciación del contenido de este trabajo.

1.1. Conceptos asociados al dominio del problema.

Recuperación de la información: La recuperación de información es el conjunto de actividades destinadas a proveer la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros.

Van Rijsbergen dice que «[...] la recuperación de información está relacionada con la recuperación de aquellos documentos que sean probablemente relevantes para la necesidad de información del usuario expresada en una petición.»

Gerard Salton expresa que «[...] la recuperación de información está relacionada con la representación, almacenamiento, organización y acceso a items de información.»

Ricardo Baeza–Yates, señala que «[...] el área de recuperación de información ha crecido mucho más allá de sus objetivos primarios de indización de texto y búsqueda de documentos útiles en una colección.»

Charles Meadow menciona que «[...] la recuperación de información es un proceso de comunicación. Es un medio por el que los usuarios de un sistema o servicio de información pueden encontrar los documentos, registros, imágenes gráficas, o registros de sonido que satisfagan sus necesidades o intereses.»

Frederick W. Lancaster indica que «[...] la recuperación de información, tal y como se utiliza habitualmente, es sinónimo de búsqueda de literatura; es el proceso de buscar en una colección de documentos (utilizando el término documento en su más amplio sentido) para identificar aquellos que tratan de un determinado tema.»

Más recientemente, manifiesta que [...] en la actualidad la recuperación de información "convencional" significa la búsqueda *online* en bases de datos electrónicas, de forma interactiva y en tiempo real. Normalmente, esto implica que el usuario construye una estrategia de búsqueda usando términos con distintas relaciónes lógicas (booleanas) y que el programa de búsqueda simplemente divide la base de datos en dos conjuntos: elementos recuperados y elementos no recuperados.»

Stephen Harter habla de una visión más restringida de la recuperación de información al considerar solamente los sistemas de recuperación de información en línea accesibles públicamente; [...] recuperación de información *online*, o búsqueda *online*, es un proceso en el que un ser humano utiliza un terminal informático para interactuar con un servicio de búsqueda, en un intento de satisfacer una necesidad de información.»

Noticia: derivado del latín *notitĭa*, el concepto de noticia da nombre al contenido de una información que nunca antes había sido comunicada. En otras palabras, la noticia constituye un saber o un conocimiento nuevo.

En los medios de comunicación masivos, por noticia se entiende a un texto o un testimonio que le permite al público estar al tanto de un episodio novedoso**,** reciente o fuera de lo común que se ha desarrollado en una comunidad específica o en un contexto particular, lo que amerita su difusión.

Tarea de recuperación: aquellas rutinas algorítmicas ejecutadas por el sistema de información en respuesta a una solicitud del usuario. (BAEZA YATES, R.; RIBEIRO -NETO, B. 1999).

cURL: es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos. Soporta los protocolos FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP, entre otros. La primera versión se publicó en 1997 y se basó en una pequeña herramienta llamada *httpget.* El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es una herramienta válida para simular las acciones de usuarios en un navegador web.

Algoritmo de recuperación: es el conjunto de métodos documentales, rutinas de tratamiento de información y procedimientos automáticos de tipo matemático-estadístico, ya predefinido en el funcionamiento de un programa informático, tales como la depuración, indexación, comparación de consultas, aplicación de modelos de recuperación, representación, evaluación y análisis necesarios para que el sistema de información satisfaga las necesidades de información del usuario. El orden en que se ejecutan, la casuística de la consulta y la experiencia del usuario, son factores que influyen en la ejecución de los algoritmos de recuperación, generando un grado de variabilidad en los resultados obtenidos. (5)

Filtrado: proceso de refinamiento y perfección de la consulta del usuario por el que se delimita, especifica o amplía la búsqueda original, una vez que los resultados de la búsqueda satisfacen parcialmente la demanda informativa del usuario. (6)

Coincidencia exacta: es el mecanismo por el cual sólo los documentos que satisfacen algunos criterios y rasgos bien especificados en la consulta son recuperados y devueltos al usuario como una respuesta unívoca, cumpliéndose al 100% en sus expectativas. (6)

Recuperación de datos: la recuperación de elementos (tuplas, los objetos, páginas Web, documentos) cuyo contenido cumple los requisitos especificados en una consulta de usuario basada en expresión regular o por coincidencia de patrones. (BAEZA YATES, R.; RIBEIRO-NETO, B. 1999).

Minería de datos: la extracción de nuevos datos, documentos o información parcial de cualquier tipo, mediante métodos de *crawling*. (BAEZA YATES, R.; RIBEIRO-NETO, B. 1999).

Clustering: es la agrupación de documentos que satisfagan un conjunto de propiedades comunes. El objetivo es aunar aquellos documentos que están relacionados entre sí. El *clustering* puede ser utilizado, por ejemplo, para expandir una consulta de usuario con nuevos términos propios del contexto de los documentos recuperados. (BAEZA YATES, R.; RIBEIRO -NETO, B. 1999).

1.2. Evolución del significado de la recuperación de la información.

La recuperación de la información se limitaba a los documentos escritos, pero esto se redefinió con el surgimiento de los diferentes materiales multimedia, fue así que los buscadores de información de internet dejaron solamente de buscar información en formato de texto para introducir términos como imágenes, videos o audios, debido a estos términos la recuperación de textos, recuperación documental y recuperación de información son tratados equivalentemente. En la actualidad se está intentando migrar desde la recuperación de documentos a la recuperación de pregunta – respuesta, que responde con un dato concreto y no con el conjunto de documentos que tienen estos datos.

La información responde a que es algún tipo de elemento y las propiedades que lo describe, pero solo una parte de la información indica cómo se elabora o desarrolla un proceso, este tipo de información básicamente es conocimiento. El conocimiento implica dos cuestiones fundamentales: la existencia de un fin y una relación con otra información de un sistema para lograr un objetivo. Así el desarrollo de ontologías, agentes inteligentes y la inteligencia artificial ha propiciado un cambio hacia la recuperación de la información.

Los modelos de recuperación tratan de calcular el grado en que determinado elemento de información responde a determinada consulta. En general esto se consigue calculando los coeficientes de similitud (Coseno, Phi, etc.). Los tres modelos más utilizados son:

- Booleano: se crea un conjunto con los elementos de la consulta y otro con los documentos, para medir la correspondencia.
- Vectorial: en el que la consulta y los términos del documento se representan mediante dos vectores, y se mide el grado en que ambos vectores divergen.
- Probabilístico: se calcula la probabilidad en que el documento responde a la consulta. Frecuentemente utiliza retroalimentación. La retroalimentación se basa en que el usuario indique los documentos más semejantes a su respuesta idónea, para así reformular la consulta.

1.3. Técnicas y algoritmos en la recuperación de la información.

Técnicas avanzadas de recuperación de información

Las técnicas avanzadas de recuperación de información son todos aquellos procesos destinados a la recuperación de información, desde la generación de las colecciones, su depuración, indexado, tratamiento textual, clasificación, almacenamiento, recuperación mediante modelos booleanos, vectoriales, probabilísticos, basados en el lenguaje, así como todos aquellos elementos que inciden en cualquier aspecto relacionado como por ejemplo el interfaz de consulta, el comportamiento del usuario, la retroalimentación de las consultas y la representación de la información. La recuperación de información no puede concebirse como un elemento único e indisoluble como por ejemplo única y exclusivamente un modelo o algoritmo de recuperación. La recuperación de información debe estudiarse como un compendio de procesos altamente interrelacionados que conforman una verdadera cadena documental en recuperación de información. (5)

1.4. El proceso de rastreo

Para que la recuperación de información tenga lugar, es necesario comenzar a estudiar los procesos de *webcrawling*. Un *webcrawler* es un programa especializado en el análisis y rastreo de sitios, páginas y recursos de la web, de forma automática o semiautomática, recuperando como objeto principal, todos los enlaces o lo que es lo mismo el tejido hipertextual de la web. De esta forma es posible determinar que una página web contiene cinco enlaces de los que se obtienen a su vez diez. Este fenómeno se le conoce como nivel de profundidad del análisis. Cada enlace extraído con el *webcrawler* es analizado para obtener además terceros enlaces que son igualmente analizados, desentrañando una estructura escalar y jerárquica que puede ser almacenada y procesada por el programa. De esta forma, se pueden realizar análisis de sitios web con distintos niveles de profundidad y poder obtener el número total de enlaces para cada nivel, el número de enlaces internos, externos, documentos ofimáticos, etc.

De hecho, los grandes buscadores emplean miles de programas *webcrawler* para generar un mapa de la web que es utilizado por los usuarios para recuperar cualquier contenido enlazado y publicado en un servidor web remoto. No obstante, un *webcrawler* no actúa por sí solo. Requiere de una lista de enlaces de dominios, sitios o páginas web con las que trabajar inicialmente. Esta lista de páginas web objetivo, se denomina *seed* o semilla. A partir de los enlaces indicados en la semilla el *webcrawler* comienza un análisis de cada página uno a uno, recuperando todos sus contenidos. Inicialmente el contenido recuperado de las páginas web es el código fuente completo de la misma. (7)

Este código es fundamentalmente HTML. Dentro de la codificación HTML, se encuentra el texto de interés para su indexación y su recuperación efectiva. Para que tenga lugar el proceso de indexado se hace obligada la eliminación del código HTML, a fin de obtener el texto limpio de códigos. Una vez se tiene el texto completo de la página se realiza un proceso denominado tokenización que toma su nombre del inglés *token* cuyo significado es fichas o muestras. El proceso consiste en la división del texto en elementos más pequeños, dicho de otra forma, en palabras. De hecho, para poder tratar adecuadamente el texto de la página web resulta imprescindible tratar palabra por palabra para aplicar cambios sustanciales denominados normalización y depuración del texto. Ello consiste en la transformación de vocales y consonantes acentuadas en su equivalente sin acentuar, la transliteración de caracteres en otros idiomas, el escapado de comillas simples, dobles, apóstrofos, la supresión de caracteres ASCII no compatibles, la sustitución de caracteres especiales por su código equivalente, etc.

Resulta evidente que el análisis palabra por palabra facilita la ejecución de todos estos cambios, en vez de tenerlos que hacer en bloque, ya que en muchas ocasiones la identificación de tales casos de normalización resulta bastante compleja. Además, palabra por palabra también se identifica de forma más sencilla si el término resulta ser una palabra vacía. Las palabras vacías, también denominadas *stopwords*, son términos que carecen de carga semántica y que no tienen una validez clara a la hora de recuperar información, dada su alta frecuencia de aparición y poca representatividad del contenido. También son eliminados ciertos verbos y adverbios demasiado comunes como para ofrecer una representatividad y precisión de los contenidos de la página web. Finalmente, también se lleva a cabo, en muchos casos, el proceso de reducción morfológica o *stemming*. En este caso se consigue comprimir el tamaño de las palabras y por ende de los textos de la página web, ya que la reducción morfológica consiste en la supresión de los prefijos, sufijos, géneros y desinencias de los términos del texto, con el objetivo de mejorar, la exhaustividad de la recuperación de los contenidos. Cuando el texto ha sido normalizado y depurado, se llevan a cabo una serie de procesos de tipo estadístico-matemáticos.

Un **sistema de recuperación de información** procesa archivos de registros y peticiones de información, e identifica y recupera de los archivos ciertos registros en respuesta a las peticiones de información. (SALTON, G. 1989). La recuperación de información se centra en la representación, almacenamiento, organización y acceso a elementos de información. Estos procesos deberían proporcionar al usuario la capacidad de acceder a la información que necesita. Sin embargo, existe un problema bastante importante en lo referente a la caracterización de las necesidades de información del usuario, que no suele ser fácil de solucionar. (SÁNCHEZ JIMÉNEZ, R. 2011).

Es el proceso por el cual las demandas informativas y documentales del usuario son resueltas en un sistema de información, compuesto por un corpus documental de volumen variable, cuyo tratamiento de indexación y almacenamiento hacen posible su estructuración, interrogación y representación. (BLÁZQUEZ, M. 2012). El proceso de *crawling* resulta de gran importancia para generar la colección o lo que es lo mismo la base de conocimiento para el sistema de recuperación de información. Ello es debido no sólo a la extracción de los textos obtenidos a través de la web, sino por la capacidad de diferenciar distintos tipos de contenidos en las páginas que se encuentran altamente estructurados. Éstos son los enlaces propiamente dichos, los enlaces a documentos ofimáticos, los enlaces a documentos audiovisuales, los enlaces a documentos gráficos, los metadatos, las meta-etiquetas, los títulos y titulares de la página, los párrafos, los canales de sindicación, el texto depurado y el código fuente original.

El proceso real de un programa *webcrawler* toma las direcciones URL especificadas en el archivo de semilla o seed.txt. Para cada enlace, se comprueba la validez del mismo, verificando que se encuentra activo mediante su ejecución vía socket con el puerto 80. Si el enlace no funciona, está roto o no se encuentra el contenido, el sistema lo deshecha, seleccionando un nuevo enlace de la semilla. En caso contrario, el *webcrawler* ejecuta una serie de funciones de tipo cURL, que permiten descargar el código fuente de la página del enlace. Dicho código fuente se encuentra en formato HTML y es almacenado en una variable que permita posteriormente la manipulación y extracción de los contenidos depositados dentro de ella.

En este sentido seleccionar la información de todo el código fuente puede realizarse de dos formas distintas. En primer lugar, empleando técnicas de expresiones regulares REGEXP que conlleva una mayor complejidad. En segundo lugar, usando la técnica DOM (*Document Object Model*) que permite crear un árbol jerárquico de la estructura de etiquetas HTML de la página web que fue almacenada en la variable. Ello permite, la extracción de metadatos, meta-etiquetas, enlaces, párrafos, titulares y demás elementos a través de un lenguaje de consulta, filtrado y selección denominado XPath. De esta manera, se distinguen todos sus elementos de forma ordenada e independiente, para ser finalmente procesados por la base de

datos del sistema. Dado que se trata de un *webcrawler*, el proceso no acaba hasta que finalizan todos los enlaces de la semilla y posteriormente hasta que no se analizan los enlaces derivados de cada enlace de la semilla, hasta que se alcance el nivel de profundidad tope para el análisis, tal como fuera configurado el programa.

1.5. Mecanismos de depuración de páginas web para la extracción y procesamiento de textos.

Cuando se generan colecciones de documentos basados en páginas web, los programas *webcrawler* efectúan un proceso de extracción del código fuente, mediante el empleo técnicas cURL y DOM. Al hacerlo no sólo se adquiere el texto sujeto a recuperación, sino todo el conjunto de etiquetas en formato HTML y CSS que lo acompañan. Si tales etiquetados no son eliminados, no se puede iniciar el procesamiento de la información y su correspondiente tratamiento. Por ello, se demuestra la importancia de aplicar mecanismos de depuración del código fuente, que facilite la extracción limpia de los textos, que serán la materia prima con la que se componen las colecciones sobre las que se recupera la información.

Preparación de la colección de documentos.

Preparar los documentos implica en esencia los procesos destacados a continuación. No obstante, pueden aplicarse otros derivados del procesamiento del lenguaje natural, analizadores sintácticos para el reconocimiento de las oraciones del texto, identificación de la naturaleza de los complementos y sintagmas de las frases, etc. Tales funciones se situarían entre el proceso de normalización de textos y el de indexación, puesto que antes de proceder a su almacenamiento, se requeriría del análisis semántico-sintagmático correspondiente.

Proceso de depuración.

- Depuración y supresión de código fuente.
- Identificación de casos especiales.

Normalización de textos.

- Tokenización.
- Conversión a minúsculas, eliminación de signos de puntuación y acentos.
- Eliminación de palabras vacías.
- Transliteración y reemplazo de caracteres especiales.

Indexación.

- Reducción morfológica.
- Almacenamiento.
- Fichero inverso.
- Fichero índice.

Para efectuar un proceso de depuración y limpieza del contenido, es preciso identificar qué bloque de HTML contiene la información central del documento. En esencia el *webcrawler* debe ser capaz de determinar qué elementos corresponden al interfaz visual, la navegación hipertextual, los menús, la publicidad, los banners, así como cualquier otro elemento accesorio que no forma parte del contenido. (8)

Por regla general, cuando se trata de depurar el texto central de una página web, el *webcrawler* actúa dentro del dominio del cuerpo del sitio web <body></body>, eliminando todas las etiquetas <table></table>, <iframe></iframe>. A esta primera medida se le suman otras muchas relativas, a la identificación de contenidos esenciales, como por ejemplo el encabezamiento, el resumen, los párrafos de contenido, claramente identificados por etiquetas <h1></h1>, <h2></h2>, <h3></h3>, , <blockquote></blockquote>, <cite></cite>. Aun así, la idiosincrasia y variedad de las páginas web, obligan a un reconocimiento de casos, estilos. Este es el caso del empleo de la etiqueta que corresponde a las capas en HTML y que pueden adquirir cualquier estilo en CSS que por ejemplo imite el resultado de una etiqueta de encabezado <h1></h1>. Este tipo de situaciones son muy comunes y hacen que el *webcrawler* contemple el análisis de los estilos con que se editan las informaciones. Además, existe el problema añadido de utilizar recursivamente el anidamiento de distintos elementos en HTML, lo que dificulta aún el proceso de depuración.

Tokenización: es el proceso que descompone los textos de una colección en sus unidades mínimas, las palabras o términos propiamente dichos. A tales elementos se les denomina "*tokens*" que conforman una lista de ítems que se utiliza para su análisis PNL (Procesamiento del lenguaje natural), estadístico, lingüístico, almacenamiento y posterior recuperación de información. Para llevar a cabo tal proceso, se utilizan los espacios entre las palabras del texto como divisores de los distintos "*tokens*".

Conversión a minúsculas, eliminación de signos de puntuación y acentos: para permitir un proceso de indexación limpio, previamente es necesario convertir el texto a minúsculas y eliminar todos los signos de puntuación del texto. De hecho, este punto de la depuración simplifica a *posteriori* el reconocimiento y proceso de cruzado de la consulta del usuario, permitiendo que independientemente de que escribiera su

consulta correctamente, pueda ser recuperada bajo cualquier circunstancia. De esta forma el sistema de recuperación identifica mediante codificación hexadecimal qué caracteres debe reemplazar, modificar o en su defecto eliminar. Es habitual que tales programas integren la referencia completa de caracteres, denominadas también como "tablas de equivalencias entre caracteres".

Transliteración y reemplazo de caracteres especiales: En muchos casos, la conversión de un texto a minúsculas, la supresión de acentos y signos de puntuación no es suficiente. En muchos casos el idioma en el que está escrito el documento o las particularidades del texto implican el uso de caracteres especiales que requieren transliteración o reemplazo por otro carácter más sencillo y equivalente en el teclado estándar. Estos procesos resultan complejos, puesto que en todo momento se debe asegurar la identificación del carácter original.

Eliminación de palabras vacías: Las palabras vacías, irrelevantes o "*stopwords*" son aquellas que por sí solas carecen de significación y que, por su altísima frecuencia de aparición en los textos, generan un ruido innecesario para la recuperación de información. La eliminación de estos términos (preposiciones, artículos determinados, artículos indeterminados, pronombres, conjunciones, contracciones y ciertos verbos y adverbios) mejora la afinación en los modelos de recuperación. Los estudios correspondientes a este fenómeno fueron iniciados por Hans Peter Luhn en 1958 con su investigación sobre el índice KWIC, una técnica de indexación que organizaba las palabras según su consideración como claves para la recuperación o no de la información, teniendo en cuenta el contexto del documento.

Este proceso derivó en la acuñación del término "palabra vacía" para referirse a aquellas con un bajo poder discriminatorio y representativo del contenido del documento. Los análisis estadísticos efectuados por Luhn, demostraron que la indexación era un proceso más rápido, cuando se prescinde de tales términos y favoreciendo la economía de espacio requerido para el almacenamiento de la información. También se demostró, que entre un 30 y un 50% de las palabras de un texto corresponden a tal categoría. De hecho y pese a ser práctica habitual (RIJSBERGEN, C.J. 1979), hasta nuestros días, se siguen utilizando listas de palabras vacías para la depuración de los textos. No obstante, la técnica de eliminación de palabras vacías, se viene suavizando, debido a la introducción de técnicas de PNL (Procesamiento del Lenguaje Natural) que tienen en cuenta la significación de tales palabras cuando están acompañadas de sustantivos, en casos en los que no pueden ser separadas o eliminadas por conformar una denominación propia, así como por pérdidas en la significación semántica de un sintagma, frase o palabra.

1.6. El proceso de indexación.

Indexar es la acción de construir un fichero inverso de forma automática o manual. Este proceso es necesario para localizar y recuperar rápidamente cada uno de los términos del texto de un documento. Esto significa que a cada palabra se le asigna un identificador del documento en el que aparece, un indicador de la posición que ocupa en el texto (párrafo, línea, número de carácter de inicio) y un número de identificación para ese término propiamente dicho (único e irrepetible). De esta forma se conoce la posición exacta de cada término en los documentos de la colección y posibilita el posterior análisis de frecuencias. (8)

Para llevar a cabo la indexación, tal como apunta (Baeza-Yates,1999), se requiere de una importante capacidad de computación y por ende existe gran dependencia de las prestaciones del hardware para llevarlo a cabo. Si el corpus documental es la propia red WWW, se necesitará un motor de indexación distribuida. Eso significa que existen decenas de equipos informáticos (servidores), trabajando con una agrupación de contenidos determinada, colaborando en común para generar un gran índice que será utilizado a la postre por el motor de búsqueda. La indexación, también deberá ser dinámica. Esta propiedad significa que ante un corpus cambiante como puede ser la web, se necesitan reindexaciones continuas de los contenidos y por ende su modificación y ampliación.

Todo ello justifica que, junto al proceso de indexación, se trate de reducir al máximo el tamaño de tales archivos o tablas de la base de datos para conseguir la mejor relación entre tiempo de ejecución de las consultas y exhaustividad del fichero inverso. A esta misión se la denomina "compresión de la indexación" y en ella se circunscriben los procesos de depuración, tales como la supresión de palabras vacías, la normalización de palabras, la transliteración de caracteres especiales y la reducción morfológica o "*stemming*".

La compresión de la indexación.

- Depuración de los textos de los documentos de la colección.
- Supresión de palabras vacías (primer filtro).
- Reducción morfológica.
    - *Stemming*.
    - Lematización.
- Supresión de palabras vacías (segundo filtro).
- La ley de Zipf y la frecuencia de aparición.
- Técnica de cortes de Luhn: Cut-on y Cut-off.

- Cálculo del punto de transición.

Reducción morfológica.

La reducción morfológica es el proceso por el cual se depuran todos los términos de un texto, reduciendo su número de caracteres, simplificando su forma original, género, número, desinencia, prefijo, sufijo en una forma de palabra más común o normalizada, debido a que la mayor parte de ellas tienen la misma significación semántica. Este proceso reduce el tamaño de los términos, del diccionario, fichero inverso y mejora el "*recall*" o exhaustividad de los resultados en la recuperación de información.

***Stemming*.**

Efectúa una reducción de las palabras a sus elementos mínimos con significado, las raíces de las palabras, de hecho "*Stem*" significa tallo o raíz. De esta forma, los procesos de *stemming*, acotan las terminaciones de las palabras a su forma más genérica o común.

La ley de Zipf y la frecuencia de aparición.

En 1949 el lingüista y científico George Kingsley Zipf, formuló la ley empírica (de tipo potencial) que lleva su nombre y con la que se determina que la frecuencia de cualquier palabra es inversamente proporcional a la posición que ocupa en la tabla de frecuencias. Esto significa que la palabra más frecuente de un texto tiene una frecuencia de aparición que dobla a la de la segunda palabra más frecuente. La segunda palabra más frecuente tendrá una frecuencia de aparición que dobla a la de la tercera palabra más frecuente. Dicho de otra forma, la frecuencia de aparición de una palabra es inversamente proporcional a su número de orden. Y de esta forma se repetiría el esquema potencial de la ley Zipf con todos los términos del texto.

Técnica de cortes de Luhn: Cut-on y Cut-off.

La expresión logarítmica de los términos de un documento muestra una curva pronunciada con los términos de altísima frecuencia de aparición y su inverso, aquellos términos de muy baja frecuencia de aparición. Este hecho, ya adelantado por Zipf, dio lugar al empleo de la técnica de cortes que propuso Luhn en 1958, conjetura por la que ya se venía intuyendo que los términos situados en los extremos del eje de abscisas y de ordenadas serán los que menos poder de resolución o representatividad tendrán para un determinado documento dentro de la colección. Consiste en la eliminación de los términos de altísima frecuencia de aparición (Cut-on) y términos de bajísima frecuencia de aparición (Cut-off), entre los que se incluyen los "Hápax", términos cuya frecuencia de aparición es la unidad. (9)

Cálculo del punto de transición.

Para averiguar de forma científica la estimación del corte superior "Cut-on" e inferior "Cut-off ", una de las soluciones más estudiadas es el cálculo del punto de transición o inflexión entre los términos generales y especializados, cuyas características pueden resumirse en:

Términos con alta frecuencia de aparición.

- Generales.
- Tienden a unificar el lenguaje.
- Proporcionan nexos de unión en torno al texto.
- Pulsión.
- Artículos, preposiciones, conjunciones, contracciones, pronombres, algunos adverbios y verbos.
- Rangos cercanos al origen del eje.
- Técnica de recorte aplicada: Cut-on.

Términos con baja frecuencia de aparición.

- Específicos.
- Tienden a diversificar el lenguaje.
- Detallan el contenido del texto.
- Repulsión.
- Terminología especializada de una determinada área de conocimiento, vocabulario técnico, científico, Hápax.
- Rangos situados en el extremo del origen.
- Técnica de recorte aplicada: Cut-off.

El punto de transición no es más que una frecuencia de aparición intermedia a partir de la cual, se determina el porcentaje de términos tendientes a la generalidad o a la especificidad (según se separen de la frecuencia de aparición del punto de transición) para determinar dónde efectuar los cortes.

1.7. Sistemas homólogos

PANDORA (Australia‟s Web Archive) es un desarrollo realizado íntegramente por Cran Consulting, S.L. que, basándose en el motor de indexación Full Text Lucene (desarrollo de dominio público bajo licencia de software libre de Apache), permite la indexación de archivos digitales, para su inclusión en una base de

datos documental, almacenados en repositorios ARC y WARC. Para realizar búsquedas en la base de datos documental.

Pandora posee un repositorio de documentos, permitiendo a sus usuarios la recuperación de los mismos a partir de avanzados criterios de búsqueda tales como tipo de documento, cabecera, la fecha o rango de fechas y/o términos contenidos en el texto completo de los documentos indexados. (10)

PADICAT es un sistema que se basa en la aplicación de una serie de programas informáticos que permiten la captura, el almacenamiento, la organización y el acceso permanente a las páginas Web publicadas en Internet. Posteriormente a la fase de análisis y test de software, los desarrolladores determinaron que utilizarían el programa informático Heritrix, el cual es el encargado de compilar las páginas Web tal y como las ve el usuario que navega por Internet y almacenarlas en archivos comprimidos en formato .ARC o WARC. El programa Heritrix se complementa con NutchWax, o bien la combinación de Hadoop y Wayback, que llevan a cabo unos procesos de indexación de la información compilada, permitiendo utilizar estos índices para localizar los recursos dentro de la colección mediante sus respectivas interfaces de consulta: Wera permite la búsqueda por palabras clave a través de los índices generados por NutchWax y Wayback permite la consulta directa por URL en los índices generados por Hadoop y el mismo Wayback. (Padicat, 2015). (11)

Herramientas	Rastreo	Indexación	Distribución
PANDORA	Nutch	Lucene	Privativa
Padicat	Heritrix	NutchWax	Privativa
Orión	Nutch	Solr	Libre

Tabla No.1 Caracterización de los sistemas homólogos. Elaboración propia.

Caracterización de los sistemas homólogos para la selección del marco de trabajo

Heritrix es un rastreador (o crawler) de ficheros web a través de internet. Su licencia es open-source y está escrito completamente en JAVA. Su interfaz de configuración es accesible usando un navegador web, haciéndolo muy versátil y cómodo de usar, aunque también puede ser lanzando desde línea de comandos. El objetivo principal de Heritrix es ser un "rastreador de archivo" obtener copias completas, precisas y profundas de los sitios web. Esta incluye obtener contenido gráfico y otro contenido no textual. Los recursos se almacenan exactamente como se recibieron: no truncamiento, cambios de codificación, cambios de

encabezado, etc. Los rastreos se inician, supervisan y ajustan a través de una interfaz de usuario web, lo que permite conocer qué URL deberían ser visitadas y cuáles no debería. En cuanto a Nutch algunas de las cosas que hace de manera diferente son: solo recupera y guarda contenido indexable, puede truncar o cambiar el formato del contenido según sea necesario, guarda el contenido en un formato de base de datos optimizado para indexación posterior; se ejecuta y controla desde una línea de comandos y se enfatiza el volumen de recolección bajo condiciones predeterminadas en lugar de la satisfacción exacta de los parámetros personalizados.

Teniendo en cuenta las características y beneficios expuestos anteriormente se decide seleccionar como herramienta de rastreo Scrapy, debido a que Heritrix es utilizado principalmente por la comunidad de archivo, mientras que Nutch se utiliza para un mayor número de casos de uso (por ejemplo, indexación, *scraping*) pero resulta deficiente a la hora de rastrear *tags* específicos de un sitio web por ejemplos los “divs” debido a que en su proceso de rastreo Nutch elimina todas las etiquitas html para darle paso al indexado. Scrapy cuenta en su arquitectura con *spiders* los cuales se les da un conjunto de instrucciones que permitirán extraer el contenido deseado.

1.8. Tecnologías, herramientas y metodologías asociadas al desarrollo de la solución.

Scrapy

Scrapy es un *framework* de código abierto escrito en Python para rastrear sitios web y extraer datos estructurados que se pueden utilizar para una amplia gama de aplicaciones útiles, como extracción de datos, procesamiento de información o archivo histórico. Aunque Scrapy se diseñó originalmente para rastrear la web, también se puede usar para extraer datos mediante API (como Amazon Associates Web Services) o como rastreador web de propósito general.

Scrapy hace que sea más fácil construir y escalar grandes proyectos de rastreo al permitir a los desarrolladores reutilizar su código. Scrapy también proporciona un *shell* de rastreo web que puede ser utilizado por los desarrolladores para probar sus suposiciones sobre el comportamiento de un sitio. (12)

En Scrapy las solicitudes se programan y procesan de forma asincrónica, esto significa que Scrapy no necesita esperar a que se complete y procese una solicitud, puede enviar otra solicitud o hacer otras cosas mientras tanto. Esto también significa que otras solicitudes pueden continuar incluso si alguna solicitud falla o se produce un error al manejarla.

Si bien esto le permite realizar rastreos muy rápidos (enviando múltiples solicitudes concurrentes al mismo tiempo de una manera tolerante a fallas) Scrapy también le da control sobre el rastreo a través de algunas configuraciones. Puede hacer cosas como establecer un retraso de descarga entre cada solicitud, limitar la cantidad de solicitudes concurrentes por dominio o por IP, e incluso usar una extensión de autoaceleración que intente resolver esto automáticamente.

Apache Solr

Solr es un motor de búsqueda de código abierto basado en la biblioteca Java del proyecto Lucene, con APIs en XML/HTTP y JSON, permite el resaltado de resultados, búsqueda por facetas, caché y una interfaz para su administración. Corre sobre un contenedor de servlets Java como Apache Tomcat en versiones anteriores a la 5.0, a partir de esta corre como servidor independiente. Sus principales características incluyen:

- búsquedas de texto completo
- resaltado de resultados
- *clustering* dinámico
- manejo de documentos ricos (como Word y PDF)
- es escalable
- permite realizar búsquedas distribuidas
- replicación de índices

La principal característica de Solr es su API estilo REST, ya que en vez de usar drivers o APIs programáticas para comunicarnos con Solr podemos hacer peticiones HTTP y obtener resultados en XML o JSON. (13)

Apache Tomcat

Apache Tomcat (también llamado Jakarta Tomcat o simplemente Tomcat) funciona como un contenedor de *servlets* desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los *servlets* y de *JavaServer Pages* (JSP) de *Oracle-Corporation* (aunque creado por *Sun Microsystems*). Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Sus principales características incluyen:

- configuración dinámica: fragmentos web (librerías pueden embeber partes de un web.xml de modo que no sea necesario añadirlos al web.xml de la aplicación.

- soporta anotaciones: los filtros, *Servlets* y *Listeners* pueden definirse por anotaciones, sin necesidad de crear un web.xml
- API *Servlet* extendida: permite añadir *Servlets* y *Filter* después del arranque de la aplicación
- modo embebido simplificado: Tomcat 7 incluye un API para esto (al estilo Jetty).
- mejoras en *logging*: con un formateador que escribe *logs* en una única línea.
- alias: permite incluir ficheros externos dentro de una aplicación, como directorios de imágenes o Javascript, de modo que puedan ser compartidos entre todas las aplicaciones. (14)

MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: licencia pública general y comercial por O*racle-Corporation* y está considerada como la base datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. Sus principales características incluyen:

- probado con un amplio rango de compiladores diferentes
- funciona en diferentes plataformas
- usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl
- uso completo de *multi-threaded* mediante *threads* del *kernel*, puede usarse fácilmente multipleCPUs si están disponible
- proporciona sistemas de almacenamiento transaccionales y no transaccionales
- usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice
- un sistema de reserva de memoria muy rápido basado en *threads*. (15)

Freeling

FreeLing es una biblioteca en C ++ que proporciona funcionalidades de análisis de lenguaje (análisis morfológico, detección de entidad nombrada, etiquetado PoS, análisis sintáctico, desambiguación de sentido de palabra, etiquetado de función semántica, etc.) para una variedad de idiomas (inglés, español, portugués, italiano, francés, alemán, ruso, catalán, gallego, croata, esloveno, entre otros). FreeLing también proporciona un front-end de línea de comandos que se puede utilizar para analizar textos y obtener el resultado en el formato deseado (XML, JSON, CoNLL). (16)

Visual Paradigm

Visual Paradigm es una herramienta CASE: (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación,

pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. La herramienta está diseñada para un amplio rango de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistemas y cualquiera que esté interesado en construir un sistema confiable a gran escala a través del uso del enfoque orientado a objetos. VP-UML da soporte a los últimos estándares de Java y notación UML y provee a la industria una gran variedad de código autogenerado y soporte para Java para realizar ingeniería inversa. Además, VP-UML permite una integración completa con herramientas de desarrollo como Eclipse, Borland, Netbeans para dar soporte en la etapa de implementación del software. La transición del análisis al diseño y luego a la implementación se realiza sin percances con este tipo de herramienta CASE, por lo que disminuye significantemente el esfuerzo empleado en todas las etapas del ciclo de vida del desarrollo. (17)

Netbeans

Netbeans es un entorno de desarrollo gratuito y de código abierto que permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: Java, PHP, Groovy, C/C++, HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. Suele dar soporte a casi todas las novedades en el lenguaje Java. Cualquier *preview* del lenguaje es rápidamente soportada por Netbeans. Tiene asistentes para la creación y configuración de distintos proyectos, incluida la elección de algunos *frameworks*. Buen editor de código, multilenguaje, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y semánticas, plantillas de código, *coding tips*, herramientas de refactorización. (18)

Java JDK 1.8

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web

Metodología AUP-UCI

Un área importante de la ingeniería de software es el desarrollo de metodologías y modelos. En la actualidad se han desarrollados disímiles esfuerzos encaminados al estudio de los métodos y técnicas para lograr una aplicación más eficiente de las metodologías en busca de sistemas más efectivos y de mayor calidad con

la documentación necesaria en perfecto orden y en el tiempo requerido. Una metodología impone un proceso de forma disciplinada sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema. Existen diferentes modelos y metodologías que han sido en los últimos años herramientas de apoyo para el desarrollo del software. Sommerville, en su libro Ingeniería del Software (2005) menciona que:

- Modelo de desarrollo de software: es una representación simplificada del proceso para el desarrollo de software, presentada desde una perspectiva específica.
- Metodología de desarrollo de software: es un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP.

Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

1.9. Conclusiones del capítulo

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema antes planteado llegando a las siguientes conclusiones:

1. Las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación permitieron una mayor comprensión de la propuesta de solución.
2. El análisis de los diferentes criterios y mecanismos para la indexación y el procesamiento de textos, permitió identificar las tendencias actuales para el desarrollo de la aplicación.
3. Se han analizado algunas de las soluciones existentes en el campo de la recuperación de información y después de hacer una caracterización de ellas se determina la necesidad de crear un sistema que responda a las necesidades del cliente.
4. El estudio realizado de las diferentes herramientas, tecnologías y leguajes de programación a utilizar, permitió la selección de los más adecuados para dar solución al problema planteado.

Capítulo 2 Análisis y diseño del módulo de recopilación y almacenamiento de noticias para el sistema SACAN

En este capítulo se describe la propuesta de solución para un sistema automatizado de recopilación de almacenamiento de noticias que permita optimizar el tiempo de búsqueda como el tratamiento de la información para generar descriptores que servirán para su clasificación por temática. Los requisitos funcionales y no funcionales del sistema, así como los casos de uso generados del mismo.

2.1. Propuesta de solución.

El proceso comienza con un administrador de procesos en segundo plano o como se le conoce demonio, llamado cron, el cual está presente en los sistemas unix, el mismo ejecuta procesos a intervalos regulares (cada un minuto, cada una semana, cada un mes) dependiendo de las especificaciones de su configuración. Este se ejecutará un comando en el que la herramienta Scrapy comenzará a rastrear los diferentes sitios web o fuentes que se encuentren en la configuración del *spider.*

El flujo de Scrapy está compuesto por varios componentes, este es controlado por un motor de ejecución, el motor obtiene las solicitudes iniciales para rastrear desde el *spider*, luego programa las solicitudes en el planificador y pide que se rastreen las próximas solicitudes. El planificador retorna las próximas solicitudes al motor y envía las solicitudes al descargador pasándolo a través del *middleware*. Una vez finalizada la descarga del sitio, el descargador genera una respuesta (con esa página) y la envía al motor pasando por el *middleware* del descargador (process_response ()). El motor recibe la respuesta del descargador y la envía al *spider* para procesarla, pasándola a través del *spider middleware* (process_spider_input ()). El spider procesa la respuesta y retorna los elementos rastreados y las peticiones nuevas a seguir mediante el *spider middleware* (process_spider_output ()). El motor envía los elementos procesados a través de los pipelines, luego envía las solicitudes procesadas al planificador y solicita las próximas peticiones para rastrear, este proceso se repite hasta que no quedan más peticiones en el planificador. Al finalizar el rastreo se exporta un archivo json con los elementos extraídos del sitio.

Ahora, el usuario puede buscar información sobre las páginas web rastreadas a través de Solr al importarse los datos generados por el archivo json. Estos metadatos serán procesados previamente antes de guardarse en base de dato específicamente el de contenido, donde se procesará con técnicas PLN (Procesamiento de Lenguaje Natural) para determinar cuáles son las palabras que ejercen de alguna manera más fuerza en

el texto las que servirán en un proceso más adelante como descriptores a partir de una temática determinada.

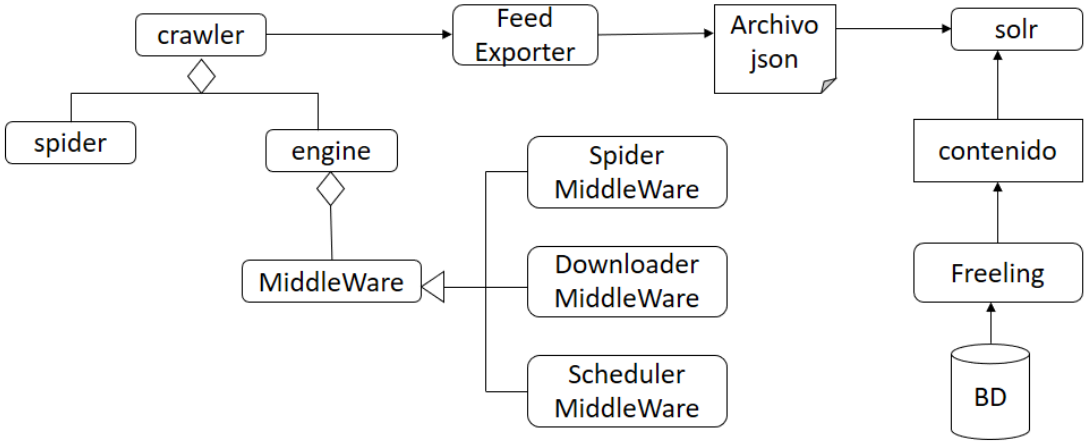


Fig1. Proceso de búsqueda y almacenamiento de noticas. Elaboración propia.

2.2. Modelo dominio.

Un Modelo *de* Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado en uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física.

Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo

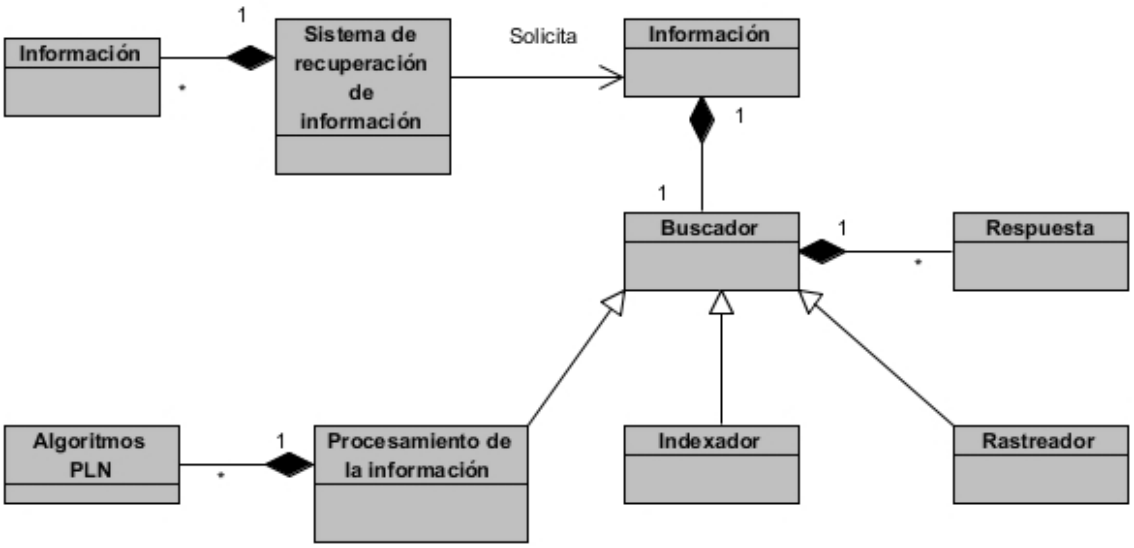


Fig2. Diagrama del modelo de dominio. Elaboración propia.

Información: la información está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje basado en un cierto fenómeno o ente. La información permite resolver problemas y tomar decisiones, ya que su aprovechamiento racional es la base del conocimiento.

Recuperación de información: proceso del descubrimiento automático de documentos relevantes de acuerdo a una cierta búsqueda. Documentos relevantes disponibles en la web tales como noticias electrónicas.

Buscador: sistema que opera indexando archivos y datos en la web para facilitar la búsqueda de los mismos respecto a términos y conceptos relevantes al usuario con sólo ingresar una palabra clave.

Indexador: tiene como propósito la elaboración de un índice que contenga de forma ordenada la información, esto con la finalidad de obtener resultados de forma sustancialmente más rápida y relevante al momento de realizar una búsqueda.

Rastreador: es un programa diseñado para explorar páginas Web en forma automática. La operación normal es que se le da al programa un grupo de direcciones iniciales, el *crawler* descarga estas direcciones, analiza las páginas y busca enlaces a páginas nuevas.

Procesar información: es en general, la acumulación y manipulación de elementos de datos para producir información significativa. También se les refiere a operaciones relativamente básicas como codificar, comparar, localizar, almacenar dando cuenta de la inteligencia humana y la capacidad para crear conocimiento.

Algoritmos PLN: para esta parte se estará utilizando la herramienta Freeling, que en su versión actual proporciona identificación de lenguaje, tokenización, división de oraciones, análisis morfológico, detección (Entidades Nombradas)NE y clasificación, reconocimiento de fechas / números / magnitudes físicas / moneda / razones, codificación fonética, etiquetado PoS, análisis superficial, análisis de dependencia, anotación de sentido basada en (WordNet)WN, desambiguación de sentido de palabra y resolución de correferencia.

Respuesta: Documentos indexados ya procesados para ser guardados en la base de datos.

2.3. Especificación de los requisitos de software.

Los requisitos se han convertido en un punto clave en el desarrollo de las aplicaciones informáticas. Un gran número de proyectos de software naufragan debido a una mala definición, especificación o administración de requisitos. Factores tales como requisitos incompletos o mal manejo de los cambios de los requisitos, llevan a proyectos completos al fracaso total. Acorde a la (IEEE: Standard Glossary of Software Engineering Terminology) podemos definir un requisito como: condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Requisitos funcionales: los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

Requisitos no funcionales: los requerimientos no funcionales son los que especifican criterios para evaluar la operación de un servicio de tecnología de información, en contraste con los requerimientos funcionales que especifican los comportamientos específicos.

La siguiente tabla describe los requisitos funcionales y su nivel de prioridad.

Nº	Nombre	Descripción	Prioridad
RF1	Programar CRON	Permite automatizar el proceso de rastreo de acorde a su configuración predefinida.	Media
RF2	Rastrear sitios web a partir de un semillero.	Permite al usuario a través de un semillero la especificación de diferentes fuentes para el rastreo.	Media

RF3	Indexar sitios web para la extracción de metadatos.	De acuerdo a los diferentes enlaces de los sitios web el sistema indexará su contenido para la extracción de los metadatos.	Media
RF4	Almacenar en base de datos los metadatos	El sistema almacenará en base de datos todos los metadatos antes definidos y encontrados en la indexación.	Alta
RF5	Comparar contenido en bases de datos	Para evitar la duplicidad de contenido de noticias a la hora de almacenar se comparará el artículo de la noticia contra fuente.	Alta
RF6	Analizar contenido en lenguaje natural	El contenido de noticias se analizará con técnicas de procesamiento para lenguaje natural lo que posibilitará encontrar posibles descriptores para las noticias.	Alta

Tabla No.2 Requisitos funcionales. Elaboración propia.

La siguiente tabla describe los requisitos no funcionales y su nivel de clasificación.

N^0	**Nombre del requisito no funcional**	**Clasificación**
RNF1	El sistema deberá ser utilizado por los usuarios que tengan acceso a internet	Usabilidad
RNF2	Los servidores para el indexado deben contar con un disco duro de más de 500 GB de almacenamiento	Hardware
RNF3	Los servidores para el almacenamiento de noticias deben contar con un disco duro de más de 1 TB de capacidad.	Hardware
RNF4	Los servidores donde se instalarán cada uno de los componentes del sistema deben tener una distribución GNU/Linux Ubuntu.	Software
RNF5	Los servidores deberán tener soporte para Python superior a 2.7	Software
RNF6	Los servidores deberán tener soporte para java Jdk-8	
RNF7	Se requiere la instalación de la herramienta Scrapy para realizar el rastreo de los sitios web	Software
RNF8	Se requiere la instalación de un servidor Solr para la indexación de sitios web para extraer metadatos.	Software
RNF9	Se requiere la instalación de un servidor de base de datos para el almacenamiento de las noticias.	Software

RNF10	La información manejada en el sistema debe estar protegida mediante llaves certificadas.	Seguridad
RNF11	El sistema no debe requerir información confidencial de los usuarios para su uso.	Seguridad

Tabla No.3 Requisitos no funcionales. Elaboración propia.

2.4. Arquitectura del sistema.

En esencia, un componente es una pieza de código preelaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los "ingredientes de las aplicaciones", que se juntan y combinan para llevar a cabo una tarea. Es algo muy similar a lo que podemos observar en el equipo de música que tenemos en casa. Cada componente de aquel aparato ha sido diseñado para acoplarse perfectamente con sus pares, las conexiones son estándar y el protocolo de comunicación está ya preestablecido. Al unirse las partes, obtenemos música. (20)

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes. El uso de este paradigma posee algunas ventajas:

1. **Reutilización del software**: Nos lleva a alcanzar un mayor nivel de reutilización de software.
2. **Simplifica las pruebas**: Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
3. **Simplifica el mantenimiento del sistema**: Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
4. **Mayor calidad**: Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

De la misma manera, el optar por usar componentes de terceros en lugar de desarrollarlos, posee algunas ventajas:

1. **Ciclos de desarrollo más cortos**: La adición de una pieza dada de funcionalidad tomará días en lugar de meses o años.
2. **Mejor ROI**: Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.

3. **Funcionalidad mejorada**: Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible.

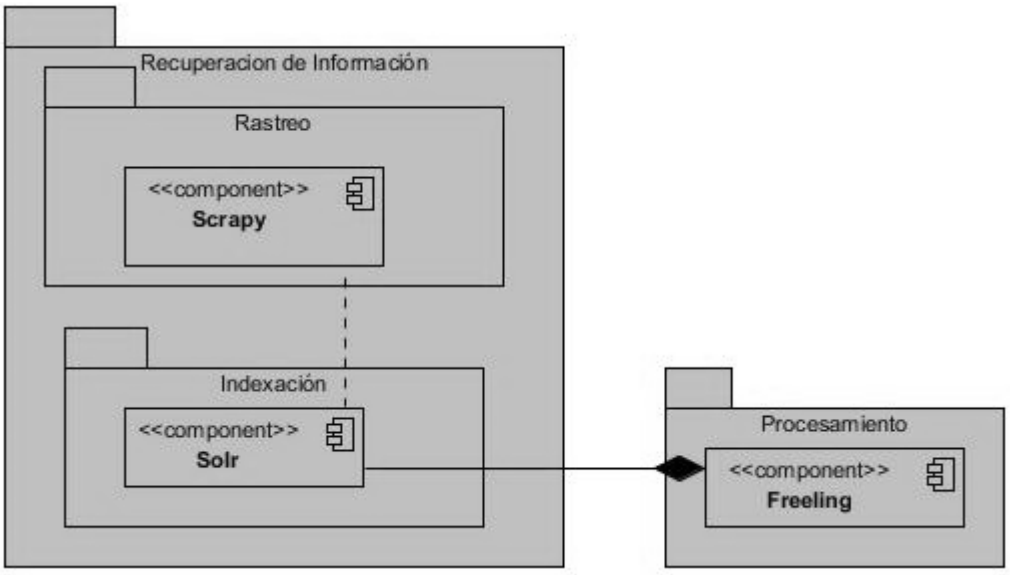


Fig3. Diagrama de arquitectura del sistema. Elaboración propia.

El Diagrama muestra la relación y la dependencia existente en las herramientas utilizadas para la configuración.

El *crawling* es el proceso automatizado de visitar páginas web con el objetivo de recuperar contenido. El contenido que se extrae podría tener muchas formas: texto, imágenes o vídeos. Las herramientas que realizan este trabajo se conocen comúnmente como arañas web, robots, o indexadores automáticos. Scrapy tiene soporte integrado para seleccionar y extraer datos de fuentes html/xml utilizando selectores CSS ampliados y expresiones XPath, usando métodos de ayuda para extraer expresiones regulares. Scrapy cuenta con una consola interactiva para probar las expresiones CSS y XPath para el rastreo, lo que es muy útil al depurar los *spiders*. También cuenta con soporte integrado para generar exportaciones en múltiples formatos (JSON, CSV, XML) y almacenarlos en múltiples *backends* (FTP, S3, sistema local de archivos), cuenta con un soporte de codificación robusto y auto-detección para tratar declaraciones de codificaciones rotas y no estándar.

Solr es el programa de código libre perteneciente al proyecto Apache que se ocupa de las tareas de búsqueda y recuperación de información. Solr emplea la biblioteca Lucene para las tareas esenciales de

indexación y recuperación, pero a través de Tomcat o Jetty permite una configuración local integral que posibilita el desarrollo de cualquier aplicación de búsqueda y recuperación. Sin embargo, es necesario instalar por separado Scrapy y Solr, configurarlos correctamente, y posteriormente integrarlos de manera que las páginas obtenidas con Scrapy en el fichero json puedan ser indexadas y recuperadas mediante Solr.

El paquete FreeLing consiste en una biblioteca que proporciona servicios de análisis de lenguaje como análisis morfológico, reconocimiento de fecha, etiquetado PoS, etc.

2.5. Modelo de Datos.

Típicamente un modelo de datos permite describir las estructuras de la base de datos: El tipo de los datos que hay en la base y la forma en que se relacionan. Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada. Para guardar los campos que se describen a continuación es necesario configurarlos en la búsqueda del motor de indexación de Solr.

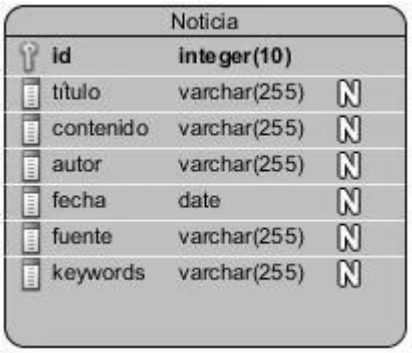


Fig4. Diagrama de modelo de datos. Elaboración propia.

2.6. Modelo de despliegue.

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. Un Nodo es un elemento de hardware o software. En el modelo de despliegue que aparece a continuación se propone tener por separado los servidores de Scrapy y Solr pues ambos una vez comienzan el servicio son grandes consumidores de recursos, aunque pueden funcionar en un mismo servidor si cuenta con buenas prestaciones.

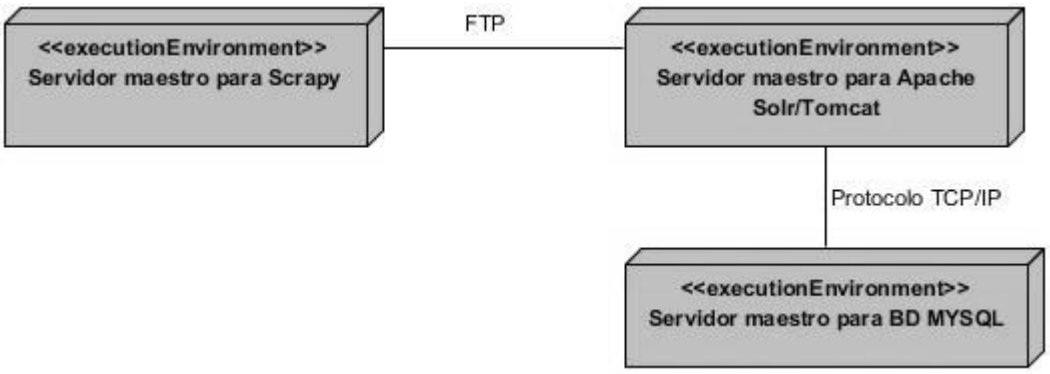


Fig5. Diagrama de despliegue. Elaboración propia.

2.7. Historias de usuario

Las historias de usuario representan fragmentos de funcionalidad o comportamiento que se desea que posea un producto. El término fue introducido por Kent Beck como parte de la metodología *Extreme Programming* para desarrollar una forma de definición de requisitos informal que pudiera favorecer el entendimiento entre todo tipo de perfiles, ya fueran técnicos o no.

Los puntos de historia aportan mucha más información a las estimaciones pues representan el tiempo que se puede tardar en completarlas, la complejidad de las mismas, la incertidumbre que encierran, etc. lo que permite medir la productividad del equipo y su evolución. Son una forma de definir el requerimiento de manera simple y concisa, teniendo en mente lo que el sistema hará cuando se utilice, es la unidad de incremento, de estimación por los desarrolladores y de control del proyecto.

A continuación, se muestran las historias de usuarios descritas para cada requisito funcional.

Número: 1	**Nombre de requisito**: Programar CRON
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Media	**Tiempo**: 36 horas
Riesgo: Alto	**Tiempo real**: 30 horas
Descripción: Establecer un fichero batch en el sistema para que se ejecute con el cron, el cual contiene todos los comandos de ejecución de los diferentes servicios. Permite establecer cada que tiempo se ejecutará el script	

Observaciones:

Tabla No.4 Historia de usuario #1. Elaboración propia.

Número: 2	**Nombre de requisito**: Rastrear sitios web a partir de un spider
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Media	**Tiempo**: 60 horas
Riesgo: Alto	**Tiempo** real: 54 horas
Descripción: Configurar en la herramienta Scrapy el archivo creado como spider el cual se encargará de parsear la página y extraer los datos necesarios (tittle, content, url, etc), y a su vez los diferentes sitios a rastrear.	
Observaciones:	

Tabla No.5 Historia de usuario #2. Elaboración propia.

Número: 3	**Nombre de requisito**: Indexar sitios web para la extracción de metadatos
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Media	**Tiempo**: 60 horas
Riesgo: Alto	**Tiempo real**: 50 horas
Descripción: Crear core para solr con las especificaciones del contenido a rastrear de Scrapy. Una vez rastreado los sitios web permite consultar los documentos indexados en el servidor obteniendo los valores de los campos establecidos en el fichero schema.xml del core de Solr.	
Observaciones:	

Tabla No.6 Historia de usuario #3. Elaboración propia.

Número: 4	**Nombre de requisito**: Almacenar en base de datos los metadatos
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Alta	**Tiempo**: 90 horas
Riesgo: Alto	**Tiempo real**: 90 horas
Descripción: Una vez indexado el contenido en el solr, a través de un api de java podremos ser capaces de conectarnos a un servidor mysql e insertar el contenido que nos haga falta en base de dato.	
Observaciones:	

Tabla No.7 Historia de usuario #4. Elaboración propia.

Número: 5	**Nombre de requisito**: Comparar contenido en bases de datos.
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Alta	**Tiempo**: 48 horas
Riesgo: Alto	**Tiempo real**: 46 horas
Descripción: Para insertar en base de dato se tendrá en cuenta si hay repeticiones de noticias las cuales se comparan teniendo en cuenta la fuente y el contenido.	
Observaciones:	

Tabla No.8 Historia de usuario #5. Elaboración propia.

Número: 6	**Nombre de requisito**: Analizar contenido en lenguaje natural
Programador: Yunior Chacón Sorio	**Iteración** Asignada: 1
Prioridad: Alta	**Tiempo**: 36 horas
Riesgo: Alto	**Tiempo real**: 30 horas
Descripción: Una vez almacenado en base de dato el campo contenido será procesado en lenguaje natural utilizando técnicas de entidades nombradas para sacar la principal idea del texto, permitiendo ser usadas como posibles descriptores	
Observaciones:	

Tabla No.9 Historia de usuario #6. Elaboración propia.

2.8. Conclusiones del capítulo.

En este capítulo se han abordado los elementos de análisis y diseño del módulo de recuperación y almacenamiento de noticias del sistema SACAN, arribando a las siguientes conclusiones:

1. La elaboración de un modelo de dominio con sus respectivas descripciones permitió una mayor comprensión del sistema que se quiere implementar.
2. La identificación de los requisitos funcionales, permitió un mayor entendimiento de los principales procesos del sistema propuesto.
3. La identificación de los requisitos no funcionales permitió profundizar con mejor detenimiento las características y condiciones necesarias para el sistema que se propone desarrollar.
4. Se definió la arquitectura del sistema que permitió garantizar una mayor flexibilidad en la ejecución de la solución logrando dependencia entre las capas de las mismas.

Capítulo 3 Implementación y validación para el Módulo de Recuperación y Almacenamiento del Sistema SACAN

En este capítulo se describen los artefactos pertenecientes a esta etapa de desarrollo como son el del diagrama de componentes y los estándares de codificación y los casos de prueba.

3.1. Diagrama de componente

En el capítulo anterior se mencionaba que la arquitectura del sistema es basada en componentes Solr/Scrapy/Freeling dicha arquitectura se basa en la unión de diferentes herramientas, apoyada por Apache para realizar la indexación, en el rastreo se cuenta con la ayuda herramientas y plugins desarrollados en Python. En esta sección se abordarán los principales componentes de módulos en el sistema, además de los componentes que interactúan en la capa de datos que estará mantenido por un gestor de base de datos accesible mediante consultas sql.

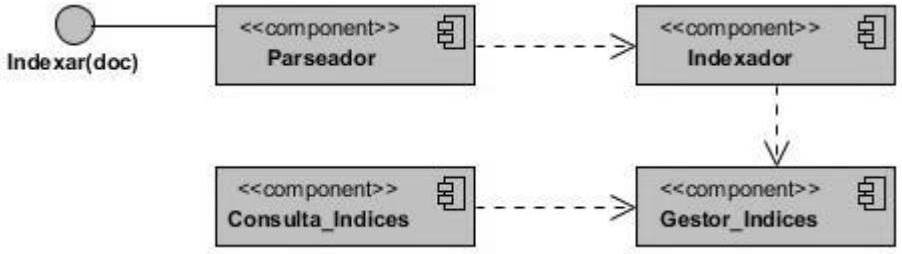


Fig6. Diagrama de componente de Lucene. Elaboración propia.

Lucene es una librería de código abierto para la recuperación de información implementada en java en sus inicios. La misma indexa y busca datos almacenados en ficheros ya sea documentos almacenados localmente hasta páginas web remotas y archivos de texto simple, la figura anterior se muestran los principales componentes que lo componen para parsear e indexar un documento a una base de dato a través de un índice propio.

Solr completa como una extensión las características de Lucene mediante un *plugin* extensible para añadir formatos que no son soportados por la librería Lucene. Entre sus principales características están la posibilidad de las réplicas de servidores, distribución y paralelización de sistemas. Solr engloba todo el

componente de Lucene en su totalidad añadiendo la escalabilidad y consultas avanzadas, el siguiente diagrama se muestran los componentes básicos donde están presente los gestores de expresiones y los controladores para replicas.

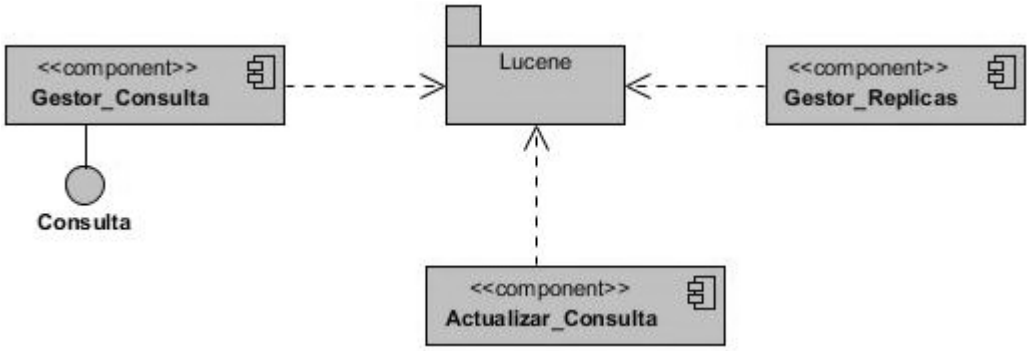


Fig7. Diagrama de componente de Solr. Elaboración propia.

Scrapy será el rastreador encargado de recuperar las páginas web que serán indexadas por Solr el cual incluye diferentes componentes como es el *spider*, *engine*, *downloader*, *schueduler* y el *ítem pipelines*.

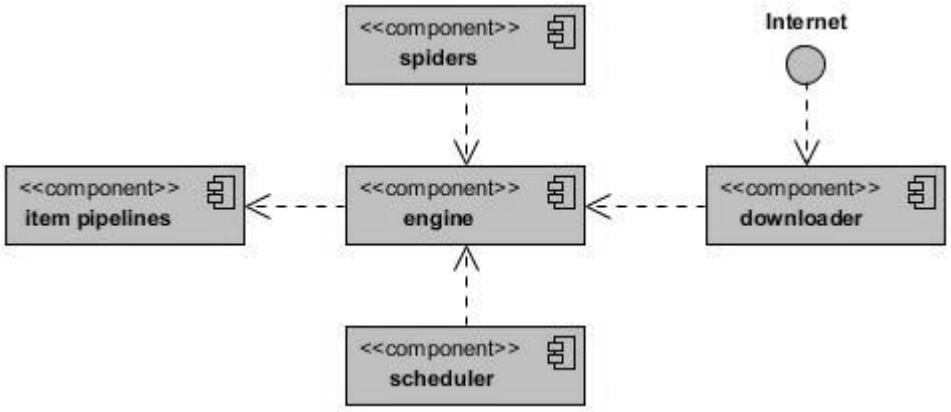


Fig8. Diagrama de componente de Scrapy. Elaboración propia.

*Scrapy Engine***:** es el responsable del flujo de datos entre todos los componentes del sistema y desencadenar eventos cuando ocurren ciertas acciones.

Scheduler: recibe peticiones del *engine* y las pone en cola, luego se las reenvía nuevamente al *engine* cuando solicite una petición.

Downloader: es el responsable de traer las páginas web y mandárselas al engine, y este a su vez al *spider.*

Spiders: son clases escritas por usuarios de Scrapy para parsear respuestas y extraer los elementos de ellos o peticiones adicionales a seguir.

Item Pipeline: es el responsable de procesar cada elemento una vez que han sido extraídos por el *spider*, incluyen típicas tareas como limpieza, validación y persistencia (insertar elementos en base de dato).

Downloader MiddleWare: son *hooks* específicos entre el *engine* y el *downloader que* procesan peticiones cuando pasan del *engine* al *downloader* y las respuestas que pasan del *downloader* al *engine*.

Freeling es el módulo encargado de proporcionar las funcionalidades de análisis al texto. Su funcionamiento básico consiste en obtener un análisis a partir de un texto determinado, el cual trabaja de manera independiente sin ningún tipo de dependencia de otro modulo del sistema, dicho análisis lo realiza desde una perspectiva morfosintáctica, además Freeling posee diccionarios propios y reglas gramaticales. Las interfaces de Freeling se desarrollan empleando tecnología Java mediante la librería JNI (Java Native Invocation) la cual permite invocar el código nativo de Freeling que es C++. Esta interfaz provee funcionalidades básicas de análisis de texto como son lematización, etiquetado morfológico, conteo de palabras, detección de idiomas, etc.

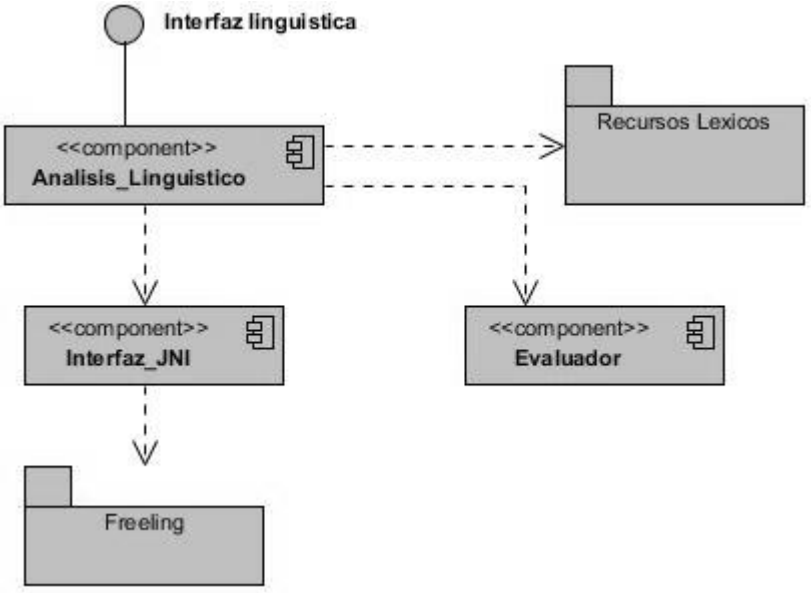


Fig9. Diagrama de componente de Freeling. Elaboración propia.

3.2. Estándares de codificación.

Se definen estándar de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible. Hay muchas razones útiles sobre el porqué usar estándares de codificación como son: facilitan el mantenimiento de una aplicación, dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación. Permite que cualquier programador entienda y pueda mantener la aplicación, en muy raras ocasiones una aplicación es mantenida por su autor original. Los estándares de programación mejoran la legibilidad del código al mismo tiempo que su compresión rápida. Para realizar dicho estándar se utilizó la herramienta IDE Netbeans y como lenguaje de programación Java. (21)

Deben evitarse los ficheros de gran tamaño que contengan más de 1000 líneas. En ocasiones, este tamaño excesivo provoca que la clase no encapsule un comportamiento claramente definido, albergando una gran cantidad de métodos que realizan tareas funcionales o conceptualmente heterogéneas.

Cada fichero fuente Java debe contener una única clase o interfaz pública. El nombre del fichero tiene que coincidir con el nombre de la clase. Cuando existan varias clases privadas asociadas funcionalmente a una clase pública, podrán colocarse en el mismo fichero fuente que la clase pública. La clase pública debe estar situada en primer lugar dentro del fichero fuente.

```
public class MysqlConnect {

    private static final String DATABASE_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/sacan";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "yuniorl2";
    private static final String MAX_POOL = "250";
```

Fig10. Código de la clase MysqlClass. Elaboración propia.

Como norma general se establecen 4 caracteres como unidad de sangría. Los entornos de desarrollo integrado (IDE) más populares, tales como NetBeans, incluyen facilidades para formatear código Java.

No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.

El carácter inicio de bloque ("{") debe aparecer al final de la línea que contiene la sentencia de declaración.

El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de "{".

Los métodos se separarán entre sí mediante una línea en blanco.

Las líneas y espacios en blanco mejoran la legibilidad del código permitiendo identificar las secciones de código relacionadas lógicamente.

```
public static String quote(java.sql.Connection link, String str)
        throws Exception
{
    if (str == null) {
        return "NULL";
    }
    return "'"+mysql_real_escape_string(link,str)+"'";
}

public static String nameQuote(java.sql.Connection link, String str)
        throws Exception
{
    if (str == null) {
        return "NULL";
    }
    return "`"+mysql_real_escape_string(link,str)+"`";
}
```

Fig11. Código de la clase MysqlClass. Elaboración propia.

Se utilizarán espacios en blanco en los siguientes casos:

Entre una palabra clave y un paréntesis. Esto permite que se distingan las llamadas a métodos de las palabras clave. Por ejemplo:

```
while (resultSet.next()) {
    String user = resultSet.getString("iasd");
    String website = resultSet.getString("url");
    System.out.println("asad: " + user);
    System.out.println("asd: " + website);
}
```

Fig12. Código de la clase MysqlClass. Elaboración propia.

Tras cada coma en un listado de argumentos. Por ejemplo: objeto.unMetodo(a, b, c);

Para separar un operador binario de sus operandos, excepto en el caso del operador ("."). Nunca se utilizarán espacios entre los operadores unarios (p.e., "++" o "--") y sus operandos. Por ejemplo:

a += b + c;

a = (a + b) / (c + d);

contador++;

Para separar las expresiones incluidas en la sentencia "for". Por ejemplo:

for (expresion1; expresion2; expresion3)

Al realizar el moldeo o "casting" de clases. Ejemplo:

```
String host = (String) resultDoc.getFieldValue("host");
String titulo = (String) resultDoc.getFieldValue("name");
String link = (String) resultDoc.getFieldValue("url");
String content = (String) resultDoc.getFieldValue("content");
```

Fig13. Código de la clase cliente_Solar. Elaboración propia.

Se deben evitar las asignaciones de un mismo valor sobre múltiples variables en una misma sentencia, ya que dichas sentencias suelen ser difíciles de leer.

Es una buena práctica el uso de paréntesis en expresiones que incluyan distintos tipos de operadores para evitar problemas de precedencia de operadores. (22)

Los atributos de instancia y de clase serán siempre privados, excepto cuando tengan que ser visibles en subclases herederas, en tales casos serán declarados como protegidos.

```
private Properties getProperties() {
    if (properties == null) {
        properties = new Properties();
        properties.setProperty("user", USERNAME);
        properties.setProperty("password", PASSWORD);
        properties.setProperty("MaxPooledStatements", MAX_POOL);
    }
    return properties;
}
```

Fig14. Código de la clase MysqlClass. Elaboración propia.

El acceso a los atributos de una clase se realizará por medio de los métodos "get" y "set" correspondientes, incluso cuando el acceso a dichos atributos se realice en los métodos miembros de la clase.

3.3. Pruebas y validación de la propuesta de solución.

Pruebas unitarias

Se llaman pruebas unitarias porque descomponen las funciones del programa en comportamientos comprobables discretos que se pueden probar como unidades individuales. Las pruebas unitarias tienen el mayor efecto en la calidad del código cuando son parte integral del flujo de trabajo de desarrollo de software. (23)

JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java, permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación. En la actualidad las herramientas de desarrollo como Netbeans cuentan con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas. (24)

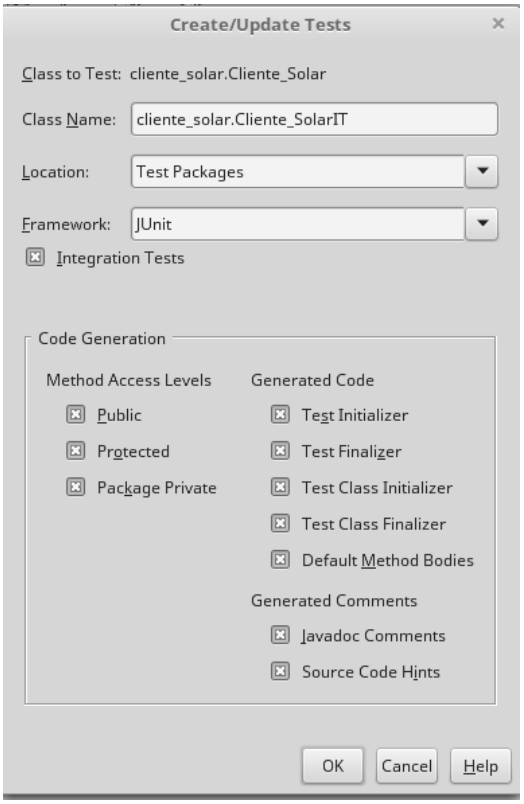
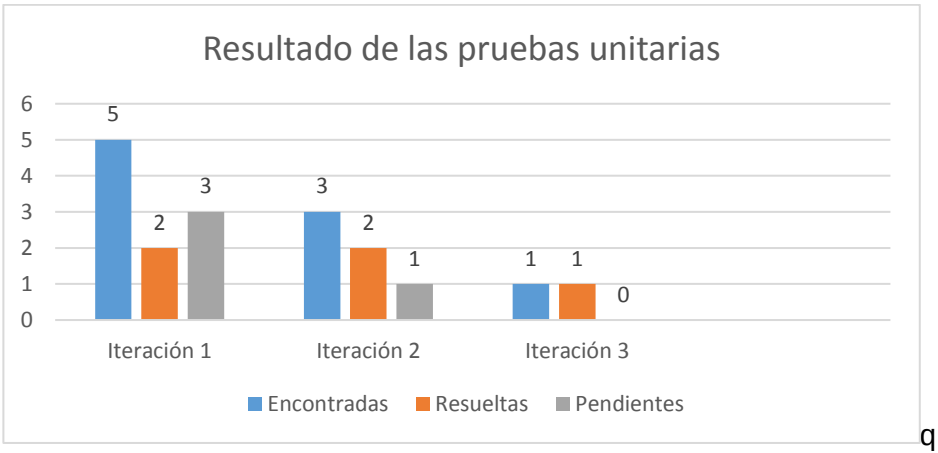


Fig15.Crear prueba. Elaboración propia.

Para llevar a cabo la detección de las no conformidades presentes en la aplicación desarrollada, se realizaron 3 iteraciones de pruebas a través de la herramienta Junit. Las no conformidades encontradas radican mayormente en las consultas de inserción en la base de datos debido a que mysql utiliza un tipo de colección atendiendo a los diferentes caracteres de código ASCII, ciertos caracteres que utilizan código de 3 bytes y otros de 4 bytes, estas fueron reincidentes en las iteraciones hasta que fueron erradicadas. En la siguiente tabla se muestran los resultados obtenidos en cada iteración de prueba.



q

Fig16. Resultado de la prueba de la clase cliente_solar.Cliente_SolarIT. Elaboración propia.

Pruebas de integración

El *testing* o fase de prueba es una de las fases más cruciales en el desarrollo del software que se encuentran dividido por una serie de pruebas las unitarias antes desarrolladas que deben de ser aplicadas antes de las pruebas de integración con la intención de probar los métodos aplicados en el avance de la aplicación. En esta fase se comprueban también las comunicaciones de forma invariable tanto si están representadas con software o con hardware. El motivo principal se encuentra en que el test integral lleva a cabo la revisión conjunta de los diferentes elementos que están presentes con el objetivo de formar el software. Se realiza la comprobación para ver que todo funciona de una manera adecuada en conjunto, dado que no es extraño que se produzcan alteraciones en el rendimiento.

Componente	Funcionalidad	Funcionalidad Integridad	Resultado de la prueba

Servidor Apache Solr	Obtener documentos indexados por Solr	El módulo de recuperación y almacenamiento de información se integra permitiendo la extracción del contenido de Solr para ser insertado en el servidor de base de dato MySQL.	El resultado de la prueba fue satisfactorio insertándose en tabla los documentos indexados por Solr.
Módulo de análisis y visualización de información.	Clasificar documentos insertados en base de datos por temática y descriptores.	El módulo desarrollado se integra permitiendo brindar datos asociados por noticia que permitan ayudar a su clasificación como son las entidades nombradas.	Se pudo integrar con un sistema desarrollado en drupal capaz de obtener datos antes insertados, permitiendo su clasificación.
Freeling	Es el encargado del procesamiento de la información.	El módulo desarrollado se integra permitiendo brindar datos asociados por noticia que permitan ayudar a su clasificación como son las entidades nombradas.	Las entidades obtenidas fueron guardadas en base de dato para ser utilizadas en un posterior análisis.

Tabla No. 10 Pruebas de integración. Elaboración propia.

3.4. Conclusiones del capítulo.

Durante el desarrollo del capítulo se realizaron pruebas que permitieron valorar el desarrollo de la aplicación llegando a las siguientes conclusiones.

- Los estándares de codificación permitieron legibilidad en el código para una interpretación más rápida.
- Las pruebas realizadas permitieron corregir posibles fallos en las funcionalidades del sistema, lo que demuestra el funcionamiento actual en el que se encuentra la aplicación.
- La implementación fue desarrollada en una arquitectura basada en componentes tal como se pudo observar en la etapa de elaboración de desarrollo, así como los entornos de despliegue.

Conclusiones

Con el empleo de los métodos de investigación teóricos se logró conocer el estado del objeto de estudio de la investigación. Analizando los principales conceptos asociados al tema de investigación y las características presentadas por las herramientas de rastreo consideradas más importantes; así como la identificación de los procesos y funcionalidades presentas en el procesamiento de los datos se comprendió de manera satisfactoria toda la situación problemática planteada.

En el trabajo realizado se demuestran las potencialidades de las herramientas de software libre Apache para la implementación de un motor de búsqueda aplicando una metodología para la construcción de buscadores especializados.

Se observó como principal ventaja que tiene los índices de Solr sobre los índices de las bases de datos relacionales, la cual radica en el modelo de datos. La característica que define a las bases de datos relacionales es su modelo de datos basado en tablas y la posibilidad de relacionarlas entre sí mediante llaves.

El desarrollo de un modelo de dominio permitió encaminar la aplicación reflejándose en los requisitos funcionales y no funcionales para una mayor compresión del sistema en cuanto a condiciones y características necesarias para su desarrollo.

Las pruebas realizadas permitieron corregir posibles fallos en las funcionalidades del sistema, apoyadas por los estándares de codificación para una rápida implementación de la aplicación.

Bibliografía

1. REESE, George. *Database Programming with JDBC and Java* [En Línea]. 2000. ISBN 1565926161. Disponible en: http://portal.acm.org/citation.cfm?id=557193

2. MICROSYSTEM, Sun and POINT, Tutorials. Java. *Time* [En Línea]. 2015. P. 17–26. Disponible en: www.tutorialspoint.com

3. OMG and OMG. Unified Modeling Language (OMG UML), Infrastructure. *Uml*. 1998. Vol. 21, no. 2, p. 89–90. DOI 10.1007/s002870050092.

4. CARRERAS, Xavier, CHAO, Isaac, PADRÓ, Lluís and PADRÓ, Muntsa. Freeling: An Open-Source Suite of Language Analyzers. *Proceedings of the 4th Language Resources and Evaluation Conference (LREC 2004)* [En Línea]. 2004. Vol. 4, p. 239–242. Disponible en: http://hnk.ffzg.hr/bibl/lrec2004/pdf/271.pdf

5. EN FREELING, Gramáticas de Dependencia, CARRERA, Jordi, CASTELLÓN, Irene, LLOBERES, Marina, PADRÓ, Lluís and TINKOVA, Nevena. Dependency Grammars in FreeLing. *Procesamiento del Lenguaje Natural* [En Línea]. 2008. Vol. 41, no. 41, p. 21–28. Disponible en: http://www.sepln.org/revistaSEPLN/revista/41/sec1-art3.pdf%5Cnhttp://nlp.lsi.upc.edu/papers/carrera08.pdf

6. PADRÓ, L. Analizadores multilingües en freeling. *Linguamática* [En Línea]. 2012. Vol. 3, p. 13–20. Disponible en: http://upcommons.upc.edu/handle/2117/14772

7. ORACLE CORPORATION. *MySQL: What is MySQL?* [En Línea]. 2012. Disponible en: https://www.oracle.com/es/mysql/index.html

8. STATE LIBRARY OF WESTERN AUSTRALIA. Pandora Archive - Preserving and Accessing Networked DOcumentary Resources of Australia. *Pandora* [En Línea]. 2011. Disponible en: http://pandora.nla.gov.au/tep/129794

9. LLUECA, Ciro, CÓCERA-SALÓ, Daniel, TORRES, Natalia, SUADES MÉNDEZ, Gerard, DE-LA-VEGA-SIVERA, Ricard, CÓCERA SALÓ, Daniel, TORRES, Natalia, SUADES MÉNDEZ, Gerard and DE LA VEGA SIVERA, Ricard. PADICAT, el archivo de Internet. In : *Actas de las XII Jornadas*

Españolas de Documentación: Málaga, 2011 [En Línea]. 2011. p. 142–146. Disponible en: http://eprints.rclis.org/15761/1/padicat_fesabid2011.pdf

10. MANNING, Christopher, SURDEANU, Mihai, BAUER, John, FINKEL, Jenny, BETHARD, Steven and MCCLOSKY, David. The Stanford CoreNLP Natural Language Processing Toolkit. In : *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* [En Línea]. 2014. p. 55–60. ISBN 9781941643006. Disponible en: http://aclweb.org/anthology/P14-5010

11. COLLOBERT, Ronan and WESTON, Jason. A unified architecture for natural language processing. In: *Proceedings of the 25th international conference on Machine learning - ICML '08* [En Línea]. 2008. p. 160–167. ISBN 9781605582054. Disponible en: http://portal.acm.org/citation.cfm?doid=1390156.1390177

12. COLLOBERT, Ronan, WESTON, Jason, BOTTOU, Léon, KARLEN, Michael, KAVUKCUOGLU, Koray and KUKSA, Pavel. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*. 2011. Vol. 12, p. 2493–2537. DOI 10.1.1.231.4614.

13. BIRD, Steven, KLEIN, Ewan and LOPER, Edward. *Natural Language Processing with Python* [En Línea]. 2009. ISBN 9780596516499.

14. WEIKUM, Gerhard. Foundations of statistical natural language processing. *ACM SIGMOD Record* [En Línea]. 2002. Vol. 31, no. 3, p. 37. DOI 10.1145/601858.601867. Disponible en: http://portal.acm.org/citation.cfm?doid=601858.601867

15. BAI, Ying. Introduction to NetBeans IDE. In : *Practical Database Programming with Java* [En Línea]. 2011. p. 155–315. ISBN 9781118104651. Disponible en: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5989218

16. NETBEANS. NetBeans IDE. *Features* [En Línea]. 2015. No. Jsr 299, p. 1–3. Disponible en: https://netbeans.org/features/java/

17. ARCAND, Jean-Francois, BARKER, Bill, CLERE, Jean-Frederic, DARWIN, Ian, HANIK, Filip, JUNG, Rainer, LAURENT, Sylvain and SHAPIRA, Yoav. *Apache Tomcat* [En Línea]. 2007. Disponible en: http://tomcat.apache.org/

18. THE APACHE SOFTWARE FOUNDATION. Apache HBase. *The Apache Software Foundation* [En Línea]. 2007. Disponible en: https://hbase.apache.org

19. PEÑAS, Anselmo. Apache Lucene. *Language Resources and Evaluation* [En Línea]. 2009. P. 1–22. Disponible en: http://lucene.apache.org/

20. JENA, Apache. *Apache jena* [En Línea]. 2013. Disponible en: http://jena.apache.org/

21. ESTRADA RAMOS, Luis Miguel. Apache Solr, un motor de búsqueda de código abierto. *Revista Digital Universitaria*. 2012. Vol. 13, no. 11, p. 1–9.

22. MANGHI, Paolo, ARTINI, Michele, BARDI, Alessia, ATZORI, Claudio, LA BRUZZO, Sandro and MIKULICIC, Marko. High-Performance Annotation Tagging over Solr Full-text Indexes. *Information Technology and Libraries* [En Línea]. 2014. Vol. 33, no. 3, p. 22–44. DOI 10.6017/ital.v33i3.4633. Disponible en: http://ejournals.bc.edu/ojs/index.php/ital/article/view/4633%5Cnhttp://ejournals.bc.edu/ojs/index.php/ital/article/download/4633/pdf

23. INGERSOLL, Grant. Better Search with Apache Lucene and Solr. In: *Triangle Java Users Group Conference November 19, 2007* [En Línea]. 2007. Disponible en: http://trijug.org/downloads/TriJug-11-07.pdf

24. APACHE. *Apache solr* [En Línea]. 2013. ISBN 9781784399641. Disponible en: http://repository.unikom.ac.id/man/php/book.solr.html

25. MOREIRA, José E., MICHAEL, Maged M., DA SILVA, Dilma, SHILOACH, Doron, DUBE, Parijat and ZHANG, Li. Scalability of the Nutch search engine. *Proceedings of the 21st annual international conference on Supercomputing - ICS '07* [En Línea]. 2007. P. 3. DOI 10.1145/1274971.1274975. Disponible en: http://portal.acm.org/citation.cfm?doid=1274971.1274975

26. PROFFIT, Brian. Hadoop: What It Is And How It Works. *Http://Readwrite.Com/* [En Línea]. 2013. Disponible en: http://readwrite.com/2013/05/23/hadoop-what-it-is-and-how-it-works/

27. THE APACHE SOFTWARE FOUNDATION. Apache Nutch. *The Apache Software Foundation* [En Línea]. 2012. Disponible en: http://nutch.apache.org/

28. MANNING, Christopher D., RAGHAVAN, Prabhakar and SCHÜTZE, Hinrich. Text classification and Naive Bayes. *Introduction to Information Retrieval* [En Línea]. 2008. No. c, p. 260. DOI 10.1109/TrustCom.2013.73. Disponible en: https://books.google.co.th/books/about/Introduction_to_Information_Retrieval.html?id=GNvtngEACAAJ&pgis=1

29. QIN, Tao, LIU, Tie Yan and LI, Hang. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*. 2010. Vol. 13, no. 4, p. 375–397. DOI 10.1007/s10791-009-9124-x.

30. KONCHADY, M. *Building Search Applications: Lucene, Lingpipe, and Gate* [En Línea]. 2008. ISBN 9780615204253. Disponible en: http://portal.acm.org/citation.cfm?id=1386652

31. JAHODA, Gerald. *Automatic information organization and retrieval*. 1970. ISBN 0070544859.

32. DONYAI, P. Information retrieval. *Information Retrieval* [En Línea]. 2009. Vol. 4, no. 2, p. 1–13. Disponible en: http://centaur.reading.ac.uk/13761/

33. HIEMSTRA, Djoerd. Information Retrieval Models. In : *Information Retrieval: Searching in the 21st Century*. 2009. p. 1–19. ISBN 9780470027622.

34. HENRICH, Andreas. Information Retrieval 1. *Information Retrieval* [En Línea]. 2008. Vol. 2, p. 421. DOI 10.1007/s10791-008-9059-7. Disponible en: https://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2008/henrich-ir1-1.2.pdf%5Cnhttps://www.uni-bamberg.de/minf/IR1-Buch

35. RIJSBERGEN, C J Van and VAN RIJSBERGEN, C J. Information Retrieval. *Information Retrieval*. 2010. Vol. 30, no. 6, p. 208. DOI 10.1016/0020-0271(68)90016-8.

36. MANNING, Christopher D. Introduction to Information Retrieval - Chapter 2: The termvocabulary and postings lists. *Information Retrieval*. 2009. No. c, p. 1–18. DOI 10.1109/LPT.2009.2020494.

37. MITRA, Mandar and CHAUDHURI, Bidyut Baran. Information Retrieval from Documents: A Survey. *Information Retrieval* [En Línea]. 2000. Vol. 2(2-3), p. 141–163. DOI 10.1023/A:1009950525500. Disponible en:

http://search.proquest.com/docview/230671268/abstract/91620A19686148FCPQ/1?accountid=15920

38. LOGAN, Elisabeth. Information retrieval interaction. *Information Processing & Management* [En Línea]. 1993. Vol. 29, no. 6, p. 794. DOI 10.1016/0306-4573(93)90108-P. Disponible en: http://linkinghub.elsevier.com/retrieve/pii/030645739390108P

39. TURNEY, Peter D. Learning Algorithms for Keyphrase Extraction. *Information Retrieval* [En Línea]. 2000. Vol. 2, p. 303–336. DOI 10.1023/A: 1009976227802. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.1829&rep=rep1&type=pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.1829

40. BELKIN, N J. Interaction with texts: Information retrieval as information seeking behavior. *Information retrieval*. 1993. Vol. 93, p. 55–66. DOI 10.1.1.85.7785.

41. SALTON, Gerard and BUCKLEY, Christopher. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*. 1988. Vol. 24, no. 5, p. 513–523. DOI 10.1016/0306-4573(88)90021-0.

42. RIJSBERGEN, C.J. Van. Information Retrieval. *Information Retrieval*. 1979. P. 208. DOI 10.1007/SpringerReference_16360.

43. ZHANG, Tong and OLES, Frank J. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval*. 2001. Vol. 4, no. 1, p. 5–31. DOI 10.1023/A: 1011441423217.

44. GREENGRASS, E. *Information retrieval: A survey* [En Línea]. 2000. Disponible en: http://www.cs.umbc.edu/cadip/readings/IR.report.120600.book.pdf

45. BAEZA-YATES, Ricardo and RIBEIRO-NETO, Berthier. Modern information retrieval. *New York* [En Línea]. 1999. Vol. 9, p. 513. DOI 10.1080/14735789709366603. Disponible en: http://web.simmons.edu/~benoit/LIS466/Baeza-Yateschap01.pdf%5Cnftp://mail.im.tku.edu.tw/seke/slide/baeza-yates/chap10_user_interfaces_and_visualization-modern_ir.pdf

46. YANG, Yiming. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval* [En Línea]. 1999. Vol. 1, no. 1, p. 69–90. DOI 10.1023/A: 1009982220290. Disponible en: http://www.cs.cmu.edu/~%5Cnhttp://dx.doi.org/10.1023/A:1009982220290

47. BAEZA-YATES, Ricardo and RIBEIRO-NETO, Berthier. Modern Information Retrieval: The Concepts and Technology behind Search. *Information Retrieval* [En Línea]. 2011. Vol. 82, p. 944. DOI 10.1080/14735789709366603

48. MANNING, Christopher D. and RAGHAVAN, Prabhakar. An Introduction to Information Retrieval. In : *En Línea* [En Línea]. 2009. p. 1. ISBN 0521865719. Disponible en: http://dspace.cusat.ac.in/dspace/handle/123456789/2538

49. SINGHAL, Amit. Modern Information Retrieval: A Brief Overview. *Readings*. 2001. P. 1–9.

50. JONES, Karen, WILLETT, Peter, SPARCK JONES, Karen, WILLETT, Peter, KAREN SPARCK, Jones and WILLETT, Peter. Readings in information retrieval. In: *Morgan Kaufmann series in multimedia information and systems* [En Línea]. 1997. p. 589. ISBN 1558604545. Disponible en: citeulike-article-id: 308698%

51. ANTONIOU, Grigoris and HARMELEN, Frank van. *Semantic Web Primer* [En Línea]. 2004. ISBN 0262012103. Disponible en: http://www.linkeddatatools.com/semantic-web-basics Page

Referencias bibliográficas.

1. TRENDS, Google. Digital Usage Trends: Half Year – IAB. [En Línea]. 2016. Disponible en: https://www.iab.com/wp-content/uploads/2016/11/Mid-Year-2016-Digital-Usage-Trends-FINAL-DRAFT-with-summary.pdf

2. STATISTA, Statistic. Fastest Growing Applications for Categories. [En Línea]. Disponible en: https://www.statista.com/statistics/251096/fastest-growing-shopping-app-categories/

3. Procesamiento de lenguaje natural - IIC. [En Línea]. Disponible en: http://www.iic.uam.es/soluciones/inteligencia-de-cliente/procesamiento-lenguaje-natural

4. La importancia de tener un NER - IIC. [En Línea]. Disponible en: http://www.iic.uam.es/inteligencia/la-importancia-tener-ner

5. BLÁZQUEZ ORLANDO. Técnicas avanzadas de recuperación de información: Procesos, técnicas y métodos. [En Línea]. 2013. Disponible en: http://eprints.ucm.es/31307

6. Conceptos básicos de la recuperación de información. [En Línea]. 2012. Disponible en: http://ccdoc-tecnicasrecuperacioninformacion.blogspot.com/2012/02/conceptos-basicos-en-recuperacion-de.html

7. BLAZQUEZ, M. Aplicaciones documentales de la recuperación de información. [En Línea]. 2013. Disponible en: http://mblazquez.es/wp-content/uploads/ebook-mbo-tecnicas-avanzadas-recuperacion-informacion1.pdf

8. ORLANDO, Blazquez. El proceso de indexación. [En Línea]. 2012. Disponible en: http://ccdoc-tecnicasrecuperacioninformacion.blogspot.com/2012/11/el-proceso-de-indexacion.html

9. Introducción a la recuperación de información. [En Línea]. Disponible en: http://www.grupolys.org/docencia/ln/biblioteca/ir.pdf

10. STATE LIBRARY OF WESTERN AUSTRALIA. Pandora Archive - Preserving and Accessing Networked DOcumentary Resources of Australia. *Pandora* [En Línea]. 2011. Disponible en: http://pandora.nla.gov.au/tep/129794

11. LLUECA, Ciro, CÓCERA-SALÓ, Daniel, TORRES, Natalia, SUADES MÉNDEZ, Gerard, DE-LA-VEGA-SIVERA, Ricard, CÓCERA SALÓ, Daniel, TORRES, Natalia, SUADES MÉNDEZ, Gerard and DE LA VEGA SIVERA, Ricard. PADICAT, el archivo de Internet. In: *Actas de las XII Jornadas Españolas de Documentación: Málaga, 2011* [En Línea]. 2011. p. 142–146. Disponible en: http://eprints.rclis.org/15761/1/padicat_fesabid2011.pdf

12. Scrapy Crawler Documentation. [En Línea]. 2018. Disponible en: https://doc.scrapy.org

13. APACHE. *Apache solr* [En Línea]. 2013. ISBN 9781784399641. Disponible en: http://repository.unikom.ac.id/man/php/book.solr.html

14. ARCAND, Jean-Francois, BARKER, Bill, CLERE, Jean-Frederic, DARWIN, Ian, HANIK, Filip, JUNG, Rainer, LAURENT, Sylvain and SHAPIRA, Yoav. *Apache Tomcat* [En Línea]. 2007. Disponible en: http://tomcat.apache.org/

15. ORACLE CORPORATION. *MySQL :: What is MySQL?* [En Línea]. 2012. Disponible en: https://www.oracle.com/es/mysql/index.html

16. CARRERAS, Xavier, CHAO, Isaac, PADRÓ, Lluís and PADRÓ, Muntsa. Freeling: An Open-Source Suite of Language Analyzers. *Proceedings of the 4th Language Resources and Evaluation Conference (LREC 2004)* [En Línea]. 2004. Vol. 4, p. 239–242. Disponible en: http://hnk.ffzg.hr/bibl/lrec2004/pdf/271.pdf

17. Visual Paradigm - Leading UML, BPMN, EA, Agile and Project. [En Línea]. 2018. Disponible en: https://www.visual-paradigm.com/

18. NETBEANS. NetBeans IDE. *Features* [En Línea]. 2015. No. Jsr 299, p. 1–3. Disponible en: https://netbeans.org/features/java/

19. PROFFIT, Brian. Hadoop: What It Is And How It Works. *Http://Readwrite.Com/* [En Línea]. 2013. Disponible en: http://readwrite.com/2013/05/23/hadoop-what-it-is-and-how-it-works/

20. MICROSOFT. Desarrollo de Software basado en Componentes. [En Línea]. 2018. Disponible en: https://msdn.microsoft.com/es-es/library/bb972268.aspx

21. ConvecionesCodigoJava. [En Línea]. 2012. Disponible en: http://www.um.es/docencia/vjimenez/ficheros/practicas/ConvencionesCodigoJava.pdf

22. Estilo-codificación. [En Línea]. 2011. Disponible en: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf

23. MICROSOFT. Pruebas Unitarias. [En Línea]. 2018. Disponible en: https://msdn.microsoft.com/es-es/library/hh694602.aspx

24. JUnit 4. [En Línea]. 2018. Disponible en: https://junit.org/junit4/