UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESIS

APLICACIÓN DEL PROCESAMIENTO INTELIGENTE DE TEXTO PARA LA INDIZACIÓN DE DOCUMENTOS DIGITALES Y OBTENCIÓN DE TÉRMINOS MULTIPALABRA.

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

CHRISTIAN DANIEL ENCISO PADILLA

DIRECTOR DE TESIS: M.I. MARCIAL CONTRERAS BARRERA

CIUDAD UNIVERSITARIA. SEPTIEMBRE 2015

AGRADECIMIENTOS.

A Dios.

A mis padres y abuelos quienes han puesto todo su esfuerzo, amor y dedicación de su parte para que yo esté aquí en este momento. A ustedes debo esto.

Emmanuel, Marcial: gracias por ser un ejemplo a seguir y una firme enseñanza día con día.

A mi familia por su incondicional apoyo. Hermanos, tíos, primos, amigos… Gracias.

Vero, Sergio, Iván, Carlos: Gracias.

A Lili, porque siempre ha estado en los momentos buenos y más aún en los difíciles.

A cada profesor en mi vida que hizo posible esto.

Liz, Alex, Fer, Fede: Gracias por la paciencia y por apoyarme en momentos importantes de mi desarrollo personal y profesional.

En especial a Don Mario. Porque sé que siempre ha cuidado de mí, a pesar de no estar… Ahora sí abuelo: ¡MISIÓN CUMPLIDA!

Índice de contenido

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

Desarrollada hace diez mil años en el medio oriente, la escritura es una extensión del cerebro humano.

Podemos comunicarnos a través de la distancia y el tiempo. ***(History channel, 2012)***

Antecedentes

INTRODUCCIÓN

En este capítulo se definen los aspectos que se consideraron como importantes para el desarrollo de la presente tesis, tal como el problema a resolver, los objetivos, la hipótesis y los alcances del sistema, no confundir con los antecedentes teóricos que dan fundamento al desarrollo del sistema, los cuales se plantean más adelante en el capítulo 2 “Introducción a ingeniería de software, a bases de datos y a estadística”.

1.1. ANTECEDENTES

En todo el mundo, millones de archivos de texto en formato digital son creados anualmente, gracias al avance tecnológico que permite que cada vez más personas tengan acceso a las tecnologías de la información. Este crecimiento en la generación de documentos digitales ha derivado en diversas problemáticas a las que la comunidad científica-tecnológica debe hacer frente. Como lo es la necesidad inherente a la creación de información de almacenar y compartir dicha información generada.

Basta decir que la importancia de la generación de nueva información no se limita a que ésta sea almacenada sino poder ser accedida posteriormente y recuperar las ideas que en un principio se decidieron guardar, para hacernos una idea del gran problema que resulta el tener todos esos archivos organizados de tal forma que encontremos justo los que necesitamos para cierta investigación o simplemente para saciar nuestra curiosidad. Es decir, ¿cómo saber qué libro pertenece a qué tema? ¿Qué libro podría relacionarse con otro libro para cierto tema y con otro para otro tema? Sería necesario leer cada libro y entonces sí, si conocemos del tema, organizarlo de cierta manera. Pero quedará la duda ¿Estaremos seleccionando el tema adecuado para ese libro? ¿Y si en el libro hay otros temas que no contemplé? Es por ello que la comunidad tecnológica se ha dado a la tarea de realizar investigaciones y sobre todo, desarrollar y poner en práctica algoritmos que permitan la detección de datos relevantes dentro de los libros de una manera eficiente y precisa.

Para algunos, la escritura marca el inicio de la historia. *Cuando el hombre pudo poner por primera vez sus ideas e impresiones por escrito*. (Pigna, 2015)

El hombre fue capaz de procesar pensamientos para compartirlos con otros seres humanos de una manera más permanente que la transmisión oral.

A partir de ese momento, el hombre encontró la utilidad de transferir sus conocimientos a otros seres humanos y sobre todo, a sus descendientes.

Una parte del trabajo había sido completada, la transferencia de ideas a través de un medio escrito, en paredes, piedras, tablas, papiros, rollos, hojas, libros y archivos electrónicos. Pero, sólo es la mitad del trabajo. La otra mitad consiste en la interpretación de esos símbolos.

Por un lado, tenemos el método clásico de la lectura. Un proceso cognitivo del ser humano quien es capaz de procesar información en un estado textual y sintetizarla, abstraerla y, en el mejor de los casos, generar nueva información a partir de la proporcionada. El proceso de la lectura es descrito por Rodríguez Gallardo como "… el proceso mediante el cual una persona es capaz de descifrar lo que otra ha escrito con el propósito de comunicar una idea, preservar un conocimiento o transmitir un mensaje." (Rodríguez Gallardo, Definiendo la lectura, el alfabetismo y otros conceptos relacionados, 2007)

Y por otro lado, con el relativamente reciente desarrollo de sistemas de cómputo, tenemos la idea del procesamiento automatizado de documentos del tipo documental textual para la extracción de información.

Si bien estos procesos de extracción de información son importantes en el momento de la transferencia de conocimiento entre personas y a través del tiempo, es, en estos tiempos, aún más importante el proceso de extracción de la información para el almacenamiento y organización de los archivos (originalmente físicos y recientemente digitales) para su posterior recuperación que se precisa en temas determinados.

Para ello, existe un proceso para almacenar de manera ordenada los textos. Una parte importante es la indización de dichos documentos, en la cual, se obtienen las palabras que mejor definen el contenido del texto para que de esa manera, se puedan realizar búsquedas sobre temas específicos y no solo por títulos o autores.

El estándar ISO (International Organization for Standarization) 5963 "Documentation- Methods for examining documents, determining their subjects, and

selecting indexing terms" define *indexing*, en inglés, como "El acto de describir o identificar un documento en términos de su contenido" (International Standard, 1985) sin embargo, es importante señalar que para este estudio y en el campo de la bibliotecología, el término indizado resulta diferente a indexado, ya que la indización:

"…es un proceso de identificación del contenido de un documento y su descripción a través de términos verbales... Es una técnica de análisis sobre el contenido de un documento que busca expresar la información más significativa a través de la asignación de términos descriptores y crear así un lenguaje de mediación entre el usuario y el documento." (Urbizagástegui & Restrepo, 2011)

Mientras que indexado, de acuerdo con el Diccionario de la Real Academia Española es el proceso de "Registrar ordenadamente datos e informaciones, para elaborar su índice" (Real Academia Española, 2001) de esta manera, un índice facilita y agiliza el acceso a la información ordenada mientras que la indexación permite conocer descriptores del documento.

El proceso de indizado de material bibliográfico en las bibliotecas de la UNAM es uno de los procesos más importantes (además de complejos y tardados) para la posterior recuperación de información y del mismo material bibliográfico mediante motores de búsqueda de la biblioteca.

Una vez que se ha mencionado el proceso de indizado, llega el momento de describir a un analista documental, quien es la persona que se encarga, entre muchas otras actividades, de hacer un proceso mental/manual de extracción de las palabras clave que definen el contenido de un ejemplar que llega a las bibliotecas, en este caso, bibliotecas de la universidad.

El material bibliográfico que adquieren las bibliotecas de la Universidad Nacional Autónoma de México, es procesado por analistas documentales, quienes luego de un proceso mental, (el primer método mencionado de extracción de información, método clásico) de lectura a conciencia y cálculo de palabras clave, realizan sus correspondientes anotaciones sobre bases de datos o particularmente el campo 650

de los registros MARC[1] para la posterior recuperación mediante IRS[2] buscando las palabras que el analista documental considera representativas del documento.

El proceso de extracción de palabras clave debe realizarse tanto en textos digitales como en ejemplares físicos, acumulando cada vez más los ejemplares que deben ser procesados. Debido a esto, se han desarrollado múltiples algoritmos para la extracción automatizada de palabras clave en textos digitales, con lo que los analistas documentales pueden descargar una parte de su trabajo en la automatización de dicho proceso. Es importante mencionar que estos algoritmos se han desarrollado en la gran mayoría para el idioma Inglés, sin embargo, debido a la estructura de los idiomas, estos algoritmos son aplicables a algunos idiomas con ligeras modificaciones y adaptaciones dependiendo de lo que más adelante se define como *corpus*.

Existen diversos algoritmos y podríamos organizarlos por el tipo de procesamiento que se realiza sobre el texto. Entre ellos, encontramos los de tipo estadístico, lingüístico, semántico, por entropía de la información, por heurística, etc. Aplicables tanto para un corpus (un conjunto de documentos textuales que se relacionan por su contenido) como para ejemplares aislados.

Los algoritmos que se han probado en esta tesis son estadísticos-algebraicos sobre ejemplares aislados, es decir, que sirven para procesar un documento a la vez mediante frecuencias individuales y colectivas de las palabras y cálculos sobre las mismas frecuencias.

A continuación se presenta un listado de algunos algoritmos de extracción de contenido aunque es importante mencionar que no son los únicos.

- Ley de frecuencias de palabras de un texto (ley de Zipf). Este algoritmo se centra en el ordenamiento de frecuencia de las palabras, encontrando que

[1] Machine Readable Cataloging

[2] Information Retrieval System (Sistema de recuperación de información)

entre las palabras más repetidas obtendremos aquellas más fáciles de pronunciar y más cortas (Urbizagástegui & Restrepo, 2011).

- Punto de transición. Consecuencia de la observación de dos fuerzas en la frecuencia de palabras, la unificación y la diversificación, la primera describe el comportamiento de un escritor a emplear términos comunes para el cuerpo del documento, mientras que la segunda, refiere a términos más sofisticados, rebuscados y específicos que detallan el contenido del texto. Por ende, el punto de transición es la región (o como lo llama el autor (Boyce & Lockard, 1975): "vecindario") de frecuencias que describen el contenido del texto sin ser las más frecuentes ni las más rebuscadas (Urbizagástegui & Restrepo, 2011).
- C-value. Es un algoritmo desarrollado para la extracción de términos multipalabra [3] en el idioma inglés, basado en lingüística y estadística adaptable al español en diferentes campos de investigación (Barrón, Sierra, & Villaseñor).
- RAKE. Rapid Automatic Keyword Extraction es un método de extracción de palabras clave en textos basado en estadística y coocurrencia de palabras, es decir, la relación entre la frecuencia de una palabra con la aparición de ésta en un vecindario de palabras, evaluando la frecuencia propia y conjunta (Berry & Kogan, 2010).

El Departamento de Ingeniería Lingüística encabezado por el Dr. Gerardo Sierra en el Instituto de Ingeniería de la Universidad Nacional Autónoma de México, llevó a cabo un importante estudio llamado "c-value aplicado a la extracción de términos multipalabra en documentos técnicos y científicos en español" orientado a la implementación en el español de un algoritmo conocido como c-value, un algoritmo basado en lingüística para la extracción automatizada de términos multipalabra en

[3] Términos descriptores de documento que constan, generalmente, de más de una palabra

textos digitales. La importancia de este estudio radica precisamente en la adaptación del algoritmo al idioma español.

El Dr. Sierra se refiere a su estudio como pionero en la implementación del algoritmo al idioma español.

"...Se presentan los primeros pasos en la implementación de un algoritmo para la extracción automática de términos multipalabra en textos técnicos y científicos, originalmente para el inglés y que se ha aplicado exitosamente para diferentes idiomas, pero no antes en español." (Barrón, Sierra, & Villaseñor).

Dicho estudio revela, entre otros resultados, la versatilidad del algoritmo para implementarse en idiomas con una estructura definida, requiriendo únicamente ajustes a los autómatas para la secuencia de palabras para frases permitidas, así como un listado de stopwords[4] en el idioma deseado y un corpus de entrenamiento para la detección de terminología específica. Extendiendo esta premisa, podríamos determinar que con ligeras modificaciones, un algoritmo de procesamiento de texto para el idioma anglosajón, podría resultar conveniente y aplicable en el idioma español. Por lo que los algoritmos mencionados en el listado pudieron ser empleados para nuestro idioma con algunas variantes.

Una vez conociendo un poco sobre los algoritmos de procesamiento inteligente de texto para extraer descriptores, toca el turno de describir los tres métodos de indización conocidos: Por unitérminos[5], por descriptores[6] y por materias[7] (Sempere Galán, 2012).

---

[4] Palabras vacías, que no proporcionan descripción relevante del texto.

[5] Palabra clave que describe de manera somera el contenido o la temática del documento.

[6] Palabras clave cuyos conceptos representan el documento en el que están contenidos. (Urbizagástegui & Restrepo, 2011).

[7] Utilizar encabezamientos que describan una y sólo una materia en concreto.

El primero de estos métodos, fue planteado por primera vez por Mortimer Taube desde el año de 1955 (Sempere Galán, 2012). Es un método bastante primitivo ya que se extraían las palabras más pequeñas y simples que lograran un acercamiento al contenido del texto. Para éste método, se creaban fichas que contenían una cabecera en la que se escribía un único unitérmino y en el cuerpo de la ficha se anotaban los registros de textos que contuvieran ese unitérmino, posteriormente, para una búsqueda se seleccionaban las fichas que tuvieran las palabras de dicha búsqueda y se comparaban los registros de las fichas seleccionadas, extrayendo así los registros coincidentes en las fichas.

La indización por descriptores es sucesor al método de unitérminos, ya que el encabezado de las fichas en lugar de estar compuesto por un único unitérmino, podía estar compuesto por expresiones un tanto más complejas, reduciendo así el problema de combinar las fichas en el caso de los unitérminos (Sempere Galán, 2012).

El tercer método data de 1876, por Charles A. Cutter (Sempere Galán, 2012). Consiste en determinar un concepto que describa una (y sólo una) materia en concreto, dejando la complejidad en la elección de dicho término y no tanto en la comparación entre diferentes registros.

Entendiendo ahora los procesos de indización, podemos focalizar el problema a resolver, siendo éste la automatización de la extracción de palabras clave para la indización, es decir, debemos concebir un sistema capaz de detectar las palabras que mejor describan el contenido de un texto con la menor intervención de un analista documental en el proceso para optimizar los tiempos en el proceso.

Galán Sempere (Sempere Galán, 2012) describe tres niveles de automatización en el proceso de indizado, los cuales se presentan a continuación:

Nivel 1: Indización asistida por computadora. El indizador humano (analista documental) opera un software que le ayude al proceso conceptual de indización.

Nivel 2: Indización semiautomática. El sistema procesa el texto y propone los términos relevantes al indizador, quien selecciona los que cree correctos.

Nivel 3: Indización automática. El proceso de indización se realiza cien por ciento por el software quien determina los términos y no es necesario que un humano los apruebe.

Si bien es cierto que actualmente existe software dedicado a la indexación como CINDEX[8], SKY index[9], entre otros. Es software propietario, desarrollado para el idioma inglés y no dedicados para la indización sino la indexación.

Teniendo en cuenta la necesidad de las bibliotecas de la UNAM de buscar una manera alterna de resolver el problema que en el siguiente subcapítulo se presenta, se decide la realización de la presente tesis y hacer frente a los cambios tecnológicos que representan una oportunidad de generar conocimiento a partir de la extracción de palabras clave para un indizado de documentos preciso. De tal forma que se facilite, a posteriori, la recuperación de documentos principalmente para realizar investigaciones que, de otra forma, sería más complicado encontrar las fuentes adecuadas.

[8] Software propietario que prepara índices para libros (CINDEX, s.f.)

[9] Software propietario para indexación (SKY Software, 2015)

1.2. PROBLEMA A RESOLVER Y TIPO DE ESTUDIO

A pesar de que la mayoría del material que llega a las bibliotecas de la Universidad Nacional Autónoma de México contiene un conjunto de palabras clave predefinidas para su indización, propuestas por el mismo autor (o editor, en algunos casos), en ocasiones estas palabras clave no abarcan el total de términos existentes en el cuerpo del texto necesarios para una correcta indización.

Es por esto, que un analista documental procesa previamente cada material para su indización e indexación. Ambos procesos son sumamente importantes y se relacionan de un modo que en ocasiones resulta complicado discernir entre uno y otro.

Como ya se mencionó, el proceso de indexación consiste básicamente en tener los elementos de un conjunto ordenados y posteriormente, generar el índice que facilite el acceso a éstos para la recuperación correspondiente, tal como sucede con los índices de los libros, donde los temas están ordenados de tal forma, que es posible acceder a ellos mediante una búsqueda superficial desde ésta parte del libro, la cual nos entrega la “posición” o ubicación de dicho tema en la estructura del libro.

Por otro lado, el proceso de indización es fundamental para la recuperación de información de una manera más precisa ya sea de manera manual o mediante los motores de búsqueda a los que los usuarios tienen acceso.

La extracción de términos descriptores de ejemplares en las bibliotecas de la UNAM, es una de las actividades que realiza un analista documental, extracción de frases que pueden ser almacenadas en registros MARC. El analista procesa el documento (artículo, revista, libro, etc.) ya sea en formato impreso o digital y manualmente genera un listado de palabras que considera relevantes y posteriormente se realiza un volcado de dichas palabras (y en pocos casos frases descriptoras) en una base de datos, en registros MARC o en algún otro tipo de documento al cual se pueda acceder para recuperar por contenido.

Este proceso puede durar días o incluso semanas solamente para la extracción de keywords[10], lo que se traduce en mayor tiempo de espera del ejemplar desde su adquisición hasta su puesta a consulta para los usuarios de la biblioteca.

Si bien la labor del analista documental no se limita a la extracción de palabras clave que solo es parte del proceso de indizado (y en su caso, generación de registros MARC), no es su única actividad. Entendemos ahora que las actividades resulten exponenciales debido a la acumulación de ejemplares tanto digitales como físicos a la espera de ser procesados para la posterior consulta de los usuarios. Es por esto que se requiere una herramienta de software que automatice el proceso de extracción de palabras clave y términos multipalabra de manera eficiente y que reduzca el tiempo que transcurre entre el momento que es adquirido hasta que puede ser consultado por un usuario de las bibliotecas.

Es por ello, que la extracción manual de descriptores resulta un problema, ya que depende tanto del número de documentos textuales a procesar, como del tiempo que toma a una persona leer el documento y extraer los descriptores relevantes, omitiendo palabras que podrían resultar relevantes, incluyendo descriptores que no lo son y, en muchos casos, ignorando términos multipalabra.

Es importante mencionar que el sistema requerido aplica exclusivamente a textos en formato electrónico y no así a ejemplares físicos.

[10] Término en inglés para palabras clave o descriptoras de contenido.

1.3. JUSTIFICACIÓN

La necesidad de almacenar, indizar, y recuperar los grandes volúmenes de documentos en formato digital ha generado múltiples nuevas problemáticas en el ámbito bibliotecológico. Es por eso que atendiendo a la necesidad de herramientas tecnológicas para la agilización del proceso de indización, se inició esta investigación y con ello, al desarrollo del sistema que permita la extracción automatizada de los descriptores más representativos de los documentos procesados para ser utilizados en el proceso de indización (y, de ser necesaria, la posterior recuperación manual o mediante IRS).

1.4. OBJETIVOS

OBJETIVO GENERAL

Desarrollar un sistema de cómputo basado en métodos estadísticos y algebraicos de procesamiento de ejemplares digitales para la extracción de keywords (palabras clave y términos multipalabra) necesarios para la optimización del proceso de indización de ejemplares en Bibliotecas de la Universidad Nacional Autónoma de México.

OBJETIVOS PARTICULARES

- Facilitar el proceso de extracción de palabras clave y términos multipalabra realizado por analistas documentales.
- Mejorar el tiempo requerido para el proceso de extracción de palabras clave y términos multipalabra para el proceso de indizado de ejemplares digitales en bibliotecas de la UNAM.

1.5. HIPÓTESIS

El desarrollo de una herramienta de software para extracción de palabras descriptoras de contenido basada en algoritmos estadísticos y algebraicos con detección de términos multipalabra aplicados en textos digitales, facilita el proceso de indización en bibliotecas de la UNAM.

1.6. Alcances del desarrollo

El desarrollo de la herramienta, permite el análisis objetivo de los textos digitales adquiridos por las bibliotecas de la UNAM, teniendo una primera etapa de entrenamiento en Biblioteca Central, particularmente en el área de la ingeniería en el idioma español. Quedando como miras secundarias el entrenamiento del sistema en diferentes ramas del conocimiento.

Capítulo 2. Introducción a ingeniería de software, a bases de datos y a estadística

INTRODUCCIÓN

En este capítulo se hará referencia a todo el marco teórico que brindará el soporte técnico para el entendimiento de esta tesis, haciendo un breve resumen de algunas de las asignaturas de la carrera de ingeniería en computación que facilitaron la elaboración del presente escrito.

En el presente capítulo se hará un breve repaso de ingeniería de software, bases de datos y estadística. Ya que son los temas más importantes para el desarrollo de esta tesis.

2.1. INGENIERÍA DE SOFTWARE

La ingeniería de software es descrita por Sommerville en (Sommerville, 2001) como una disciplina ingenieril a la que le conciernen todos los aspectos de la producción de software. Mientras que Mills (Mills, 1980) la describe como el diseño y desarrollo sistemático de productos de software y la administración del proceso del software.[11] Y por el Dr. Daniel Trejo (Medina Trejo, 2010), como el desarrollo de software de manera ágil y madura, siguiendo técnicas y metodologías que aporten valor al usuario y en su beneficio.

Entonces, la diferencia entre ingeniería de software y la ingeniería en computación, radica en el punto de aplicación de los conocimientos. Mientras que a la ingeniería en computación concierne la teoría y métodos que dan base a la computación, la ingeniería de software se relaciona con los problemas prácticos en el desarrollo de software.

De mi experiencia en la Facultad de Ingeniería, UNAM, puedo constatar que mucha de la teoría de ingeniería en computación se basa en parámetros ideales o valores predefinidos, calculados previamente, por lo que en general, no son aplicables a la realidad sin previas modificaciones para poder adaptarse al entorno en el que se desea aplicar; básicamente se desea establecer las bases del conocimiento a emplear posteriormente en el ámbito laboral. En contraste, la ingeniería de software, al seguir técnicas y metodologías probadas y estandarizadas por los diversos entes de la industria orientada precisamente al desarrollo de software, es renovada constantemente según las necesidades de la industria y se adapta, según sea el caso, con el modelo, la técnica o el estándar adecuado para cada situación.

[11] Software engineering may be defined as the systematic design and development of software prod-ucts and the management of the software pro-cess.

2.1.1. Software de calidad

Debido a que el software es un producto intangible[12], (a diferencia de los productos de otras disciplinas en la ingeniería) y que representa una abstracción, pudiéramos decir que cada desarrollo podría obedecer a sus propias reglas (Winkler, Biffl, & Bergsmann, 2013). Sin embargo, los acuerdos y estándares adoptados por diversas empresas y organizaciones, permiten definir procesos mediante los cuales podemos evaluar un producto de software como bueno, obviamente pasando por alto el enfoque del software. Su propósito específico. Y tomando únicamente las características generales que lo componen.

Si bien esta tesis no pretende ser un manual sobre cómo desarrollar software de calidad, es importante mencionar los aspectos que se toman en cuenta para el desarrollo del sistema desarrollado.

Podemos definir las características de un software de calidad según sea el área del software desarrollado. Por ejemplo, para los sistemas de consejo de toma de decisiones, se mencionan en el libro de Suryn (Suryn, 2014) como exactitud, analizabilidad e idoneidad. Mientras que para los sistemas transaccionales, las características más representativas de la calidad se miden en funcionalidad, confiabilidad, usabilidad y eficiencia. Para los sistemas de transacciones financieras, las características principales para ser considerado de calidad, son la funcionalidad y la confianza.

En resumen, funcionalidad, confiabilidad, usabilidad y eficiencia. Otras características del software de calidad son descritas en la tabla 2.1, en la cual se especifican 4 características pilares de un software de calidad.

[12] Resulta intangible per se, sin embargo, los resultados de un software carente de calidad pueden verse reflejados como pérdidas económicas, de salud y, en el peor de los casos, la vida.

Característica	Descripción
Mantenimiento	Debido a que el cambio de los requerimientos de sistemas con respecto del tiempo es inevitable, es necesario que el software sea desarrollado de una manera tal que permita la evolución de una manera sencilla para cubrir con las nuevas necesidades.
Confianza	Un sistema debe ser fiable y seguro
Eficiencia	Un sistema eficiente es aquel que hace un óptimo uso de los recursos con que cuenta, tales como memoria, tiempo de procesamiento o espacio en disco.
Usabilidad	La usabilidad es un atributo que hace referencia a una buena interfaz de usuario, de tal manera que permita una mejor interacción entre el sistema y la persona que lo utilizará.

TABLA 2.1 Software de calidad. Fuente: (SOMMERVILLE, 2001)

2.1.2. CICLO DE VIDA DEL SOFTWARE

Todo sistema desarrollado obedece a un ciclo de vida, en el cual se establece cada etapa conocida de la vida útil de un sistema.

El software, como cualquier desarrollo por la humanidad, obedece a cierto patrón: el ciclo de vida. Mientras que para un ser vivo, el ciclo es nacer, crecer, reproducirse y morir. Para el software es un tanto más complejo, ya que, dependiendo de su criticidad, de éste puede depender la pérdida de dinero, el fracaso de proyectos e inclusive el deceso de personas.

Dependiendo del tipo de desarrollo, los desarrolladores y la organización, el ciclo de vida del software puede tomar ciertas características, regresiones a estados anteriores o el desmantelamiento y reutilización de código.

En términos generales, el ciclo de vida se conforma por una buena la planeación (con aspectos como las especificaciones y el diseño), la ejecución del plan (con aspectos como el desarrollo e implementación) y el mantenimiento (donde se evalúa el desarrollo para actualizar el software, el desmantelamiento y la reutilización de código).

En la ilustración 2.1 se muestra el ciclo de vida del software.

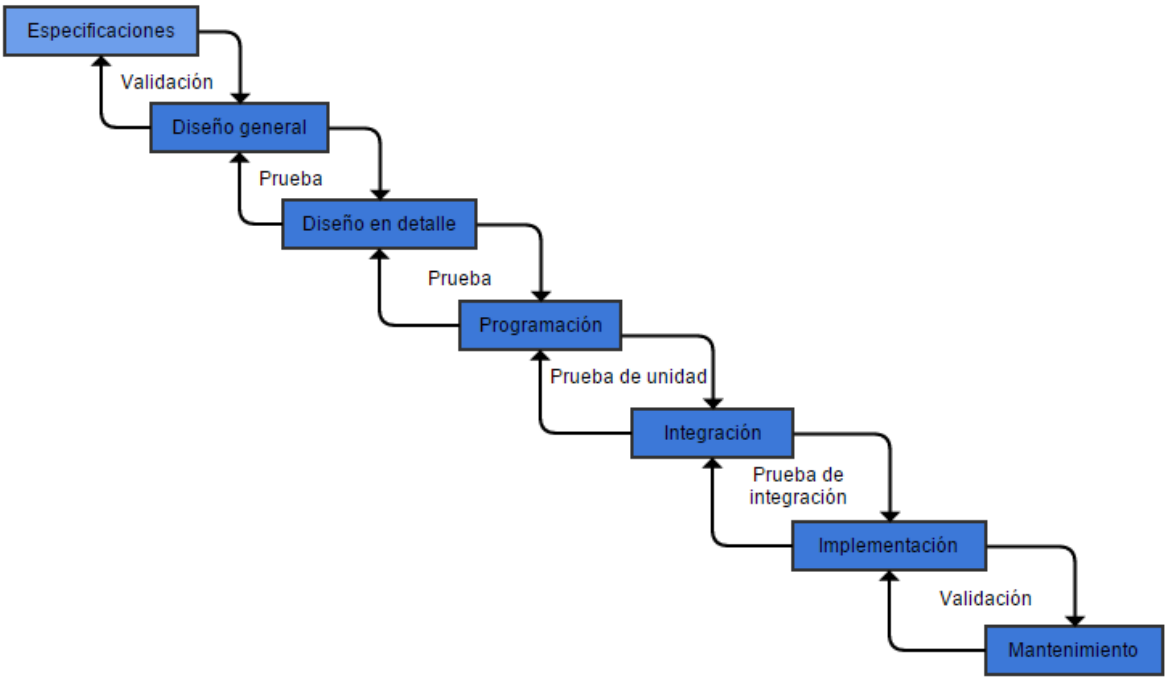


ILUSTRACIÓN 2.1 Ciclo de vida del software. Fuente: (SOMMERVILLE, 2001)

En la ilustración 2.1, podemos observar la relación que guarda cada bloque tanto con su etapa posterior como la o las anteriores, esto nos da a entender que de cada etapa del ciclo de vida, el comportamiento natural es llegar a la etapa siguiente, sin embargo, al ser un producto intangible, en un mundo en constante cambio, puede ser modificado en cualquiera de sus etapas; incluso en el ocaso de su funcionalidad para poder reestructurar una de sus etapas anteriores y resurgir adaptado a las nuevas necesidades con funcionalidades no previstas en un comienzo. Esto generalmente implica una mayor inversión de recursos; generalmente el más costoso es el tiempo, por lo que para evitar en la mayor medida la necesidad de regresar a una etapa anterior, la toma de requerimientos (en el análisis) resulta una de las partes más importantes (si no es que la más importante) en el ciclo de vida del software ya que de ella dependen las actividades a realizar, la realización del

presupuesto y asignación de recursos a las actividades para el desarrollo. Es por ello, que se debe ser bastante cauteloso al momento de decidir cuándo se ha descrito de manera clara y satisfactoria el producto solicitado.

Es importante remarcar el hecho que mientras más avanzado está el proyecto, mayor será el costo de cambios sobre etapas anteriores, no sólo por recursos monetarios sino por tiempo y retraso de entregables en las etapas subsecuentes.

Una manera de facilitar la *toma de requerimientos* del sistema y su *diseño*, es el modelado del sistema, dicho modelado permite de manera gráfica observar mediante un diagrama los componentes y la relación que existe entre ellos. Es así que se puede tener una idea a grandes rasgos de los elementos (o módulos) que conformarán al sistema, así como una descripción de la relación (y/o dependencia) entre módulos.

Generalmente, el modelado del sistema se realiza como un diagrama de bloques el cual muestra, como se menciona anteriormente, los subsistemas como bloques y su relación con otros subsistemas. Se denota mediante flechas que marcan la dependencia o interdependencia de un módulo con respecto a otro, así como el flujo de datos entre ellos.

Si pudiéramos hacer un acercamiento a cada bloque del modelado del sistema, encontraríamos uno o más componentes funcionales, es decir, un elemento del diagrama de bloques está compuesto por al menos un componente que realiza una tarea en específico, llamado componente funcional.

Existen diferentes modelos de desarrollo de software, tales como el modelo de desarrollo en cascada, evolutivo, desarrollo de sistemas formales y desarrollo basado en reutilización, etc.

Cabe mencionar que no son únicos ni absolutas soluciones a todos los desarrollos y que incluso, puede haber una mezcla de éstos, dependiendo del tipo de desarrollo y de las necesidades de la solución.

El modelo de desarrollo en cascada, empleado en el desarrollo del sistema de esta tesis, es descrito en (Sommerville, 2001), toma los procesos fundamentales de cada etapa de la ingeniería de software y los representan como cinco actividades fundamentales.

- Análisis y definición de requerimientos. Se detallan los alcances, objetivos y restricciones del sistema.
- Diseño del sistema. Se establece una arquitectura del sistema así como una abstracción de subsistemas y sus relaciones.
- Implementación y pruebas unitarias. En esta etapa, el sistema se asimila como un conjunto de programas unitarios y se verifica que cada uno de ellos esté cumpliendo con las especificaciones de manera correcta.
- Integración y pruebas de sistema. Los programas definidos y probados en la etapa anterior, son integrados y probados como un solo sistema para asegurar que cada requerimiento se haya cumplido. Luego de las pruebas, el software puede ser liberado al usuario.
- Operación y mantenimiento. Generalmente es la parte más larga del ciclo de vida del software ya que es cuando se entrega para ser utilizado y los errores no detectados en el desarrollo inicial son corregidos.

La ilustración 2.2 muestra el modelo de desarrollo en cascada propuesta por Isaac Arteta (Arteta, s.f.), donde podemos apreciar que son básicamente las mismas etapas aunque con un nombre simplificado.

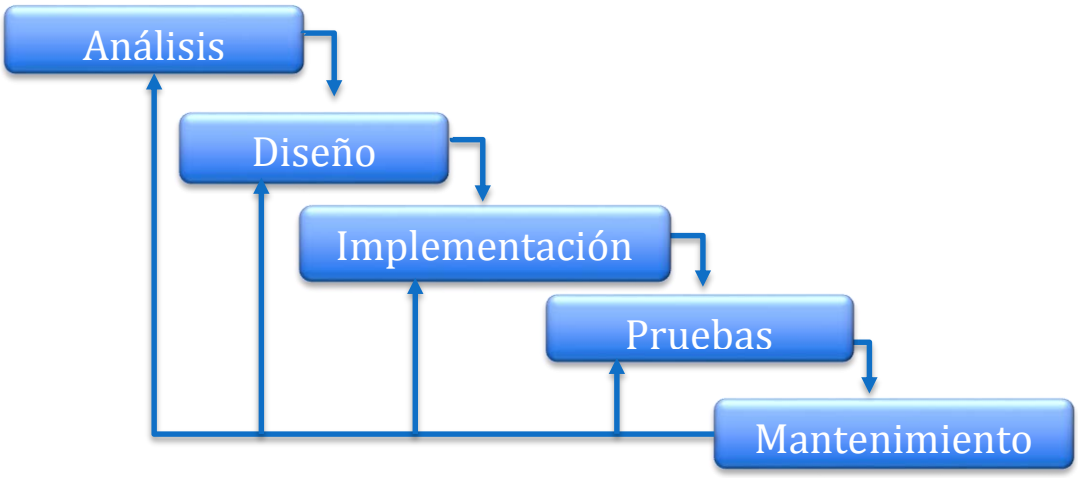


ILUSTRACIÓN 2.2 Modelo de desarrollo en cascada. Fuente: (ARTETA, S.F.)

Para la definición de requerimientos de un sistema se necesita una interacción con el usuario final del sistema, está orientado a descubrir los requerimientos del sistema en su conjunto. Sommerville (Sommerville, 2001) menciona que se encuentran tres tipos de requerimientos:

- Requerimientos funcionales. Comprende las funciones básicas del sistema.
- Propiedades del sistema. Abarca las propiedades no funcionales del sistema pero que influyen directamente en los requerimientos de subsistemas. Tales como la disponibilidad, la seguridad y el rendimiento, entre otros.
- Características que el sistema no debe exhibir. Es una manera de purgar las opciones para el usuario final, de tal manera de volverlo más usable.

Este es un proceso sumamente importante debido a que esta es la etapa donde se definen los acuerdos con respecto al sistema y siendo la base de todo el desarrollo, debe estar explícito cada detalle que se pretende del sistema, es recomendable “desmenuzar” cada problema en partes más simples de tal manera que en las etapas siguientes se pueda desarrollar, implementar y validar que cada requerimiento se ha cumplido.

En la etapa del diseño del sistema, se define la manera en que cada componente aportará para la funcionalidad entera del sistema.

En la ilustración 2.3, extraída del libro de Ian Sommerville (Sommerville, 2001), se puede observar el conjunto de pasos a seguir para un correcto diseño del sistema.

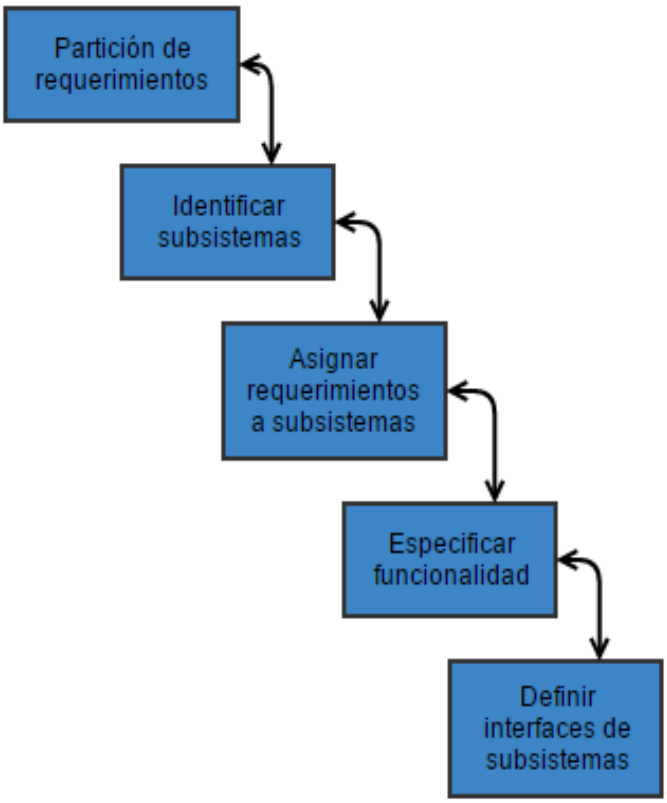


ILUSTRACIÓN 2.3 Diseño de un sistema. FUENTE: (SOMMERVILLE, 2001)

- Particionar los requerimientos. En este punto, los requerimientos son analizados y agrupados.
- Identificar subsistemas. Dependiendo de la agrupación de los requerimientos, se identifican los subsistemas que se necesitarán.
- Asignar requerimientos a subsistemas. En base a la agrupación de requerimientos y a los subsistemas identificados, se entrelazan para su solución.
- Especificar la funcionalidad de cada subsistema. Se definen las funciones específicas de cada subsistema así como las relaciones entre subsistemas.
- Definir interfaces de subsistema. La definición de interfaces de cada subsistema permite el desarrollo en paralelo de los subsistemas.

Llega el momento de aplicar un viejo y conocido dicho “Divide y vencerás”. Ya que al fraccionar el problema en pequeños subsistemas encargados de una tarea en específico, se consigue mayor flexibilidad en el desarrollo, (futura) reutilización de

código y, probablemente, lo más importante en el desarrollo de un sistema complejo y robusto: facilidad en la administración de recursos para el desarrollo de dicho sistema.

Generalmente, los subsistemas pueden ser desarrollados en paralelo, siempre y cuando, el diseño logrado en la etapa anterior lo permita, esto quiere decir que para que los subsistemas puedan ser desarrollados en paralelo, no debe existir una dependencia crítica entre ellos.

La flexibilidad que otorga un modelado correcto del sistema permite
modificaciones para adaptar a las necesidades de este punto.

Una vez que los subsistemas fueron desarrollados por separado, la siguiente tarea es “ponerlos todos juntos” y hacer que trabajen e interactúen de manera correcta para lograr así un sistema completo. A pesar de que puede realizarse una integración de todos los subsistemas al mismo tiempo, por razones de manipulación, es recomendable realizar una interacción incremental parcial, es decir, realizar la unión de subsistemas de manera paulatina para llevar un mejor control sobre la interacción y operación entre los subsistemas.

Además, aunque los subsistemas sean desarrollados de manera paralela, eso no garantiza que el desarrollo de éstos culmine simultáneamente, por lo que de esperar hasta la finalización del subsistema más complejo, la integración de subsistemas tomará más tiempo que si se integran de manera paulatina los subsistemas conforme se terminan éstos.

Otro aspecto a tomar en cuenta para la integración de subsistemas consiste en la facilidad de la detección de errores. De manera intuitiva podemos entender que al ir conjuntando los subsistemas de manera dosificada, será más fácil detectar cuándo un subsistema está trabajando de manera correcta y cuándo no.

Una vez que todos los subsistemas han sido integrados satisfactoriamente y solucionados los conflictos que esto pudo ocasionar, es tiempo de colocar al sistema en el ambiente sobre el cuál funcionará. Para esto, Sommerville (Sommerville, 2001)

sugiere tomar en cuenta algunos aspectos como el entorno sobre el cual el sistema será instalado, sobre todo cuando no es el mismo que sobre el que fue desarrollado; los usuarios potenciales y su curva de aprendizaje del sistema; la coexistencia del sistema con otros sistemas en el mismo entorno y los posibles problemas físicos en el momento de la instalación, tales como el espacio, el cableado, el ambiente, el mobiliario entre otros.

Posterior a la instalación del sistema en el ambiente final donde se ejecutará, puede resultar necesario un tiempo de aprendizaje por parte del operador, lo cual requiere sesiones de entrenamiento para el correcto uso del sistema.

Por otro lado, la operación del sistema puede verse afectada por factores que no se tomaron en cuenta en el proceso de diseño, tales como limitantes de hardware, convivencia con otros sistemas o incluso causar confusión al operador por utilizar múltiples sistemas en el mismo entorno.

Como se mencionó anteriormente, el ciclo de vida de los sistemas puede llegar a un punto en que requiera una reestructuración de alguna etapa anterior para poder seguir brindando solución.

Es por eso que la mayoría de sistemas robustos suelen tener un ciclo de vida bastante largo, ya que deben evolucionar tanto cuando las necesidades del usuario cambien como para la corrección de errores en el diseño original.

La evolución de un sistema debe hacerse de manera cuidadosa y observando minuciosamente los detalles sobre los cuales el cambio pudiera tener repercusiones.

No es tan fácil determinar cuándo un sistema debe evolucionar y cuándo dar paso a la generación de un nuevo sistema que lo reemplace. De hecho, en algunas ocasiones resulta más sencillo tomar nuevamente los requerimientos de las necesidades emergentes que modificar un sistema diseñado en el pasado para necesidades del pasado.

Cuando se ha decidido por la finalización de la utilización de un sistema, su ciclo de vida ha llegado a su fin, sin embargo, puede darse el caso que algunos de los subsistemas generados anteriormente pueda ser reutilizado en el nuevo sistema. Aunque en algunos casos puede llegar a ser costoso, un error de decisiones en el pasado puede llevar a que el usuario “se case” metafóricamente con un sistema o partes del mismo, impidiéndole así desecharlo del todo. En otros casos, los datos deben ser modificados para la correcta implementación del nuevo sistema, como es el caso de las bases de datos.

2.2. BASES DE DATOS

Un dato por si solo es una representación simbólica (Wikipedia, 2015), siendo uno solo, se conoce como dato aislado, mientras que en conjunto, una vez procesados y relacionados entre sí para una interpretación se consideran como información (Informática Moderna, 2015).

Las bases de datos ofrecen una alternativa poderosa para almacenar y gestionar grandes cantidades de información. Una base de datos es una gran cantidad de información sistematizada y procesada para un almacenamiento, ordenamiento y posterior consulta.

2.2.1. Tipos de bases de datos

Existen varios tipos de bases de datos (Academia, 2015), (Academia, 2015), como las siguientes descripciones:

- Modelos tradicionales. Son aquellos utilizados por las primeras empresas en informatizarse. Consisten en ficheros electrónicos simples que contienen los datos. Son Manejados directamente a través del sistema operativo y no con un sistema de gestión de bases de datos.
- Modelo jerárquico. Tiene como objetivo establecer una jerarquía de fichas, de tal manera que cada ficha pueda contener a su vez listas de otras fichas y así sucesivamente.
- Modelo en red. Su estructura es parecida a la del modelo jerárquico, aunque más compleja. Define varios conceptos:
    - Registro: Es la abstracción de cada una de las fichas almacenadas en un fichero convencional.
    - Campos o elementos de datos: Cada uno de los elementos que componen y definen una ficha.

  - Conjunto: es el concepto que permite relacionar diferentes tipos de registro.

- Modelo relacional. Modelo que define la base de datos como el conjunto de tablas, procesos y relaciones entre sí que dan lugar a la integridad de los datos almacenados en ella.
- Modelos orientados a objetos. Trata los problemas desde un punto de vista realista y modelando cada uno de ellos como abstracción de un conjunto de elementos que interrelacionan entre sí para resolver el problema. Este modelo define varios conceptos:
  - Clase: conjunto de atributos que definen las características generales de los objetos. Los objetos que comparten dichas características se agrupan en la misma clase.
  - Estado: son las características propias de cada objeto, nos permite guardar la situación del objeto que varía con el tiempo.
  - Encapsulación: Es la manera que el objeto “se comporta” ante estímulos tanto del exterior como de otro objeto, es decir, para que un sistema funcione, los objetos envían mensajes a otro objeto, dicho objeto “sabrá” (previamente definido) qué hacer o cómo comportarse al recibir los mensajes.
  - Mensaje: cada uno de los estímulos que se envían a un objeto.
  - Herencia: Es la jerarquía con que se determinan las clases, es decir, los atributos que definen a una clase, definirán a clases que sucedan de ésta.

2.2.2. Bases de datos relacionales

Al almacenar información en una base de datos se pretende obtener fiabilidad[13], integridad[14] y disponibilidad[15] de dicha información. El modelo relacional facilita dichas características al trabajar de manera conjunta tablas y procesos que se relacionan entre sí.

Churcher (Churcher, Beginning SQL Queries From novice to professional, 2008) define a las bases de datos relacionales como un conjunto de tablas[16] que almacenan información sobre aspectos de cierto tema

En el modelo relacional, una tabla es una matriz bidimensional compuesta por filas (elementos) y columnas (atributos). La intersección entre una fila y una columna arroja un registro (celda), este registro describe el atributo o particularidad de un elemento para una condición.

En el mundo de las bases de datos, una relación es una manera de ligar registros de dos entidades diferentes.

Churcher también describe diferentes tipos de relación (Churcher, Beginning database design from novice to professional, 2007), los cuales son listados a continuación:

- Uno a uno: cada registro de una entidad tiene uno y solo un registro ligado de la segunda entidad y viceversa.

---

[13] Fiabilidad: Probabilidad es la probabilidad de buen funcionamiento de algo (Delgado García, 2015).

[14] Integridad: Cualidad de un archivo que permite comprobar que no ha sido alterado (Delgado García, 2015).

[15] Disponibilidad: Característica de la información de poder ser accedida por quien debe acceder a ella (Delgado García, 2015).

[16] Aunque posteriormente hace hincapié que la definición es bastante somera, menciona que es un conjunto de relaciones

- Uno a muchos: cada registro tiene uno o más registros ligados de la segunda entidad.
- Muchos a muchos: Cada registro en la primera entidad puede tener uno o más registros ligados en la segunda entidad y esos registros (de la segunda entidad) pueden también tener varios registros ligados en la primera entidad.

Una vez almacenada la información, para volverla útil, se procede a su extracción y consulta. La manera de obtener los registros en una base de datos relacional se realiza a través de un lenguaje estructurado de consultas (SQL[17] por sus siglas en inglés) que cada DBMS [18] (o RDBMS [19] en particular) complementa con instrucciones particulares.

Los DBMS son paquetes de software, y en ocasiones hardware, diseñado especialmente para almacenar y recuperar información, con los cuales se implementan las aplicaciones de bases de datos.

La estructura (general) de un DBMS se describe en la ilustración 2.4, donde podemos apreciar la vía mediante la cual, un usuario de la base de datos puede acceder a la información almacenada como registros en una base de datos, pasando por la línea de comandos para sentencias SQL u otras interfaces conocidas como GUI[20], llegando al sistema de gestión de la base de datos y por último a la información almacenada en disco.

[17] SQL: Structured Query Language. Más adelante se detalla su utilización.

[18] DBMS: Database Management System

[19] RDBMS: Relational Database Management System

[20] GUI: Graphical User Interface. (Interfaz gráfica de usuario)

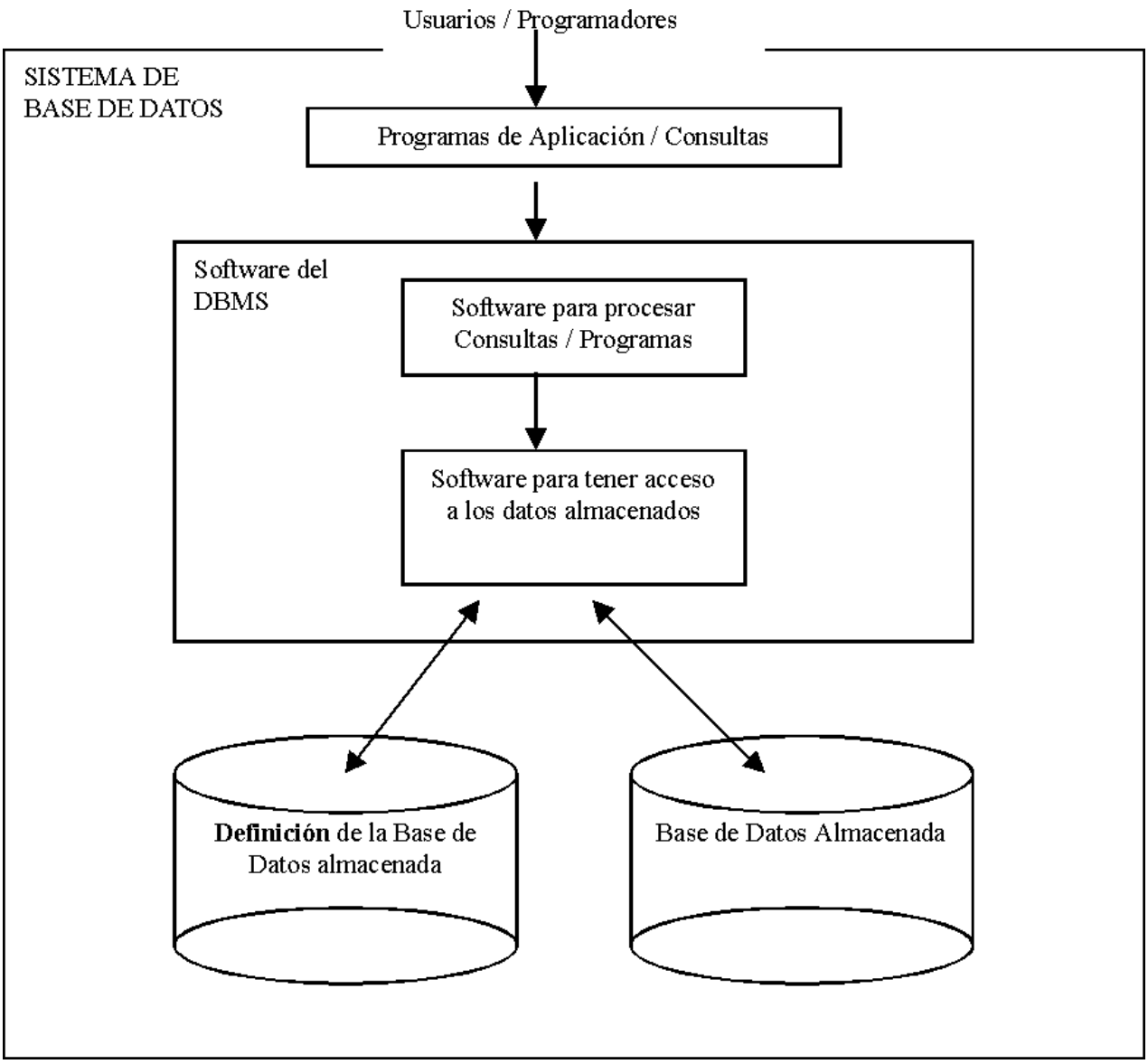


ILUSTRACIÓN 2.4 Estructura de un DBMS Fuente: (UNIVERSIDAD NACIONAL DE COLOMBIA, S.F.)

Algunos ejemplos de DBMS son:

- MySQL. DBMS desarrollado en C y en C++. Multiplataforma, multithread por kernel, seguridad por privilegios y contraseñas. Es sumamente escalable, hasta 64 índices por tabla (cada índice puede ir desde 1 hasta 16 columnas).
- Microsoft SQL srerver. Gran cantidad de herramientas administrativas y de desarrollo, incluye herramientas para el proceso analítico en línea, herramientas gráficas de diseño de base de datos.
- Oracle. Es hasta ahora el DBMS más poderoso en el mercado de motores de gestión de base de datos. Primer base de datos diseñada para cómputo en Grid, Diseñado para manipulación de grandes volúmenes de información. Posee una amplia gama de funciones exclusivas.
- PostgreSQL. Sistema de gestión de bases de datos relacionales publicado bajo BSD (Berkeley Software Distribution). Heredero de la base de datos Ingres (de ahí su nombre “Post-Ingres”). Tiene un buen manejo de alta concurrencia, Amplia variedad de tipos nativos, llaves foráneas, triggers, vistas, etc.
- SQLite. Sistema de gestión de bases de datos relacionales compatible con ACID (Atomic, Consistent, Isolated, and Durable), contenido en una biblioteca en C. Las definiciones, tablas, índices, y datos son almacenados en un sólo fichero en el servidor.
- Firebird. Sistema de administración de base de datos relacional de código abierto basado en la versión 6 de Interbase. Multiplataforma. Medianamente escalable. Seguridad basada en usuarios y roles.
- DB2. (Universal database) Desarrollado por IBM. Escalable, veloz y confiable. Sistema de administración de bases de datos relacionales multiplataforma, diseñado para ambientes distribuidos.

La ilustración 2.5 muestra algunos de los logotipos de los DBMS mencionados en el listado anterior.

ILUSTRACIÓN 2.5 Algunos de los DBMS más populares. Fuente: Collage propio. Cada imagen pertenece a las respectivas marcas.

Actualmente Oracle es el DBMS de paga más utilizado por las empresas, seguido por SQL Server de Microsoft, mientras que MySQL lleva la delantera en popularidad como solución libre seguido, de lejos, por PostgreSQL.

Aunque recientemente han surgido nuevos motores de bases de datos no relacionales, (incorrectamente) llamados NoSQL, orientados a documentos como lo son CasandraDB y MongoDB. Este último, marcando un referente en cuanto al desempeño y flexibilidad de almacenamiento. Ya que a diferencia de los motores de base de datos relacionales, en MongoDB no existe una estructura predefinida donde se almacena la información, como lo son las tablas en un RDBMS. Cada registro se almacena como documento en una colección de documentos, los cuales pueden tener diferentes definiciones y estructuras internas, guardando si acaso una única relación con los demás registros.

2.2.3. LENGUAJE ESTRUCTURADO DE CONSULTAS (SQL)

Structured Query Language, es el lenguaje estándar para el manejo de datos almacenados en una base de datos relacional.

Es un lenguaje intuitivo y fácil de aprender que permite interactuar con la base de datos a través de terminales, embebido en programas o utilerías propias de los manejadores de base de datos.

SQL permite realizar muchas tareas comunes sobre los datos como seleccionar, actualizar, ordenar, calcular valores sobre datos almacenados, copiar y combinar los datos de las tablas.

SQL está formado, además, por tres sublenguajes: DDL (Data Definition Language), DML (Data Manipulation Language) y DCL (Data Control Language).

2.2.3.1. DDL LENGUAJE DE DEFINICIÓN DE DATOS

DDL es un lenguaje del DBMS que permite llevar a cabo tareas de definición de estructuras que almacenarán los datos en la base de datos y procedimientos y funciones para consultarlos.

DDL está conformado por tres sentencias permitidas para la definición de la estructura de datos en una base de datos, las cuales se muestran en la tabla 2.2.

Sentencia	Uso	Ejemplo
CREATE	Crea un objeto dentro del manejador de base de datos como tablas, índices, vistas, procedimientos, triggers, usuarios, bases de datos, etc.	CREATE table mitabla(campo1 tipo1,...,campoN tipoN);
DROP	Elimina un objeto dentro del manejador de base de datos.	DROP table mitabla;
ALTER	Una vez que el objeto dentro del manejador de base de datos ha sido creado, puede ser modificado mediante este comando	ALTER table mitabla ADD campo3 tipo3;

TABLA 2.2 Ejemplo de sentencias permitidas en DDL. Fuente: Propia.

La lista de opciones que brindan estos comandos parecen ser interminables debido a la facilidad que permite SQL de combinar opciones y propiedades de cada objeto de base de datos, además, dichas propiedades de objetos, así como la correcta sintaxis de las sentencias pueden variar (las últimas ligeramente) dependiendo del manejador de base de datos.

Dentro del Lenguaje de Definición de Datos, se consideran ciertas restricciones de integridad que definen reglas que determinan los valores permitidos dentro de las columnas en las tablas de las bases de datos. Además, aseguran que los cambios que se realicen sobre la base de datos, no causen inconsistencia en la información Las siguientes restricciones son mencionadas en (Churcher, Beginning database design from novice to professional, 2007).

Llave principal (PK)

Cuando se tienen múltiples registros en una tabla, se corre el riesgo de duplicidad en la información, es decir, que por accidente o descuido, se tenga dos o más veces el mismo registro, pudiendo comprometer la integridad de la información, además de ocupar más espacio en el almacenamiento y ralentizando el proceso de búsqueda a lo largo de la tabla. Es por ello que se requiere una regla que impida el ingreso múltiple de un registro. Esta regla se llama Llave primaria y sirve precisamente para distinguir entre todos los registros de una tabla.

Esta restricción indica que una columna o un conjunto de ellas identifican al registro en la tabla. Se establece mediante las palabras reservadas PRIMARY KEY.

Llave Foránea (FK)

La llave foránea, por otro lado, es un campo o conjunto de ellos, en una tabla que refiere a la llave primaria de alguna tabla.

Esta restricción identifica una relación existente entre un registro de una tabla refiriéndose a la llave primaria de otra. Sus palabras reservadas son FOREIGN KEY.

NO NULO

Ya que al manejar la información, se puede encontrar el caso de que no se tenga cierto dato en el momento, los DBMS permiten dejar “campos sin llenar”, pero si en cierto caso es necesario garantizar que un campo no quede en nulo (sin información), es necesario establecer la restricción NOT NULL. Esta restricción prohíbe la inserción de datos nulos en un registro de tabla en la base de datos. Sus palabras reservadas son NOT NULL.

REGISTRO ÚNICO.

Similar a la llave primaria, esta restricción, obliga a que para la columna (o conjunto de columnas) definida, no existan dos registros iguales en la tabla, a diferencia que éste no identifica a los registros de la tabla. Su palabra reservada es UNIQUE.

RESTRICCIÓN DE CUMPLIMIENTO.

Esta restricción obliga a que los datos ingresados en la tabla cumplan con cierta condición (conjunto de valores, cadenas, fechas o caracteres entre otros) para poder ser almacenados. Su palabra clave es CHECK.

2.2.3.2. DML LENGUAJE DE MANIPULACIÓN DE DATOS.

Este lenguaje, subconjunto de SQL, proporciona a los usuarios una manera de consultar y modificar los datos almacenados en las bases de datos del DBMS.

Las sentencias permitidas, están descritas en (Churcher, Beginning SQL Queries From Novice to Professional, 2007) y ordenadas en la tabla 2.3.

Sentencia	Uso	Ejemplo
SELECT	Este comando permite al usuario extraer los datos de alguna tabla en la base de datos.	SELECT campo1 from mitabla [WHERE cond]
UPDATE	Este comando, brinda la posibilidad de actualizar los datos contenidos en una tabla de la base de datos sin la necesidad de generar un nuevo registro.	UPDATE mitabla set campo1=valor WHERE cond
INSERT	Comando utilizado para agregar nuevos registros a una tabla en la base de datos.	INSERT into mitabla (campo1) values ('valor1');

TABLA 2.3 Ejemplos de DML. Fuente: Propia. Extracción de (CHURCHER, BEGINNING SQL QUERIES FROM NOVICE TO PROFESSIONAL, 2007)

La palabra clave DISTINCT es un filtro que modifica a la sentencia SELECT y sirve para la extracción de registros cuando éstos se encuentran repetidos en una misma tabla, de tal forma que entregue únicamente los resultados sin repetir.

De igual manera, la opción COUNT es un modificador de la cláusula SELECT mediante el cual, podemos obtener la cantidad de registros que cumplen con las condiciones especificadas en la sentencia.

Este lenguaje tiene diversas cláusulas para la aplicación de las sentencias descritas anteriormente, estas cláusulas se encuentran en la tabla 2.4.

Cláusula	Uso
FROM	Especifica la (las) tabla(s) de las cuales se pretende la extracción de los registros. Las tablas se separan por comas y en la consulta se les puede agregar un "alias" para una mejor manipulación de los datos.
WHERE	Esta cláusula especifica las condiciones que el registro debe cumplir para cualquiera que sea el caso de las sentencias.
GROUP BY	Agrupa los registros resultados de una consulta de selección a una tabla en la base de datos.
HAVING	Esta cláusula fue añadida debido a que la cláusula WHERE no puede ser utilizada dentro de los grupos.
ORDER BY	Los registros resultantes de una selección de datos pueden ser ordenados de diferentes maneras y por diferentes campos.

TABLA 2.4 Cláusulas de DML. Fuente: Propia.

Dentro de una cláusula WHERE pueden utilizarse diferentes operadores tanto lógicos como de comparación para un mejor filtrado de la sentencia, evitando así la extracción, modificación o eliminación de registros no deseados.

Existen dos tipos de operadores permitidos sobre la cláusula WHERE. Por un lado se encuentran los operadores lógicos análogos a los operadores empleados en la lógica proposicional, y por otro lado, los operadores de comparación ya sean de cadenas, números, etc.

En la tabla 2.5, se muestra una descripción breve de los tres operadores lógicos permitidos en el lenguaje de manipulación de datos dentro de la cláusula WHERE.

Operador lógico	Uso
AND	El "y" lógico que permite especificar dos condiciones que deben cumplirse para la extracción/actualización de los datos en una tabla.
OR	El "o" lógico permite que la sentencia se cumpla siempre y cuando una u otra condición se cumpla.
NOT	La negación lógica permite la ejecución de una consulta siempre y cuando el campo al que hace referencia no cumpla con la condición especificada.

TABLA 2.5 Operadores lógicos. Fuente: Propia.

La ilustración 2.6, extraída del libro Beginning SQL Queries From Novice to Professional (Churcher, Beginning SQL Queries From Novice to Professional, 2007) muestra algunos ejemplos de uso de operadores lógicos así como un diagrama que facilita su compresión.

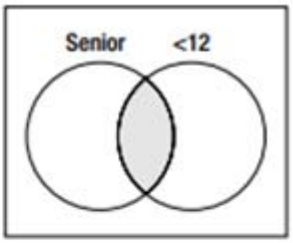
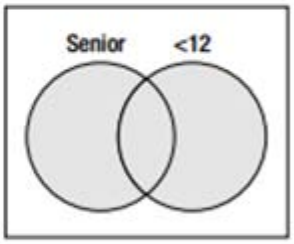
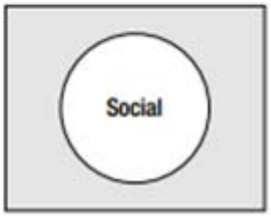
Expresión	Descripción de los datos recuperados	Diagrama de los datos recuperados
`MemberType = 'Senior' AND Handicap < 12`	Todos los miembros de tipo "Senior" con handicap menor que 12	
`MemberType = 'Senior' OR Handicap < 12`	La unión de a) los miembros que son de tipo "Senior y b) los miembros con handicap menor que 12	
`NOT MemberType = 'Social'`	Todos los miembros, excepto los que sean de tipo "Social"	

ILUSTRACIÓN 2.6 Operadores lógicos. Fuente: (CHURCHER, BEGINNING SQL QUERIES FROM NOVICE TO PROFESSIONAL, 2007)

Los operadores de comparación permitidos en el lenguaje de manipulación de datos, son mencionados por Churcher (Churcher, Beginning SQL Queries From novice to professional, 2008) y se describen de manera general en la tabla 2.6.

Operador de comparación	Uso
BETWEEN	Permite la ejecución de una sentencia mientras que el registro cumpla un intervalo de valores.
LIKE	Permite la comparación de patrones, donde pueden tomarse comodines cuando no se tiene del todo claro qué condición debe cumplirse o cuando existen registros con patrones similares.
IN	Permite especificar múltiples valores a la cláusula WHERE
“<”, “>”, “<>”, “=”, “<=”, “>=”	Estos operadores permiten la ejecución de sentencias con cláusula WHERE determinando cierto comportamiento de los valores sobre los que se desea realizar la consulta.

TABLA 2.6 Operadores de comparación. Fuente: (CHURCHER, BEGINNING SQL QUERIES FROM NOVICE TO PROFESSIONAL, 2008).

2.2.3.3. DCL LENGUAJE DE CONTROL DE DATOS

Es un lenguaje subconjunto de SQL que contiene una serie de sentencias para controlar la seguridad y los permisos en la base de datos. Permiten evitar la manipulación de datos sin autorización, previenen errores y daños a la información de la base de datos. La asignación de privilegios se realiza mediante el comando GRANT y para denegar privilegios se utiliza la palabra reservada REVOKE.

Entre muchas opciones que ofrecen los DBMS, en general se pueden otorgar y revocar privilegios sobre tablas y vistas tal como SELECT, DELETE, INSERT, UPDATE, REFERENCES, entre otros (Churcher, Beginning SQL Queries From Novice to Professional, 2007).

Las transacciones también pueden ser consideradas como parte del lenguaje de control de datos, ya que ello permite tener un mejor manejo sobre la modificación de los datos. Una transacción es un conjunto de sentencias SQL que se llevan a cabo sobre la base de datos de tal manera que todas esas sentencias se lleven a cabo exitosamente o ninguna de ellas se ejecute. Cuando una transacción es exitosa, se confirman los cambios a la base de datos mediante la palabra reservada COMMIT mientras que si en una transacción se presente un error, el comando ROLLBACK deja a la base de datos justo como antes de iniciar la transacción. Es decir, todas las instrucciones ejecutadas durante la transacción son canceladas y los datos vueltos a su estado original.

Ejemplo:

```
BEGIN
      INSERT into mitabla (campo1) values ('valor1');
      INSERT into mitabla (campo1) values ('valor2');
      INSERT into mitabla (campo1) values ('valor3');
      INSERT into mitabla (campo1) values ('Mivalor4');
      UPDATE table mitabla set campo1='valor5' where campo1 LIKE
'%valor%';
END
```

Dicha transacción inserta en la tabla “mitabla” cuatro registros y posteriormente actualiza el registro en el que el campo “campo1” tiene un valor (en este caso una cadena) que coincida con “valor” ignorando los caracteres anteriores y posteriores a dicha cadena.

Esto quiere decir, que los cuatro registros insertados en la tabla “mitabla” serán actualizados por el valor “valor5” ya que todos cumplen con la condición de que el valor en el campo “campo1” se parezca a “valor” (valor1, valor2, valor3, Mivalor4). Dicho esto, si estamos seguros de dicha actualización, la siguiente instrucción será COMMIT, para confirmar que los datos son correctos. Si por alguna razón, el valor introducido en la sentencia UPDATE es incorrecto o no queremos actualizar todos los registros mencionados, la siguiente instrucción deberá ser ROLLBACK. El único inconveniente, para este caso, sería que los cuatro datos insertados en la misma transacción no existirían en la base de datos luego de hacer el ROLLBACK descrito.

Por otro lado, mientras un usuario está modificando los registros en una tabla, otros usuarios no podrán modificarlos hasta que el primer usuario los “libere” confirmando o deshaciendo la transacción. A este efecto se le llama “bloqueo de sesiones”.

2.3. NOCIONES DE PROBABILIDAD Y ESTADÍSTICA

La Real Academia de la Lengua Española (Real Academia Española, 2001), se define a la estadística como:

Rama de la matemática que utiliza grandes conjuntos de datos numéricos para obtener inferencias basadas en el cálculo de probabilidades. Es decir, es una herramienta de suma importancia para la mayoría de (si no es que todas) las ingenierías, debido a que en general, un ingeniero debe procesar información ya sea para la toma de decisiones en cualquier proceso o incluso para el diseño de sistemas y herramientas que pretendan resolver algún problema teniendo como base un soporte sólido en cuanto al análisis de la situación y comparativas a través de diferentes variables que puedan afectar el entorno tales como el tiempo.

La palabra estadística, proviene del latín *statisticus* (Dechile.net, 2001) formada por status (estado) e icus (relativo a), lo que nos indica que en sus orígenes, fue empleada para proporcionar información a los gobernantes sobre asuntos del estado (como nación).

Bañuelos define población como el conjunto de todos los datos posibles en un experimento (Bañuelo Saucedo & Manzanarez Gómez, 2012). Dentro de este conjunto de datos, se encuentra un subconjunto determinando muestra. Ejemplificando dicha definición, podemos referir a la ilustración 2.7:

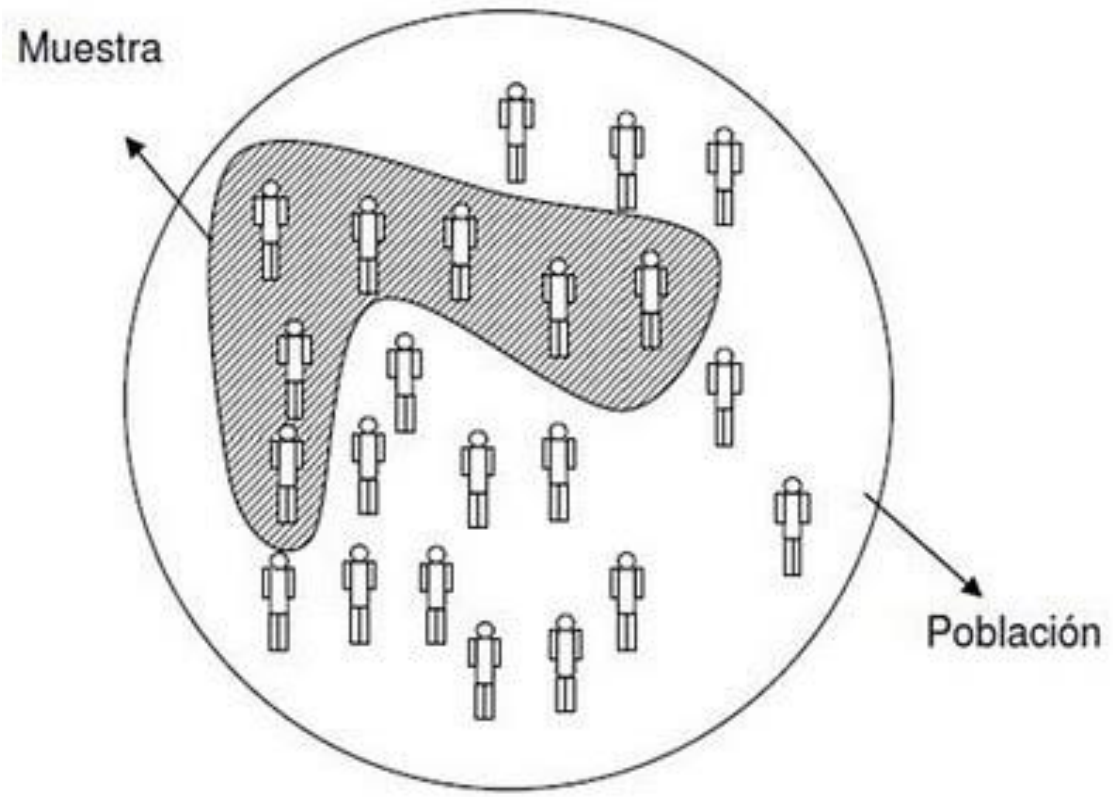


ILUSTRACIÓN 2.7 Población y muestra. Fuente: (WEB INCERTIDUMBRE, S.F.)

La muestra debe ser una parte sustancial y representativa de la población para poder tener un modelo más aproximado a la realidad, por ende, debe ser una muestra aleatoria, es decir, que cualquiera de los elementos del conjunto población pudo haber sido seleccionado en el proceso.

Una manera de clasificar a la estadística puede ser por el tipo de aplicación, encontrando así la estadística descriptiva y la estadística inferencial.

Esta tesis emplea la estadística descriptiva, que es la parte de la estadística que se focaliza en la organización y presentación de los datos para ser analizados e interpretados, mientras que la estadística inferencial es aquella con la que se puede obtener, mediante probabilidad, el grado de certeza de alguna conclusión sobre la muestra elegida.

2.3.1. DISTRIBUCIÓN DE FRECUENCIAS

"Es una técnica de agrupación usada en estadística cuando se tiene un conjunto muy grande de datos, de tal forma que el análisis posterior: gráficas y parámetros numéricos; se pueda realizar de forma más rápida" (Bañuelo Saucedo & Manzanarez Gómez, 2012). De esta manera, al tener agrupados (y ordenados) los datos podremos convertirlos en información que utilizaremos para realizar análisis, toma de decisiones y, en caso de la estadística inferencial, predicciones y conclusiones de ellos.

Al conjunto de datos ordenados y previamente supervisados, se le llama información, la información es un proceso mental de interpretación de los datos para que estos tengan sentido, ya que de manera aislada, o sin contexto, carecen de él.

En gran parte de esta tesis, se empleará el término frecuencia, siendo este concepto un importante elemento para la extracción automatizada de palabras clave. Suponiendo un conjunto población A, la frecuencia es el número de apariciones de un mismo elemento en el conjunto población A.

Otro grupo de conceptos que es sumamente necesario definir es el comprendido por las "Medidas de tendencia central". Los elementos más conocidos son tres: Media, mediana y moda.

La media aritmética, es mejor conocida simplemente como media y se define como el promedio de un conjunto de valores. Denotada por $\bar{x}$ y definido como:

$$\bar{x} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} x_i \; ; \; \textit{Datos no agrupados} \\ \frac{1}{n}\sum_{i=1}^{m} x_i f_i \; ; \; \textit{Datos agrupados} \end{cases}$$

Aunque existen otras definiciones como la media geométrica y la media armónica, para fines prácticos sólo se hará referencia a la media aritmética.

En un conjunto de datos necesariamente ordenados, se le denomina mediana al elemento o elementos que divide al conjunto en dos conjuntos de igual tamaño y se denota con el símbolo $\tilde{x}$. De una manera más simple, entendamos a la mediana como el valor que se encuentra a la mitad de los datos ordenados. Si tenemos **n** valores ordenados, la mediana se encontrará en el elemento **n**/2. Dado que el algoritmo desarrollado para el sistema de la presente tesis es un algoritmo estadístico, se utilizan estos conceptos más adelante para los cálculos de frecuencia coocurrencia y demás. Se explican a detalle en el capítulo 4.

Capítulo 3. Sistemas de Recuperación de Información (IRS[21])

INTRODUCCIÓN

Hoy día nos resultan familiares los términos informales "*googlear*", "buscar en internet" o "investigar en la web". Sin embargo, pocas veces nos ponemos a pensar cómo es que los grandes buscadores (google, bing, yahoo, etc.) logran obtener los resultados que más podrían resultarnos interesantes o realmente útiles. En este capítulo se explicará la naturaleza y funcionalidad de un sistema de recuperación de información (IRS por sus siglas en inglés) y se responderá a preguntas como "¿Qué es un sistema de recuperación de información?", "¿Cuáles son sus características?" y más importante aún "¿Qué aplicaciones tienen en bibliotecas de la UNAM?"

[21] IRS: Information Retrieval System.

3.1. ¿QUÉ SON LOS SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN?

Para responder, primero debemos entender que en realidad un buscador (de internet) realiza mediante un algoritmo[22] generalmente privado, comparación entre lo que el usuario introduce en el campo de búsqueda y la información que ya contiene en sus bases de datos; y aunque esto parezca trivial y simple, es digamos, algo más que complejo, pues se requieren algunos pasos previos para lograr tener un resultado de búsqueda satisfactorio.

Existen millones de páginas web con información que puede resultar útil y particularmente Google se basa en "más de 200 pistas que hacen que sea posible adivinar lo que en realidad se podría estar buscando" (Google, 2011).

"La recuperación de información es la disciplina encargada de la representación, almacenamiento y la organización de la información que responda a las necesidades de un usuario" (Epifanio Tula & Medeot Matías, 2008).

Entendiendo esto, se puede definir a un sistema de recuperación de información como el software desarrollado con todos los algoritmos necesarios para almacenar, ordenar, indexar, buscar y recuperar información.

[22] Algoritmos: Google define a los algoritmos como programas informáticos que buscan pistas para devolver exactamente lo que quieres. (Google, 2011)

3.2. CARACTERÍSTICAS

En el capítulo I Antecedentes, se habló, en parte, del tratamiento de los documentos para la transformación y transferencia de información. Es por eso que se requiere identificar y seleccionar los elementos esenciales de dichos documentos para su extracción y con ello, facilitar el proceso de ordenamiento, indización y almacenamiento de los documentos.

Por un lado, en la Universidad de Chile (Biblioteca Central Facultad de Derecho) se define como recuperación de información al proceso que es sometido un conjunto de documentos previamente tratados para obtener, de ese conjunto, sólo aquellos documentos que cumplan con ciertos requerimientos de búsqueda.

Mientras que por otro lado, Babb establece en una patente (Londres, Inglaterra Patente nº 3964029, 1976) que un sistema de recuperación de información permite, mediante consultas a la información almacenada en la base de datos, recuperar la información de dichos documentos. La recuperación se lleva a cabo procesando cada registro del documento y comparándolo con los registros en la base de datos y produciendo una bandera cada que existe una correspondencia.

Las consultas, más allá de ser tratadas como queries de SQL en un lenguaje técnico de base de datos, en la Universidad de Chile (Biblioteca Central Facultad de Derecho) se define que son conocidas como sentencias formales de expresión de necesidades de información.

La ilustración 3.1 muestra de manera general el funcionamiento de los sistemas de recuperación de información.

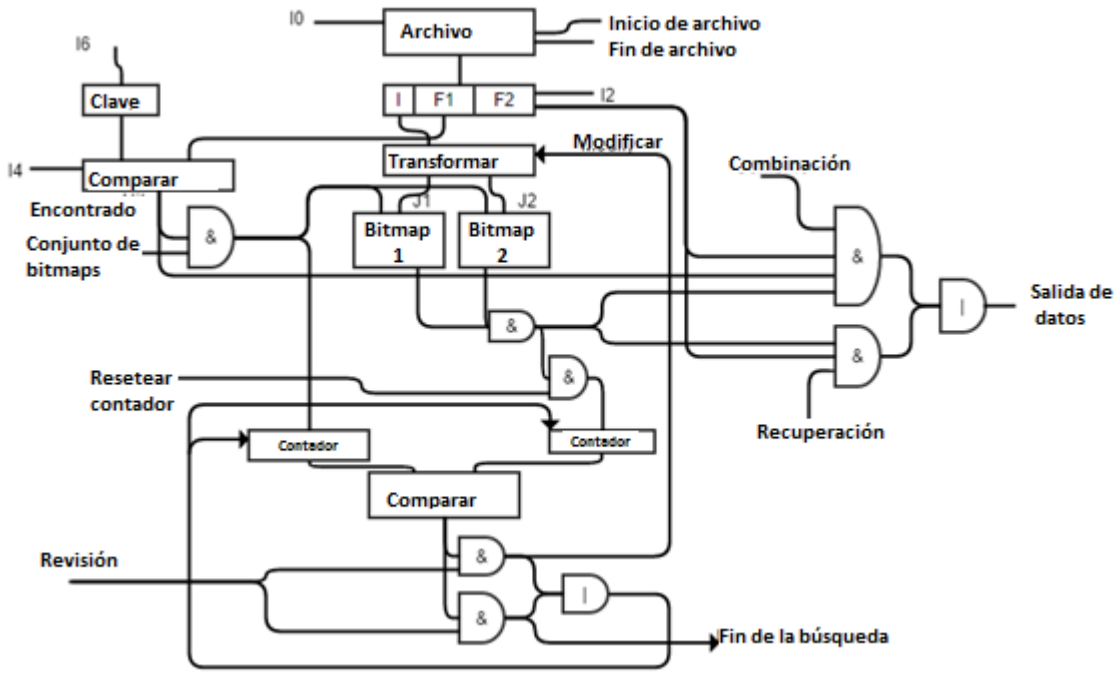


ILUSTRACIÓN 3.1 Information Retrieval System. Fuente: (LONDRES, INGLATERRA PATENTE Nº 3964029, 1976)

En la ilustración 3.1 se especifican los procesos de comparación entre las claves de entrada de la búsqueda por parte del usuario con la información procesada y transformada a partir de los archivos. Recordando que esta patente data de 1976, se debe especificar que los descriptores del documento ya se encuentran almacenados en bases de datos.

Existen tres diferentes tipos de consulta para los IRS según una publicación de la Universidad de Chile (Biblioteca Central Facultad de Derecho): por palabra única, por frase y booleana, las cuales se describen a continuación.

La primera resulta cuando el usuario solicita información puntual a buscar la cual puede aparecer en los descriptores del documento. Es una de las más improbables a suceder debido a que los usuarios generalmente no conocen las palabras con las

que un texto ha sido indizado y por ende, su búsqueda puede o no obtener la información deseada.

El segundo tipo de búsqueda corresponde a las más usuales, debido a que el usuario generalmente realiza una búsqueda por un conjunto de palabras que estructuradas forman una idea, siendo estas consultas las más trabajadas por los IRS y que requieren más procesamiento para hacer coincidir las palabras de entrada (eliminando palabras carentes de peso léxico) con las palabras que se tienen como descriptores en la base de datos.

El tercer conjunto de búsquedas podría resultar el más sencillo para los sistemas de búsqueda debido a que por ser booleana, el único condicionante resulta ser que el resultado de la búsqueda contenga o no contenga, según la preferencia del usuario, la o las palabras filtro.

3.3. APLICACIÓN EN BIBLIOTECAS UNAM

No podemos hablar del sistema bibliotecario de la UNAM sin hacer una forzosa mención a la Biblioteca Central. Ese edificio que, una vez decidido como punto de convergencia de las áreas del conocimiento universitario, se ha caracterizado por la innovación, como en la década de los 60s, cuando se creía que la reprografía sería el camino a seguir para las bibliotecas gracias a sus múltiples ventajas y ahorros de mantenimiento y traslación de obras. Respondía bien a las necesidades de la época aunque por aquellos tiempos no se contemplaba la aparición e importancia de las computadoras en el ámbito bibliotecario.

Tiempo después, en el año de 1973, se inició la puesta en operación de Librunam, un instrumento para captura y consulta de información gracias al trabajo de la Dirección General de Bibliotecas.

Luego, en 1985 se fortaleció el acervo capturado en el sistema LIBRUNAM, con lo que se podría realizar búsquedas bibliográficas en la colección completa.

También se retomó el proyecto TESIUNAM, encargado de coleccionar las tesis de los egresados de cualquier nivel de estudios en la UNAM.

Ya en la década de los 90, se da inicio a la utilización de catálogos en CD-ROM, los cuales permitían realizar consulta a las bases de datos de manera local, sobreponiéndose a los retos que en aquel momento representaban las telecomunicaciones. (Rodríguez Gallardo, Por qué una Biblioteca Central, 2001)

Posteriormente llegó el momento de la automatización de las bibliotecas, para ello se aprobó el sistema piloto de TINLIB, un sistema que se estaba probando en la biblioteca de la entonces Dirección General de Servicios de Cómputo Académico, DGSCA[23] gracias a un estudio de 10 sistemas de automatización de bibliotecas, entre los que se encontraban Dynix, Dynix Marquis, NOTIS, INNOPAC, Information Navigator (TINLIB), Zebra 2000, Geac, SIRSI, Marcop y STAR. (Gama Ramírez, s.f.)

El sistema TINLIB fue desechado por la Dirección General de Bibliotecas y en 1997 se inicia la implementación del sistema ALEPH.

[23] Años después se cambió el nombre a DGTIC: Dirección General de Cómputo y Tecnologías de Información y Comunicación.

El sistema Aleph (Automated Library Expandable Program) es un software integral que nació de la necesidad de automatizar las bibliotecas universitarias. (Acosta Mata, 2004)

El sistema ha sido desarrollado por la empresa ExLibris en los años 80 en la Universidad Hebrea de Jerusalén. Siendo un sistema que *maneja todos los aspectos de los servicios bibliotecarios tanto para el personal como para los usuarios* (Wiki ALEPH, 2014).

Como detalles del sistema, encontramos la particularidad de correr sobre sistemas UNIX, soporte de más de mil usuarios concurrentes y la realización de transacciones vía correo electrónico.

Principales características (Wiki ALEPH, 2014):

- Arquitectura cliente-servidor.
- Utiliza base de datos Oracle (Soporte RDBMS).
- Soporte completo de Unicode.
- Compatibilidad Z39.50
- Soporte MARC (USMARC y UNIMARC).
- Diseño multiplataforma.
- Cliente www compatible con Netscape. Windows/UNIX.

Debido a la necesidad de administrar el material bibliográfico, se implementó el sistema Aleph.

Actualmente, en las bibliotecas de la UNAM se cuenta con la versión Aleph 500 la cual se comenzó a implementar a partir de 2005. Anterior a ella, desde 1996, se contaba con la versión Aleph 300 por lo que ha sido necesaria la adquisición de equipo con mayor capacidad y transferir datos de aleph 300 a aleph 500 (Dirección General de Bibliotecas, 2008).

El sistema bibliotecario de la UNAM se subdivide en cinco subsistemas más (Universidad Nacional Autónoma de México, Dirección General de Bibliotecas, 2013):

- Bachillerato.
- Licenciatura y posgrado.
- Investigación científica.
- Investigación en humanidades.
- Extensión y administración universitaria.

Los porcentajes se describen en la ilustración 3.2

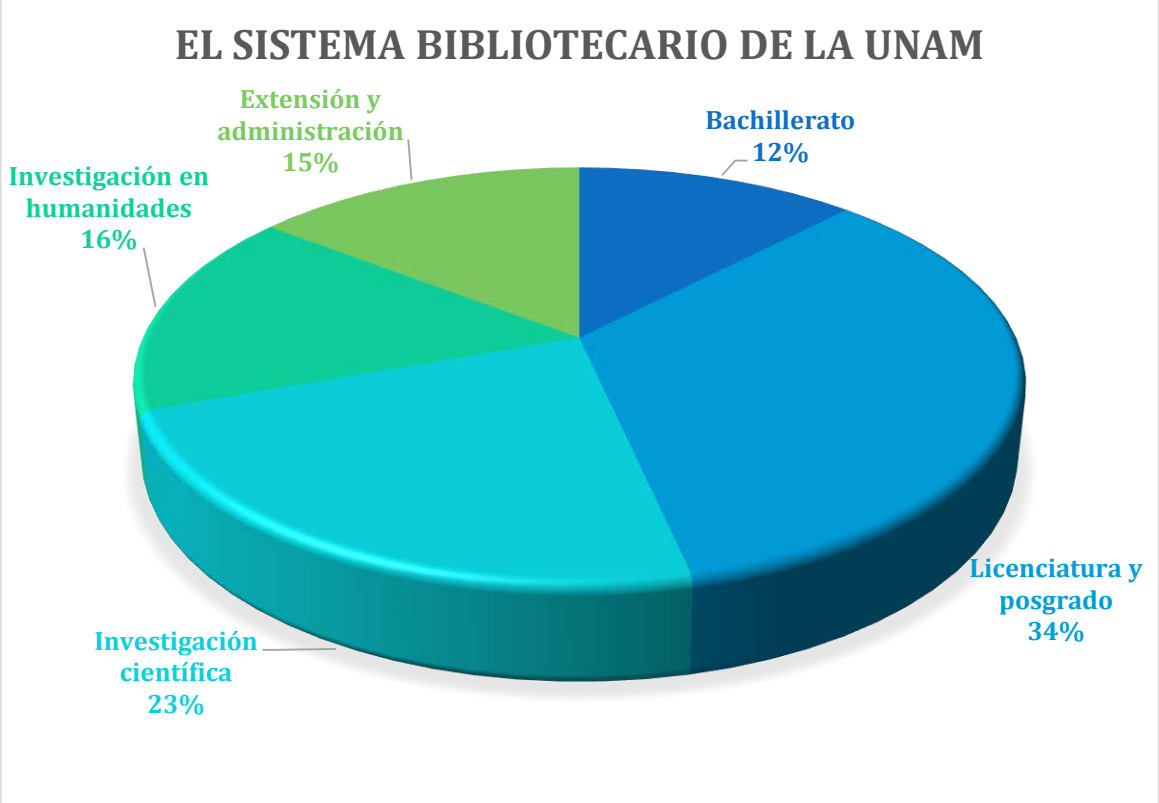


ILUSTRACIÓN 3.2 Subsistemas del sistema bibliotecario UNAM. Fuente: (DIRECCIÓN GENERAL DE BIBLIOTECAS, 2008)

En la ilustración 3.3 podemos observar que entre Investigación científica y licenciatura y posgrado se ocupa más de la mitad de instalaciones de módulos de Aleph 500 en las entidades que conforman al sistema bibliotecario de la UNAM (Dirección General de Bibliotecas, 2008).

ILUSTRACIÓN 3.3 Sistema bibliotecario de la UNAM-Bibliotecas con Aleph 500. Fuente: (DIRECCIÓN GENERAL DE BIBLIOTECAS, 2008)

Existen diversos módulos del sistema aleph, entre los que encontramos OPAC[24], catalogación, publicaciones periódicas, catálogos de autoridad, circulación, adquisiciones, utilerías, préstamo interbibliotecario y mantenimiento de claves.

Los módulos de Aleph 500 (Acosta Mata, 2004):

- OPAC permite el acceso a las colecciones públicas, mediante búsquedas ya sea por autor, título, encabezamientos de materia, clasificación, serie, etc.
- CC (catalogación). Los registros bibliográficos para el sistema librunam deben ser en formato MARC[25], ofrece la actualización de base de datos para posterior consulta mediante OPAC.
- Publicaciones periódicas: en la UNAM, se llevan a cabo la tramitación, el registro, compras y reclamaciones de revistas adquiridas, así como la base de datos SERIUNAM.
- Catálogos de autoridades: El proceso de catalogación llevando un control extensivo de autoridades, permite un mejor flujo de trabajo y acelera el proceso de catalogación.
- Circulación: Es el módulo que permite a las bibliotecas de la UNAM, conocer el estatus de un ejemplar, si ha sido prestado o devuelto y en caso de estar prestado, qué usuario lo solicitó.
- Adquisiciones: Cada biblioteca puede llevar de manera autónoma las adquisiciones que realice dependiendo sus propias políticas.
- Mantenimiento de claves: Brevemente mencionamos que para acceder a cada módulo, se debe contar con contraseñas que resguardan los niveles de seguridad.

[24] Online Public Access Catalogue

[25] Machine Readable Cataloging

En la ilustración 3.4 se muestra un comparativo entre los módulos del sistema Aleph 500 y las entidades que componen el sistema bibliotecario de la UNAM.

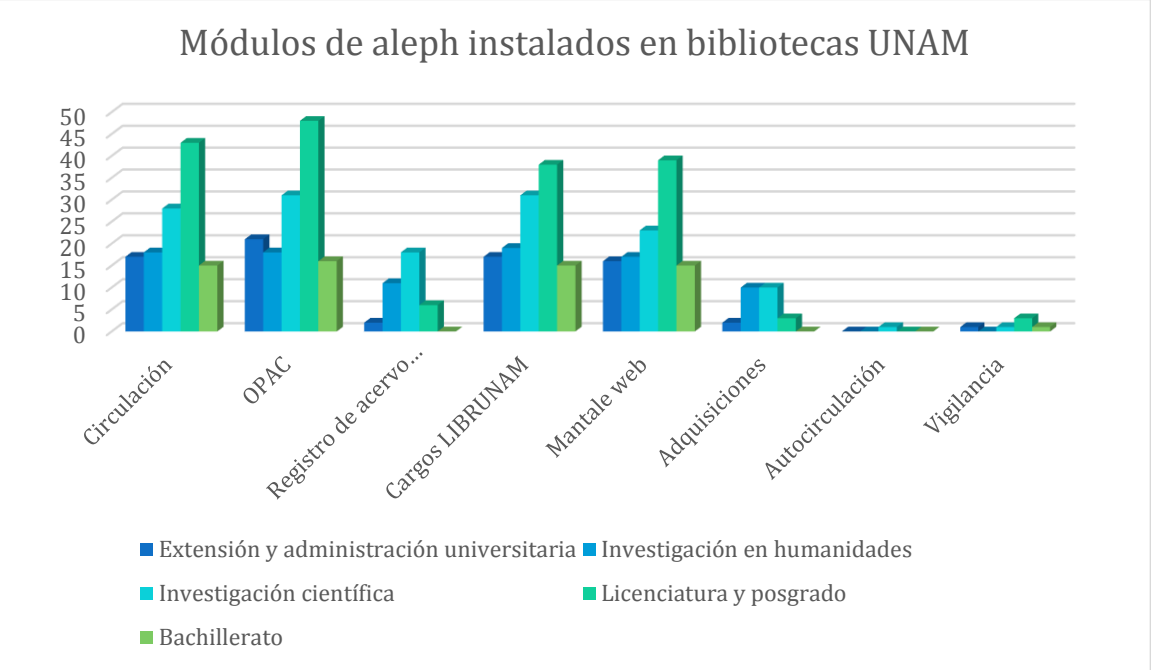


ILUSTRACIÓN 3.4 Módulos de Aleph 500 en entidades del sistema bibliotecario UNAM. Fuente: (ACOSTA MATA, 2004)

En este caso, el módulo OPAC del sistema Aleph sería nuestro IRS, ya que mediante ese módulo se pueden realizar diferentes búsquedas y se pretende tener descriptores más acertados en durante el proceso de indización para conseguir la optimización de búsqueda planteada en esta tesis.

Capítulo 4. Algoritmos de procesamiento inteligente de texto

INTRODUCCIÓN

La comunicación verbal podría parecernos *a primera escucha*[26] un sinfín de palabras aleatorias que viajan por el aire hasta nuestros oídos y entonces entendidas por el oído. El lenguaje no es tan trivial como parece. Si (realmente) prestamos atención a lo que decimos, podemos notar que existen palabras que solemos utilizar más que otras. En este capítulo, se hará una remembranza de algunos de los algoritmos existentes para el procesamiento inteligente de texto. Siendo la detección de palabras representativas el tópico que nos interesa, el estudio de los algoritmos se centra en dicha peculiaridad del procesamiento inteligente de texto.

[26] Haciendo referencia a la frase común "a primera vista" adecuándolo por no tratarse de un elemento visual.

4.1. PROCESAMIENTO INTELIGENTE DE TEXTO

Debido al crecimiento exponencial de información y el correspondiente almacenamiento en formatos digitales, se ha incrementado grandemente la demanda de archivadores robustos donde almacenar dichos volúmenes de texto en formato digital. (Rauch & Rauber, 2004).

Es esta misma acumulación de documentos digitales por lo que se ha requerido la generación de tecnologías que permitan su procesamiento de una manera un tanto diferente y especializada. Primordialmente debido al requerimiento de información completa para la toma de decisiones, a esta tecnología se le ha acuñado el término Wistech (wisdom technology). El cual describe la ecuación 4.1 (Marciniak, 2009).

ECUACIÓN 4.1 SABIDURÍA = CONOCIMIENTO + JUICIO ADAPTATIVO + INTERACCIONES.

El procesamiento de grandes cantidades de textos es indispensable hoy en día para una toma de decisiones basada en información.

El procesamiento inteligente de textos se basa en el procesamiento del lenguaje natural (NLP, por sus siglas en inglés) el cual consiste en aplicar algoritmos diseñados para la detección de patrones sobre un grupo determinado de textos, generalmente en grandes cantidades, para obtener información inmersa en el cuerpo de dichos documentos de una manera eficaz, que de lo contrario, debería revisarse a conciencia, y le tomaría años a cualquier persona lograr la abstracción de la información implícita que resulta de la combinación de conocimiento en esas cantidades de texto.

El resultado del procesamiento inteligente de textos debe ser una representación compacta del contenido de los documentos (Berry & Kogan, 2010). A esta representación, se le puede considerar como palabras clave (keywords), las cuales

describen *grosso modo*[27], la información contenida en el cuerpo del documento. Son palabras que *describen* el contenido en el texto.

4.2. LEY DE ZIPF

En cualquier escrito, podemos destacar la utilización de cierto conjunto de palabras de una manera notoriamente más frecuente. Este comportamiento ha sido estudiado desde hace más de cincuenta años por George Kingsley Zipf, quien realizó investigaciones acerca de la repetición de las palabras utilizadas en el lenguaje analizando la aparición de éstas en un texto (frecuencia de las palabras).

Zipf, siendo lingüista, notó que las palabras tienen un comportamiento singular en la mayoría de los textos que analizó. Al ordenar las palabras por frecuencia en un texto, llamó rango a la posición que ocupa la palabra en dicho listado (Braun, 1996).

En términos técnicos, la ley de Zipf hace referencia a la distribución de palabras en un lenguaje determinado, marcando una tendencia aproximada de aparición de palabras, es decir: Al tomar la frecuencia de aparición de las palabras de un texto de un campo de conocimiento específico, y ordenar las frecuencias de todas las palabras por mayor número de aparición al menor número de aparición, el rango es inversamente proporcional a la frecuencia (Parra, 2010):

ECUACIÓN 4.2 F(N)=1/R

Donde:

- F es la frecuencia.
- n es la palabra en cuestión.
- r es el rango de la palabra.

[27] De una manera superficial, somera.

Antes de proseguir con el descubrimiento de Zipf, se debe aclarar la ecuación 4.2. Debemos entender que con frecuencias muy altas, las palabras estarán en los primeros lugares del listado ordenado, es decir, el rango será pequeño. Mientras que una palabra con una frecuencia baja, aparecerá en la parte baja del listado (donde son las posiciones más altas).

A simple vista, este descubrimiento no podría parecer relevante, pero al comparar un listado de palabras frecuentes de un texto con el listado de palabras frecuentes en otro, puede observarse una tendencia de las palabras a aparecer siempre en la misma zona del listado.

Dicho de otra forma, una palabra de las categorías gramaticales artículo y preposición, para el español, o la palabra "*the*" para el idioma Inglés siempre resultan estar en los primeros lugares del rango (Parra, 2010) debido a que son palabras que en general, todas las personas utilizan ya sea por necesidad o por comodidad.

Para entender mejor este algoritmo, se muestra un ejercicio con un fragmento del texto "El desarrollo del software" (Vacas Sáez, 1992).

"el gran desafío con que se encuentra la gestión de proyectos de software consiste precisamente en limitar los productos que se desarrollan en esos proyectos a unos niveles de complejidad aceptables y manejables."

El texto contiene 33 palabras. 27 palabras diferentes, las cuales, se ordenan de mayor a menor frecuencia y en la tabla 4.1 podemos ver las primeras diez

Rango	Palabra	Frecuencia	Densidad
1	DE	3	0.090909090909091
2	EN	2	0.060606060606061
3	QUE	2	0.060606060606061
4	SE	2	0.060606060606061
5	PROYECTOS	2	0.060606060606061
6	EL	1	0.03030303030303
7	PRODUCTOS	1	0.03030303030303
8	DESARROLLAN	1	0.03030303030303
9	ESOS	1	0.03030303030303
10	A	1	0.03030303030303

TABLA 4.1 Ordenamiento de palabras por frecuencia. Fuente: Propia.

La columna “Densidad” en la tabla 4.1 es calculada mediante una simple división de la frecuencia entre el total de palabras, algo que podemos interpretar como la probabilidad de que la palabra aparezca en el texto.

Algunos autores mencionan que otra manera de calcular la ley de Zipf es contar las veces que aparece una palabra y dividirla entre el número total de palabras (Parra, 2010). En este caso, se ha optado por nombrarle a esa división como *densidad* de la palabra en el documento, debido a que es la cantidad de palabras iguales en un conjunto de palabras y también podríamos decir que representa la probabilidad de aparición de la palabra en el total del documento.

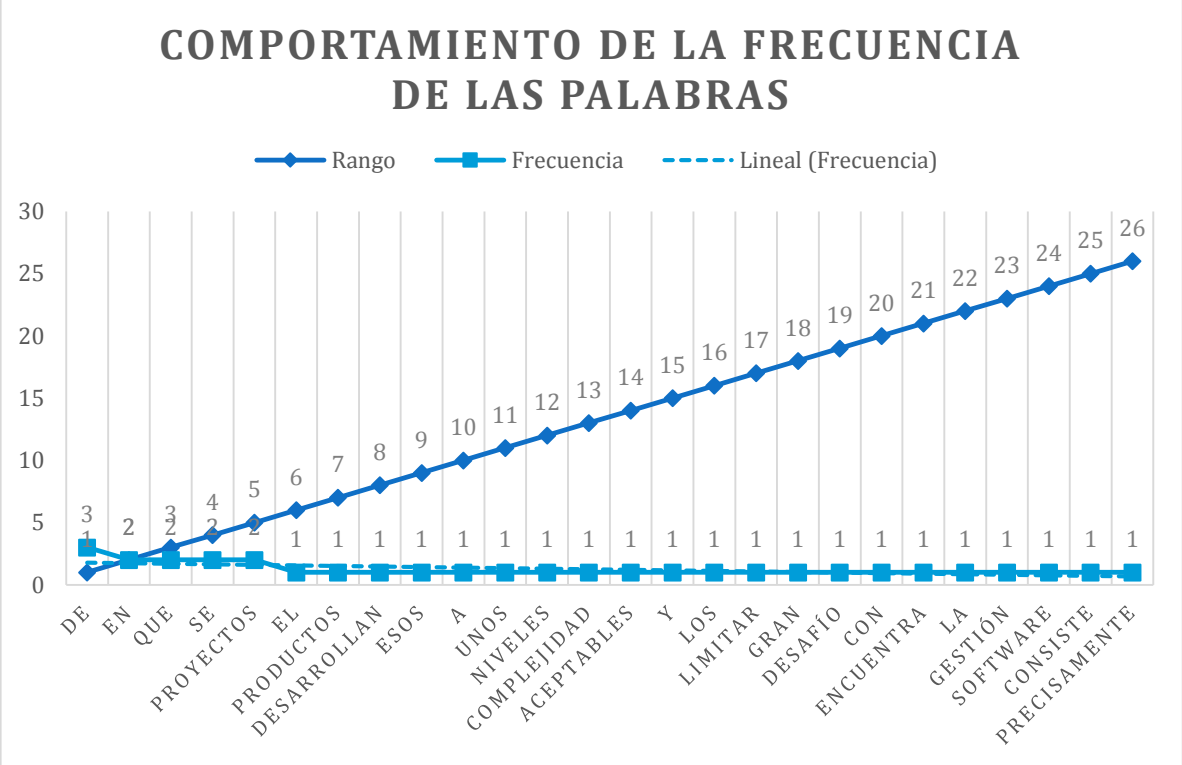


ILUSTRACIÓN 4.1 Comportamiento de frecuencias. Fuente: Propia.

De la ilustración 4.1, se puede observar el comportamiento mencionado de la relación existente entre el rango y la frecuencia de las palabras, la cual indica que son inversamente proporcionales. Mientras el rango incrementa, la frecuencia disminuye. También podemos comprobar el cumplimento de la aseveración que

dicta que en las frecuencias más altas se encuentran los artículos y preposiciones, siendo éstas las palabras de uso más común y aquellas que tienen mayor probabilidad de aparecer en un texto, tal como se muestra en la ilustración 4.2:

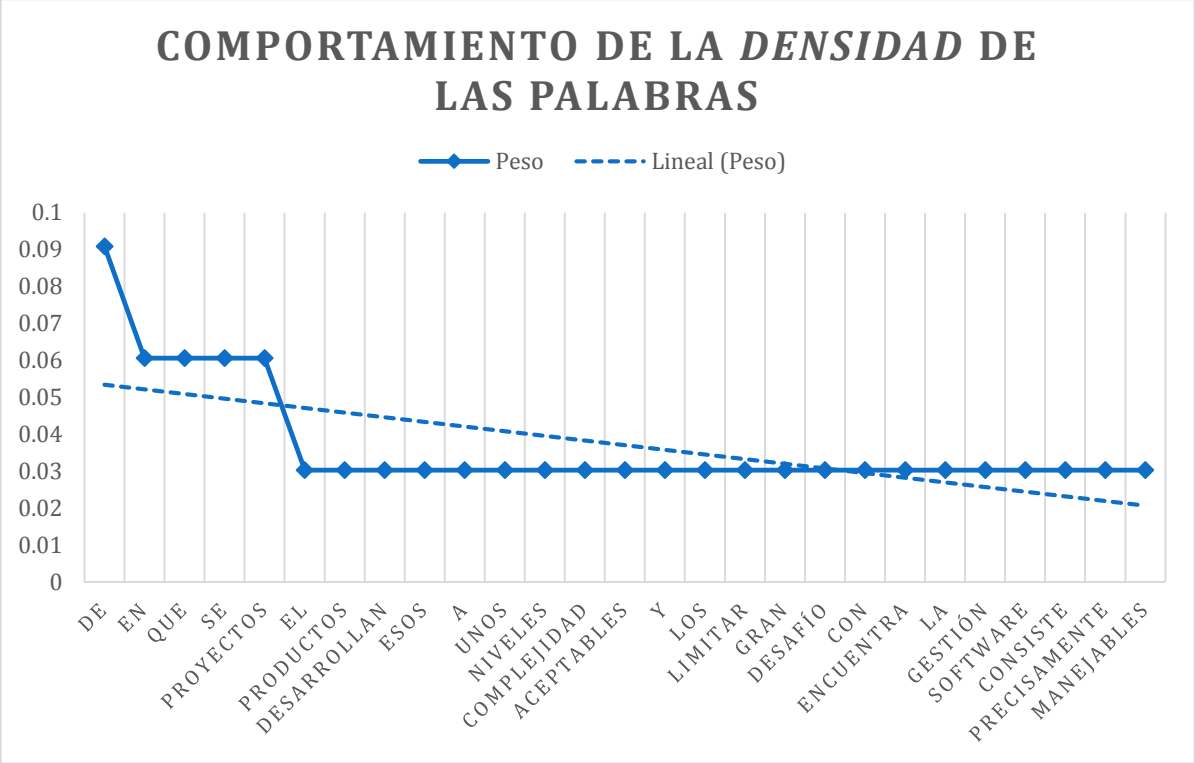


ILUSTRACIÓN 4.2 Comportamiento de probabilidades. Fuente: Propia.

En ambas ilustraciones podemos denotar la tendencia que sigue la aparición de las palabras. Cabe mencionar que en la ilustración 4.2, la escala es menor a 1 debido a que se trata de lo que, como se mencionó anteriormente se puede equiparar a la probabilidad de aparición de la palabra en el documento.

Por otro lado, las palabras menos frecuentes son aquellas que no son empleadas de manera regular y que por ende, tienen un rango alto y una densidad demasiado pequeña.

4.3. PUNTO DE TRANSICIÓN GOFFMAN

En el primer apartado de este capítulo se habló acerca del comportamiento y tendencia de frecuencia de aparición de las palabras en un documento escrito, así como la importancia de ordenarlas por rango y número de aparición con la intención de discernir entre las palabras de uso cotidiano y aquellas que son rebuscadas o de uso escaso.

Empíricamente pudo observarse el hecho de que formular una ley para el tratado de las palabras con baja frecuencia (alto rango) sería inadecuada para la descripción del texto completo (Boyce & Lockard, 1975). Este hueco en la utilidad de la ley de Zipf provocó investigaciones por parte de investigadores como DONOHUE, J. C., BAUGHMAN, J. C., MIRANDA, H. Y. LEE PAO., BOYCE, B. R. y William Goffman (Boyce & Lockard, 1975), éste último, investigador, profesor, decano y emérito de la universidad de Case Western Reserve en Cleveland, Ohio (Harmon, 2007) sugirió un procedimiento automatizado para la obtención de palabras descriptoras de un documento.

Goffman entendió que debía existir un punto en el que las palabras de frecuencias bajas dejaran de serlo sin llegar a las palabras de frecuencias altas que, como se mencionó en el apartado correspondiente, son de las categorías gramaticales artículo o preposición. Palabras con poco aporte descriptivo del texto.

Si bien la ley de Zipf permite conocer el comportamiento de las palabras más frecuentemente utilizadas en algún lenguaje, únicamente permite hacerse a la idea empírica (y por observación de gráficas) de las palabras más comunes y las menos utilizadas. Es por eso que el estudio de Goffman es más matemático, un concepto sobre la distribución de la frecuencia de palabras en un texto determinado donde "terminan las frecuencias bajas y comienzan las frecuencias altas".

El estudio, llamado “punto de transición de Goffman” establece que una vez determinado el punto de transición de la distribución de frecuencias, se puede

obtener un conjunto de palabras con contenido altamente descriptivas del texto en cuestión.

El punto de transición puede obtenerse a partir de la ecuación 4.3 (Boyce & Lockard, 1975):

ECUACIÓN 4.3 $I_n/I_1 = 2/n(n + 1)$

Donde I_n es el número de palabras en el texto con una frecuencia "*n*", I_1 es el número de palabras con frecuencia 1, pero escrita de esa manera es incompatible con la primera ley de Zipf (Boyce & Lockard, 1975).

Podríamos creer que el vecindario de palabras descriptoras de contenido comienza cuando I_n=1 (Boyce & Lockard, 1975), es decir que al recorrer el listado de frecuencias de palabras y llegar a aquella zona donde la frecuencia es la mínima posible (una única aparición de la palabra en el texto) entonces, despejando de la ecuación 4.3 y siendo I_n=1, podemos comenzar a buscar la frecuencia “n” a partir de la cual se encuentra el vecindario de palabras que en realidad describen el contenido del texto.

ECUACIÓN 4.4 $2I_1 = n^2 + n$

ECUACIÓN 4.5 $n^2 + n - 2I1 = 0$

Resolviendo la ecuación cuadrática 4.5 [28] para la frecuencia n en la que ocurre la transición de palabras menos frecuentes para convertirse en más frecuentes:

[28] Utilizando la fórmula general para ecuaciones de segundo grado

ECUACIÓN 4.6 $\boldsymbol{n = \frac{-1 \pm \sqrt{1+8I_1}}{2}}$

Al encontrar la frecuencia que marca el punto de transición, se toma un vecindario de frecuencias superiores y frecuencias inferiores para determinar el conjunto más probable a ser candidato del total de palabras en el texto.

Para entender de una mejor manera el algoritmo será aplicado al ejercicio realizado en el apartado sobre la ley de Zipf.

Del ejercicio mencionado, podemos rescatar algunos valores:

- Palabras totales: 33
- Palabras con una aparición: 22
- Frecuencia que marca el punto de transición:

ECUACIÓN 4.7 $\boldsymbol{n = \frac{-1 \pm \sqrt{1+8(22)}}{2}}$

$$\boldsymbol{n = \frac{-1 \pm \sqrt{1+176}}{2}}$$

$$\boldsymbol{n = \frac{-1 \pm \sqrt{177}}{2}}$$

$$\boldsymbol{n = \frac{-1 \pm 13.30}{2}}$$

ECUACIÓN 4.8 $\boldsymbol{n_1}$[29]**=6.15**

[29] Una frecuencia es entero, por lo que se redondea al próximo valor entero positivo más cercano: 6.

Encontrando el punto de transición con valor de 6, se muestran las palabras que tienen una frecuencia del vecindario de 6, tomando 10 elementos con una frecuencia mayor y 10 elementos con una frecuencia menor. En este caso, no existe una frecuencia mayor al punto de transición, por lo que la tabla 4.2 se conforma únicamente por las 10 primeras palabras.

Rango	Palabra	Frecuencia	Densidad
1	DE	3	0.090909090909091
2	EN	2	0.060606060606061
3	QUE	2	0.060606060606061
4	SE	2	0.060606060606061
5	PROYECTOS	2	0.060606060606061
6	EL	1	0.03030303030303
7	PRODUCTOS	1	0.03030303030303
8	DESARROLLAN	1	0.03030303030303
9	ESOS	1	0.03030303030303
10	A	1	0.03030303030303

TABLA 4.2 Vecindario de palabras en punto de transición. Fuente: Propia.

Como puede observarse en la tabla 4.2, las palabras que más aparecen son artículos, preposiciones y conjunciones, es por ello que se debe modificar el algoritmo añadiendo un paso al inicio del proceso de la ley de Zipf. El paso mencionado, consiste en filtrar aquellas palabras que se crean irrelevantes, es decir que no aporten gran valor a la descripción del documento.

El mismo fragmento de texto fue procesado con el algoritmo modificado y arrojó los siguientes resultados:

- Total de palabras (ya filtradas): 11
- Palabras con una aparición: 9
- Frecuencia que marca el punto de transición:

ECUACIÓN 4.9 $\boldsymbol{n = \frac{-1 \pm \sqrt{1+8(9)}}{2}}$

$$\boldsymbol{n = \frac{-1 \pm \sqrt{1+72}}{2}}$$

$$\boldsymbol{n = \frac{-1 \pm \sqrt{73}}{2}}$$

$$\boldsymbol{n = \frac{-1 \pm 8.54}{2}}$$

ECUACIÓN 4.10 $\boldsymbol{n_1 = 4.77}$

De las palabras resultantes, seleccionamos un vecindario relativamente pequeño, como muestra se toman 8 registros con una frecuencia superior y 8 registros con un rango superior (menor frecuencia) al punto de transición, aunque, en este ejemplo, son únicamente palabras cuya frecuencia se encuentra por debajo del punto de transición. Danto como resultado la tabla 4.3:

Palabra	Frecuencia
PROYECTOS	2
Desafío	1
NIVELES	1
DESARROLLAN	1
LIMITAR	1
CONSISTE	1
SOFTWARE	1
Gestión	1

TABLA 4.3 Punto de transición con filtrado. Fuente: Propia.

Podemos observar un nuevo comportamiento el cual selecciona palabras que definen en cierta manera el contenido del texto seleccionado para el procesamiento.

Este algoritmo es eficiente y puntual para términos de una palabra aunque dependa del conjunto de palabras *irrelevantes* para el área para tener mejor exactitud aunque sin la precisión deseada para el tipo de consulta más frecuente mencionado en el capítulo III: la consulta por palabra única.

4.4. C-VALUE

El algoritmo en turno a analizar es un algoritmo estudiado en el departamento de Ingeniería Lingüística de la Universidad Nacional Autónoma de México, encabezado por el Dr. Gerardo Sierra.

C-value es un algoritmo desarrollado para el idioma anglosajón. Basado, como se mencionó en el capítulo 1, en lingüística y estadística para la extracción de términos multipalabra y ha sido adaptado por el departamento dirigido por el Dr. Sierra para analizar documentos en español.

El algoritmo está dividido en dos etapas, una lingüística y una estadística (Barrón, Sierra, & Villaseñor). Para la primer etapa se requiere el documento pre-procesado en formato POST[30] (etiquetado gramatical) el cual consiste en asignar a cada una de las palabras del texto su categoría gramatical. De esta manera la detección de candidatos a términos se simplifica hasta el grado de detectar patrones y su frecuencia (Barrón Cedeño, Sierra, Drouin, & Ananiadou).

Esta primera etapa se encarga de obtener sintagmas[31] que pudieran ser los términos que interesan en el estudio. Esta etapa se subdivide a su vez en tres partes: (Barrón, Sierra, & Villaseñor)

- Etiquetado de cada palabra del documento con su categoría gramatical.
- Filtro lingüístico por reglas sintácticas (las posibles frases válidas en el lenguaje).
- Filtrado de stopwords que “no se espera que se encuentren en términos del área”.

[30] Part Of Speech Tagging

[31] Tipo de constituyente sintáctico formado por un grupo de palabras que forman otros subconstituyentes, al menos uno de los cuales es un núcleo sintáctivo.

La segunda etapa es la etapa estadística. Una vez que se han obtenido los sintagmas candidatos a ser términos, se realiza un conteo sobre la ocurrencia de dichos sintagmas.

Para que el conteo no sea tan trivial y se logre una aproximación más real a los sintagmas que en verdad son términos y descartar los que no lo son, se evalúan cuatro aspectos importantes.

- El primer aspecto y valor más importante es la frecuencia del sintagma.
- El segundo aspecto es la frecuencia de aparición del sintagma candidato dentro de sintagmas candidatos de mayor cantidad de palabras.
- El tercer aspecto es el número de ocurrencia de dichos candidatos de mayor longitud.
- Por último, se toma en cuenta la longitud del sintagma evaluado.

La ecuación 4.11 muestra la manera de calcular el valor (que da nombre a este algoritmo) para cada sintagma candidato (Barrón, Sierra, & Villaseñor):

ECUACIÓN 4.11

C-VALUE=

$$\begin{Bmatrix} \log_2|a| * f(a), \ \textbf{Para todos los candidatos de mayor longitud } N \\ \log_2|a| * \left(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b) \right), \textbf{Para los candidatos de longitud menor a } N \end{Bmatrix}$$

Donde:

- N= Mayor longitud encontrada en el corpus para un candidato.
- a= Sintagma candidato.
- |*a*|= Longitud del sintagma candidato *a*.
- f(*a*)= Frecuencia de ocurrencia del sintagma *a* en el corpus.
- T_a= Conjunto de candidatos de mayor longitud que contienen al sintagma *a*.
- P (T_a)= Número de candidatos de mayor longitud que contienen al sintagma *a*.
- $\sum f(b)$ = Ocurrencia total de a como subcadena de cualquier sintagma candidato *b* tal que |*a*|<|*b*|

El algoritmo analizado por el equipo del Dr. Sierra debió ser (y fue) modificado para adaptarlo al español ya que “La construcción sintáctica de términos en inglés es totalmente distinta a la de los términos en español” dando como resultado las cuatro reglas mostradas en dicho estudio para la construcción sintáctica de términos para el área de ingeniería. Las cuatro reglas, tomadas del estudio original y son mostradas a continuación:

- <NC|NP>+<PREP><*CARD*>*<NP|NC|ADJ>+
- <NC><NEG>*<ADJ|Vladj|NC|NP>
- <NC|NP>+[<ADJ><PREP><NC>]*
- <NC><PREP>]+[32]<NC>[[<ADJ><CC|PREP>]*<ADJ>]*

En el artículo en inglés (Barrón Cedeño, Sierra, Drouin, & Ananiadou), se mencionan estas cuatro reglas de la siguiente manera:

- (Noun|ProperNoun|ForeignWord)+
- (NounAdj)(PrepDE(Noun|ProperNoun))*
- NounPrepDE(Noun|ProperNoun)
- Noun?Acrnm
- NounPrepDE((NounAdj)|(AdjNoun))

Este algoritmo es de tipo semántico, mientras que el algoritmo principal programado para el Sistema es estadístico por lo que no procedemos a un análisis mayor.

[32] En el artículo original no se encuentra un corchete que abra la expresión, Probablemente por error en la edición. En este caso, se utilizó la regla (intuida):

[<NC><PREP>]+ <NC>[[<ADJ><CC|PREP>]*<ADJ>]* dado que no hay referencias de dicho corchete en el artículo original se optó por colocarlo antes de <NC>.

4.5. RAPID AUTOMATIC KEYWORD EXTRACTION (RAKE)

De entre los múltiples algoritmos que existen creados para la extracción de las palabras clave basados en estadística, probabilidad, lingüística, gramática, etc. RAKE se basa en la frecuencia y coocurrencia de palabras en un texto en formato electrónico.

Rapid Automatic Keyword Extraction es un algoritmo desarrollado "extremadamente eficiente" (Berry & Kogan, 2010) que opera sobre documentos individuales y se encarga de descubrir keywords[33]. Estas frases clave no contienen stopwords[34] debido a que generalmente son borradas en los servicios de indexación por los sistemas de recuperación de información (IR, por sus siglas en inglés) por carecer de importancia léxica.

Los parámetros de entrada para el algoritmo RAKE (Berry & Kogan, 2010) son los siguientes:

- Lista de stopwords (stoplist).
- Lista de delimitadores de frases.
- Lista de palabras delimitadoras.
- El documento de texto en formato digital al cual se le aplica el algoritmo.

[33] Keyword: son una secuencia de una o más palabras que representan el contenido del documento.

[34] Stopword: palabras que deben ser filtradas por no aportar sustancia en la descripción del documento.

Algunas de las palabras que se tomaron como stopwords se muestran a continuación:

... Acá, ahí, ajena, ajenas, ajeno, ajenos, al, algo, algún, alguna, algunas, alguno, algunos, allá, allí, allí, ambos, ante, antes, aquel, aquella, aquellas, esa, esas, ese, eso, esos, esta, estaba, estado...

Algunos de los delimitadores de frase, como signo de puntuación son los siguientes:

"...", ".", ",", ";", "¿", "¡", "?", "!", "{", "}", "[", "]", "(", ")", "\", ":", "<", ">", etc...

Adicionalmente, se realizó una modificación al algoritmo al notar que dependiendo del área del conocimiento sobre el que se trate el documento, se requiere una lista más de palabras “vacías”, adicional al listado de stopwords debido a que no son generales para todas las áreas, de manera que una palabra que sea carente de gran significado para el área de computación no necesariamente lo será para un documento redactado sobre medicina y viceversa.

El algoritmo RAKE toma la coocurrencia de las palabras como un dato fundamental y discriminatorio para la obtención final de la lista de keyword, la coocurrencia que es el resultado de dividir la suma de frecuencia de la palabra más el número de palabras con las que aparece en una frase (a esta suma más adelante le llamaremos *deeg*) entre su propia frecuencia.

Para la obtención de palabras clave candidatas mediante el algoritmo RAKE, primero se debe realizar un pre-procesamiento del texto en el cual, éste se divide en un arreglo dependiendo los delimitadores tomados de los parámetros de entrada al algoritmo: las palabras delimitadoras, los delimitadores de frases, la posición de stopwords y (adicionalmente) los signos de puntuación en el documento. El

resultado es un arreglo que contiene elementos de una o más palabras candidatas a ser keywords.

Los pasos del algoritmo se describen a continuación:

1. Pre-procesamiento del archivo de entrada filtrando las palabras innecesarias.
2. Cálculo de valores para cada una de las palabras resultantes del filtrado al documento original, necesarios para la distinción de keywords multitérmino candidatas:
    a. Freq (w): es la frecuencia de cada palabra.
    b. Deeg (w): consiste en sumar la frecuencia de la palabra más el número de palabras que conforman la(s) frase(s) donde esa palabra aparece.
3. Cálculo de coocurrencia por palabra. Es la puntuación más importante a tomar en cuenta para cada palabra ya que es la que determina, en la siguiente etapa, el peso de un vecindario de palabras y si este vecindario puede tomarse o no en cuenta como un elemento de keywords del documento y se obtiene de la siguiente manera:
    a. Cooc (w)=deeg (w)/freq (w)
4. Posteriormente se calcula el peso de la frase candidata a ser keyword del documento, siendo este valor la suma de la Cooc (w) del conjunto de palabras que forman la frase o keyword candidata.
5. El siguiente paso consiste en realizar una matriz bidimensional en la que se registre la aparición de las palabras de manera individual y colectiva. La matriz contiene un cruce de las palabras registradas en el documento donde en la diagonal principal (tupla formada por la misma palabra en la fila y la columna) se registra el número total de veces que aparece la palabra en cuestión (frecuencia de palabra) y cuando una palabra aparece acompañada de otra (tupla formada por una palabra en fila con otra diferente en la columna), se registra el número de veces que ocurre dicha correlación en la casilla donde se cruzan las palabras en cuestión. Para este paso se modificó el algoritmo original y se implementó la utilización de una base de datos

debido a la facilidad de emplear dicha herramienta para almacenar la información de los registros en tablas (emulando la matriz bidimensional). El factor definitivo de la implementación de una base de datos se debe a la naturaleza de SQL[35]: su fácil utilización para la selección, manipulación y comparación de registros. Las tablas contienen la información calculada en los dos pasos anteriores.

6. Ordenamiento descendente por el peso de cada conjunto de keywords candidatas. Se estima que una tercera parte de dicho arreglo más uno, ordenado de mayor a menor puntuación de peso de frase es el conjunto de palabras clave del documento. (Berry & Kogan, 2010).

[35] Structured Query Language es el lenguaje estructurado de consultas que permite la interacción con el motor de base de datos para la manipulación de registros.

4.5.1. DIAGRAMA A BLOQUES

Para explicar mejor el funcionamiento del algoritmo RAKE, se presenta la ilustración 4.2 con el diagrama a bloques basado en el algoritmo diseñado para archivos de texto en formato digital para el idioma inglés y modificado y adaptado al idioma español.

ILUSTRACIÓN 4.3 Diagrama a bloques Algoritmo RAKE. Fuente: Propia.

4.5.2. EJEMPLO

Para comprender de una mejor manera el algoritmo, se presenta un pequeño ejemplo en el cual, tomamos un fragmento del texto “El desarrollo del software” de Complejidad y tecnologías de la información (Vacas Sáez, 1992) al cual se le aplicará el algoritmo RAKE y se mostrarán los resultados obtenidos.

“el gran desafío con que se encuentra la gestión de proyectos de software consiste precisamente en limitar los productos que se desarrollan en esos proyectos a unos niveles de complejidad aceptables y manejables. Dicho de otra forma, se pretende reducir los grados de libertad en la producción de software para, al operar dentro de unos ciertos márgenes, mantener la complejidad resultante lo más baja posible.”

El preprocesamiento consiste en separar el texto en frases, ya sea por signos de puntuación o por palabras delimitadoras de frase. Como ejemplo de la fragmentación del texto, a continuación se muestran las primeras tres frases fragmentadas:

a) el gran desafío con que ¬¬ ¬¬ la gestión de proyectos de software ¬¬ precisamente ¬¬ ¬¬ los productos que ¬¬ desarrollan ¬¬ esos proyectos a unos niveles de complejidad aceptables y manejables
b) Dicho de otra forma
c) ¬¬ pretende ¬¬ los grados de libertad ¬¬ la producción de software para

El texto preprocesado ahora es un conjunto de líneas candidatas a ser keywords multipalabra. De las cuales, se deben eliminar palabras *carentes de peso* para el documento en cuestión. Siendo necesario definir un listado de palabras carentes según sea el área del conocimiento del documento a procesar. (Ingeniería; ingeniería en computación; ingeniería en computación en español; ingeniería en computación en español en México; etc.).

El siguiente paso, ya propio del algoritmo, es aplicar cálculos de frecuencia y coocurrencia a cada palabra/frase; en seguida, se listan las primeras 10 palabras o frases candidatas:

- desafío
- gestión proyectos software
- desarrollan
- proyectos manejables
- grados
- producción software

Posteriormente, el algoritmo pide calcular:

a) F_w que es la frecuencia de la palabra w_i,
b) $deeg_w$ (palabras vecinas con las que aparece w_i en cada frase donde se repite, incluida su propia frecuencia) y
c) $cooc_w$ que es el resultado de dividir $deeg_w/F_w$

Donde:

- w_i es la palabra en cuestión.

En la tabla 4.4 se muestra el resultado de los cálculos para dos palabras en particular:

Palabra	deeg	freq	coocurrencia
SOFTWARE	5	2	2.5
PROYECTOS	5	2	2.5

TABLA 4.4 Resultados calculados. Fuente: Propia.

Podemos observar en la tabla que la palabra *software* tiene deeg=5, freq=2 y coocurrencia=2.5. Esto se debe a que las frases en las que aparece la palabra son:

- gestión proyectos software
- producción software

Analicemos ambas frases. La primera frase cuenta con 3 palabras, mientras que la segunda cuenta con 2. En total son 5 palabras.

La frecuencia de la propia palabra es 2.

Deeg (w) se calcula sumando precisamente el tamaño de las frases donde la palabra en cuestión aparece: 3+2 y obtenemos el 5, resultado en la primera columna.

Por último, la coocurrencia es el cociente de dividir deeg entre la frecuencia (5/2).

El peso en el documento de cada frase candidata se obtiene al sumar el valor de la coocurrencia de cada palabra en la frase, logrando así diferenciar las frases que contienen las palabras más representativas del documento. Esto, dependiendo de la aparición en conjunto de las palabras que forman la frase.

El listado se ordena de mayor a menor según el peso de la frase.

El algoritmo RAKE sugiere que se seleccione *aproximadamente* una tercera parte del total de líneas más 1.[36] (Berry & Kogan, 2010)

Para este ejemplo, se toman las 3 frases relevantes del texto procesado y se muestran en la tabla 4.5.

[36]Se explica en el paso 5 de la descripción del algoritmo.

#	Compuesto	veces	Valor calculado
1	Gestión¨¨PROYECTOS¨¨SOFTWARE	1	15.5
2	Producción¨¨SOFTWARE	1	10.5
3	Desafío	1	4
4	PROYECTOS	1	4

TABLA 4.5 Frases candidatas. Fuente: Propia.

Estos datos, pueden ser posteriormente almacenados para ser consultados desde un IRS como se menciona en el capítulo III de la presente tesis.

A simple vista, también podemos hacernos una idea general de lo que el autor está hablando en el texto original. Como ejercicio mental, recordemos el texto al que se le aplicó el algoritmo y encontraremos frases como “la gestión de proyectos de software”, “grados de libertad en la producción de software”, etc.

Encontramos, entonces, algunas palabras que hacen alusión a esas frases y que al ser procesadas por el algoritmo RAKE tienen un valor calculado bastante notorio.

Capítulo 5. Desarrollo de algoritmo

INTRODUCCIÓN

En este capítulo, se muestran los pasos adoptados para el desarrollo de los algoritmos planteados en el capítulo anterior. Los problemas encontrados durante la codificación y la manera es que se llegó a la resolución de los mismos.

Las modificaciones realizadas sobre los algoritmos originales para una mejoría en cuanto a facilidad de programación y optimización de resultados.

5.1. MODELADO DEL SISTEMA

Para cualquier desarrollo de software es necesario llevar un orden de tal modo que todas las etapas del desarrollo sean coherentes y formales (Kraffczyk Fuentes, 2003) es por ello que se modeló el sistema de la siguiente manera; de tal forma que resultara simple al momento de conceptualizar el objetivo y traspasar esa idealización a código funcional que formara parte del sistema final.

Los módulos del sistema son los siguientes:

- Interfaz web de usuario IWU (WUI).
    - Parte visible del sistema.
    - Módulo con el que el usuario interactúa.
    - Sirve como puente de entradas y salidas al sistema.
- Carga de archivos.
    - Módulo para enviar archivos al servidor.
    - Debe permitir cierto formato y tamaño de archivos.
    - Almacenamiento con nombres clave de archivos.
- Preprocesamiento de texto.
    - Módulo filtro del texto cargado.
    - Moldea el texto de entrada para un procesamiento posterior.
- Procesamiento de texto.
    - Algoritmos inteligentes de detección de palabras clave.
        - RAKE
- Base de datos.
    - Almacenamiento del texto procesado.
- Análisis de resultados.
    - Procesamiento del texto almacenado en base de datos.
    - Decisiones de importancia para palabras clave
- Respuesta de información.
    - Ordenamiento de los resultados y entrega de los mismos a la IWU.

Este modelado permite el desarrollo modular en cascada (que, a la postre, puede convertirse en evolutivo) del sistema para conseguir el producto final. La interacción entre módulos se muestra en la ilustración 5.1.

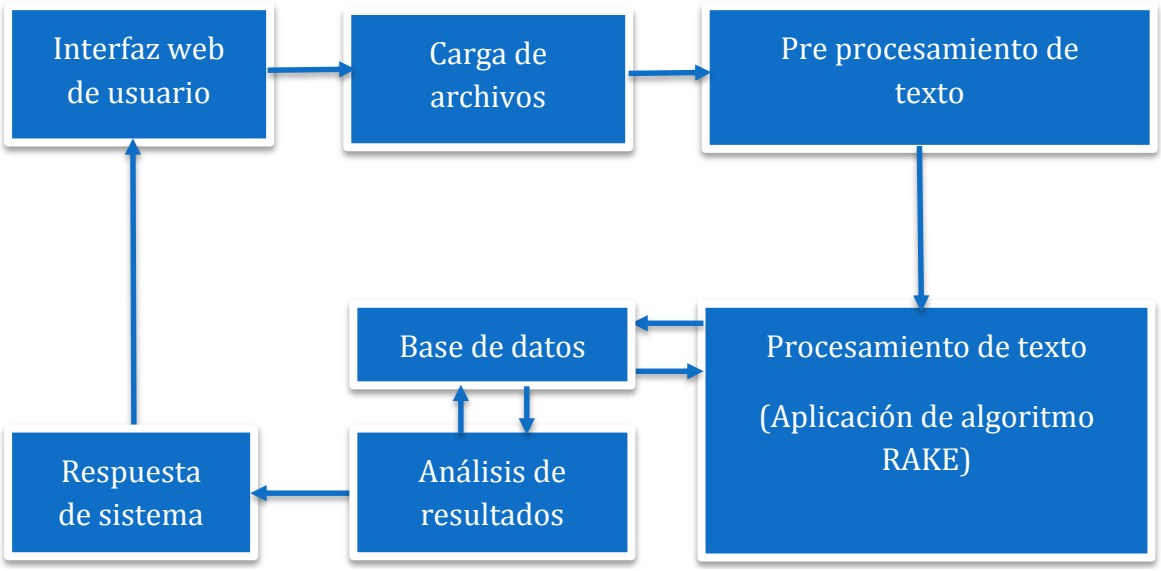


ILUSTRACIÓN 5.1 Modelado del sistema. Fuente: Propia.

5.2. DIAGRAMA A BLOQUES Y PSEUDOCÓDIGO

Aunque ya se ha planteado tanto el diagrama a bloques y pseudocódigo de los algoritmos trabajados durante esta tesis, queda por presentar los que representan al trabajo final, es decir, el sistema como unificación de los algoritmos planteados capaz de entregar un conjunto de palabras clave representativas del texto procesado. El diagrama a bloques se muestra en la ilustración 5.2.

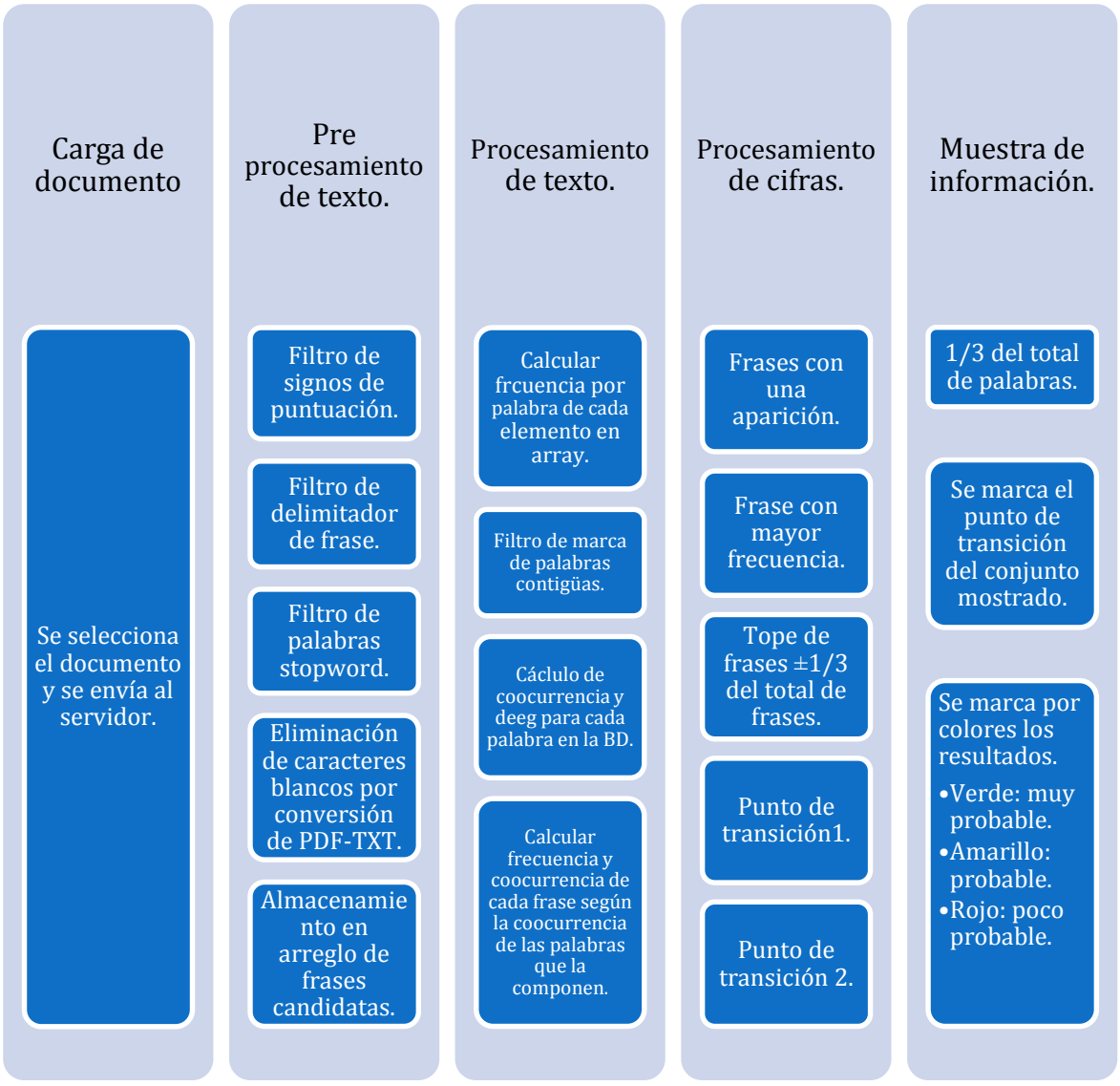


ILUSTRACIÓN 5.2 Diagrama a bloques. Fuente: Propia.

5.3. DESARROLLO

El desarrollo de este sistema se ha llevado a cabo en el departamento de Producción de la Dirección General de Bibliotecas, Biblioteca Central, UNAM.

Las definiciones del equipo sobre el que es ejecutado actualmente se encuentran en la tabla 5.1, incluyendo hardware y software. Es importante mencionar que no son los requisitos mínimos y que el sistema no se limita a los siguientes valores para un correcto funcionamiento:

Aspecto	Valor
Hardware	
Procesador	Intel core i7 (64 bit)
RAM	8 GB
Software	
Sistema operativo	Windows 7 64 bit
Servidor web	Apache 2.4.7
PHP	5.5.9

TABLA 5.1 Definición de Hardware y software que soporta al sistema. Fuente: Propia.

A continuación se describe la toma de requerimientos que funcionaron para un buen desarrollo del sistema.

La definición de requerimientos, queda en parte descrita en el capítulo 1, en el cual se menciona la necesidad de un sistema capaz de procesar de manera inteligente y automatizada un texto de entrada, de tal forma que detecte palabras unitérmino y multitérmino que definan el contenido del texto introducido.

REQUERIMIENTOS

- El software debe ser capaz de recibir un archivo en texto plano.
- El software debe ser capaz de procesar el texto introducido y calcular mediante la combinación de uno o más algoritmos de procesamiento inteligente de texto.
    - Ley de Zipf.
    - Punto de transición de Goffman.
    - Rapid Automatic Keyword Extraction.
- El software debe ofrecer un listado con las palabras más relevantes del texto.
- El software debe ofrecer un listado con las frases más relevantes del texto.
- El software debe indicar del listado anterior, cuáles son las frases más probables de ser representativas del texto.
- El software debe ser accesible vía web.
- El software debe ser accesible vía web.
- El software debe ser de simple manejo.
- El software debe entregar resultados de fácil lectura para el usuario.

5.3.1. CASOS DE USO

Gráficamente, se representa el diagrama de casos de uso en la ilustración 5.3, en la que se especifica la interacción que tiene el usuario con el sistema, así como las posibles respuestas que el sistema determine.

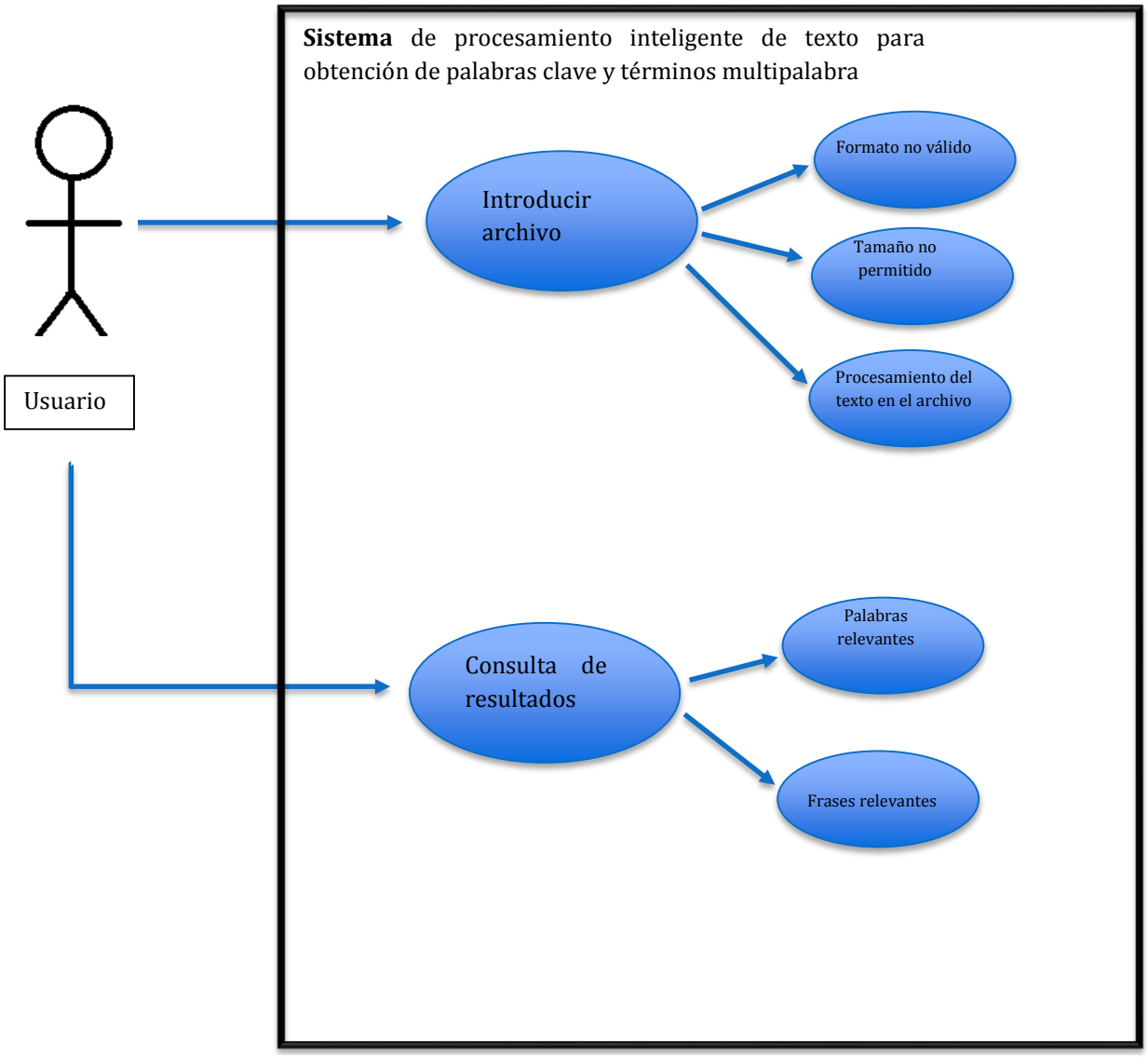


ILUSTRACIÓN 5.3 Casos de uso. Fuente: Propia.

5.3.2. DISEÑO DE SISTEMA

El diseño del sistema puede observarse en la ilustración 5.4, en la que se denota de una manera más clara las interacciones entre los módulos (y submódulos) del sistema, así mismo, se denota el proceso que sigue el tratamiento del texto introducido.

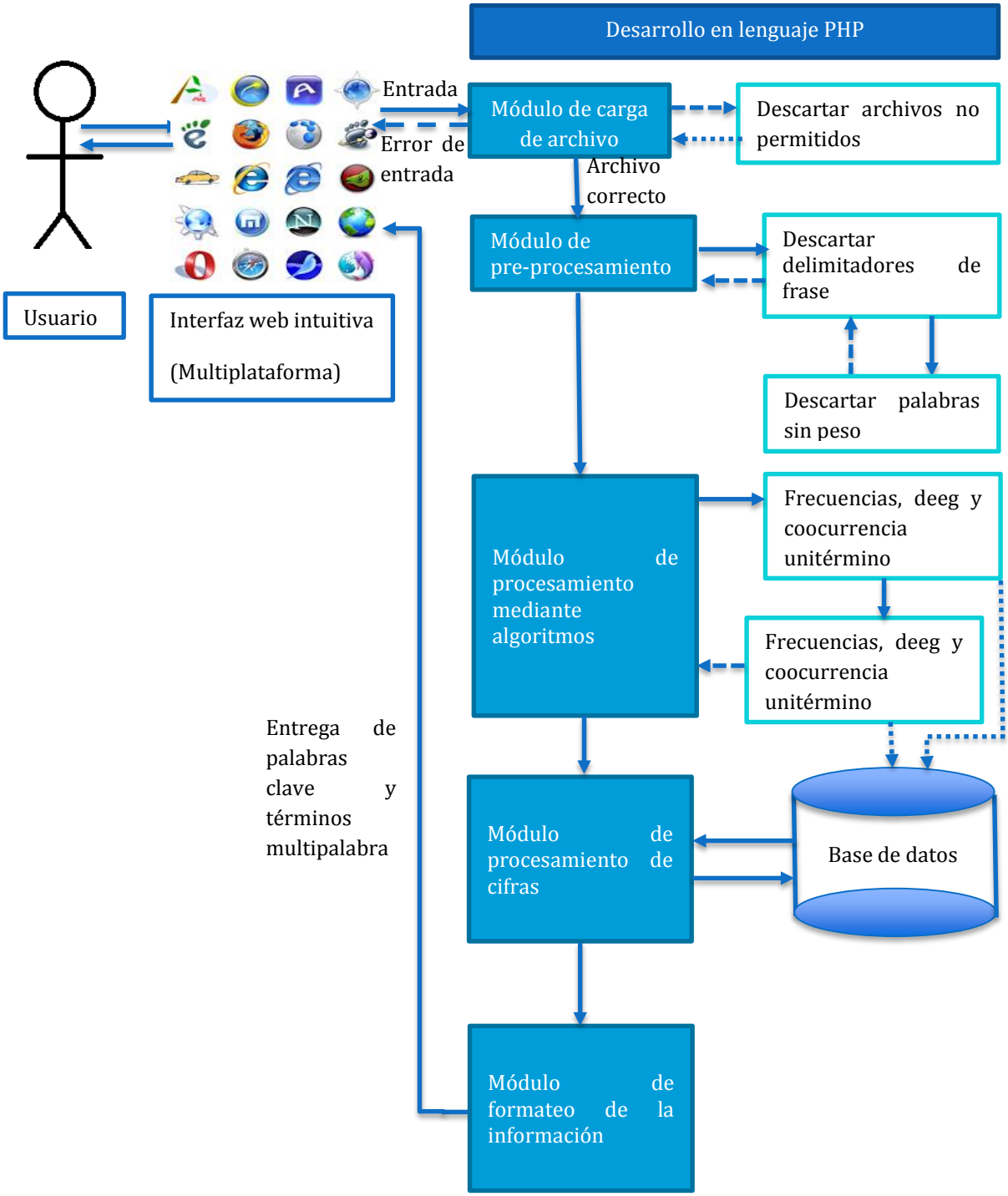


ILUSTRACIÓN 5.4 Diseño del sistema. Fuente: Propia.

5.4. RESULTADOS

Se han realizado diversos experimentos con artículos de diferentes áreas del conocimiento, de diferente tamaño y autores muy variados.

Se han realizado múltiples pruebas particulares con un corpus relacionado a la ingeniería y particularmente a la ingeniería de software las cuales han permitido *entrenar* el sistema, haciéndole aprender palabras nuevas que agregar al listado de verbos, palabras irrelevantes en el área y los denominados stopwords.

Isidoro Gil (Gil Leiva, 1999) describe un método para evaluar la consistencia[37] entre la indización manual y la automatizada, el cual consiste en la ecuación 5.1.

ECUACIÓN 5.1 $C=T/((A+B)-T)$

Donde:

- C es la consistencia entre la indización automatizada y la manual.
- T es el número de términos comunes asignados por los dos sistemas o indizadores.
- A es el número de términos asignados por el sistema o indizador número 1.
- B es el número de términos asignados por el sistema o indizador número 2.

Adicionalmente, para calcular el porcentaje de consistencia, el resultado C debe multiplicarse por 100.

Para conocer la consistencia entre el método de indización manual (aquellas palabras clave propuestas en el artículo) y las palabras clave propuestas por el sistema, procedemos a realizar las siguientes pruebas.

[37] Se define como consistencia al parecido entre la lista de descriptores obtenida de manera manual por un analista documental y la lista de descriptores obtenida de manera automática por un sistema.

Nombre del artículo:

- *La ley de Zipf y el punto de transición de Goffman en la indización automática*

Páginas como PDF: 22

Páginas como solo texto (arial 12): 46

Palabras: 7589

Tamaño de archivo en formato .txt: 49 kB

Autor (es):

- Rubén Urbizagástegui Alvarado.
- Cristina Restrepo Arango.

Palabras clave propuestas en el artículo:

- Ley de Zipf.
- Punto de Transición de Goffman.
- Bibliometría.
- Cienciometría; Infometría.

Tiempo estimado por el sistema: 14 minutos.

Tiempo real: 7 minutos y 40 segundos.

15 n-términos descriptores propuestos por el sistema (el símbolo ¨¨ es la interpretación del sistema un espacio/stop word):

1. Palabras.
2. **Ley¨¨Zipf.**
3. **Punto¨¨transición¨¨Goffman.**
4. Punto¨¨transición.
5. Palabras clave.
6. Frecuencia.
7. Investigación Bibliotecológica*.
8. Capsicum*.
9. Goffman.
10. Indización.
11. Zipf.

12. Indizador.
13. Pubescens*.
14. Cromosomas.
15. Rango.

Calculando la consistencia, a partir de la ecuación 5.1 tenemos:

C=T/((A+B)-T)

C=2/((4+15)-2)

C=2/17

C=0.11.

Porcentaje de consistencia = 11%

Pero ¿esto qué quiere decir? Simplemente quiere decir que eligiendo 15 descriptores arrojados por el algoritmo RAKE, el sistema se parece a un humano eligiendo las palabras relevantes tan solo en un 11%.

Isidoro Gil (Gil Leiva, 1999) también se explica que para considerar una consistencia aceptable, ésta debe encontrarse entre el 25 y el 60%. Partiendo de ello, ahora queremos saber cuántos términos mostrados por el sistema, deberíamos considerar para entrar en el rango mínimo:

C=T/((A+B)-T)

B=(T/C)+T-A

B=(2/.25) +2-4

B=6

Tomando en cuenta las primeras 6 palabras, resultan más significativas como descriptoras que el resto de las palabras en el listado tendríamos un sistema que selecciona descriptores de manera muy similar a como lo hace el humano que procesó el archivo mencionado.

Pasemos ahora al siguiente ejemplo:

Nombre del artículo:

- *Estandarización de los procesos asociados al desarrollo de proyectos informáticos: un caso de estudio* (Sepúlveda Cuevas & Cravero Leal).

Páginas como PDF: 15

Páginas como solo texto (arial 12): 69

Palabras: 7981

Tamaño del archivo en formato .txt:55 kB

Autor (es):

- Samuel Sepúlveda Cuevas.
- Ania Cravero Leal.

Palabras clave propuestas en el artículo:

- Proyectos Informáticos.
- Mejora de Procesos.
- Calidad del Software.
- Investigación-Acción.

Palabras clave en inglés (Propuestas en el artículo):

- IT Projects.
- Process Improvement.
- Software Quality.
- Action Research.

Tiempo estimado por el sistema: 15’ 29’’

Tiempo real: 8’ 1’’

20 n-términos descriptores propuestos por el sistema:

1. Software.
2. Etapa.
3. Calidad.
4. Desarrollo.
5. Producto.

6. Etapas.
7. Proyecto.
8. ISO.
9. Diseño.
10. Software Engineering.
11. **Proyectos informáticos.**
12. **Procesos.**
13. Estándares.
14. Equipo.
15. **Calidad del software.**

Apliquemos ahora la ecuación 5.1 para calcular la consistencia y, en su caso, cuántas palabras deberíamos tomar del listado para que entre dentro del porcentaje mínimo sugerido para que el sistema se asemejara al humano que seleccionó las palabras clave.

C=T/((A+B)-T)

C=3/((4+15)-3)

C=.18

Porcentaje de consistencia: 18%

B=(T/.25)+T-A

B=(3/.25)+3-4

B=11.

Sin embargo, podemos notar que la primera palabra ofrecida por el sistema en encontrarse entre las palabras proporcionadas en el artículo como keyword aparece en la posición 11. Como consecuencia, podríamos definir este resultado como entrenador, ya que nos permite entender que el texto podría tener otros keywords como *software*, *calidad*, *desarrollo* o *estándares*.

Es importante mencionar que la consistencia no es un referente de qué tan bien o mal extrae un sistema automatizado, una persona experta o un principiante, los descriptores de un documento. Únicamente sirve para determinar qué tanto el proceso se parece entre dos entidades que extraen las palabras clave. Entre esa semejanza, se incluyen los errores y omisión de descriptores. Partiendo de esta idea, debemos considerar si queremos que el sistema se parezca -o no- al proceso manual o a otro método de extracción. Por ejemplo, la palabra bibliometría, en el primer ejercicio, no aparece en el cuerpo del documento, sin embargo, el autor decidió que podría ser un descriptor por materia, acompañado de otros descriptores del tipo unitérmino y multitérmino.

El sistema es accesible desde internet a través de la ruta 132.248.67.57/algoritmos/RAKE/

Para el desarrollo del sistema se ocuparon 2 listas tanto de stopwords como de verbos y delimitadores de frase.

El primer archivo tiene por nombre Stopwords.palabras con un total de 1334 palabras y un tamaño de 9KB.

El segundo archivo, tiene por nombre verbos.palabras con un total de 2606 palabras para un tamaño de 22 KB.

Ambos archivos se encuentran en constante crecimiento conforme se prueba el sistema y se especializa en una rama del conocimiento.

Conclusiones

Una vez desarrollado el sistema, me di cuenta de que el algoritmo RAKE presenta un leve inconveniente: entre más grande sea el archivo procesado, mayor será el tercio de registros que sugiere, es decir, de un archivo con 10 términos detectados por el algoritmo RAKE como “posibles candidatos”, se selecciona el tercio superior (3.33~): 3 registros “keywords”. Pero al procesar un archivo con mil términos, el algoritmo sugerirá 333 términos como keywords. Es por esto que decidí crear una ventana variable dentro del tercio que marca el algoritmo. Un subconjunto.

Noté que de alguna manera, el caso de la ley de Zipf se repetía en el listado de palabras que arrojaba el algoritmo RAKE, pero no se puede aplicar tal cual el punto de transición de Goffman sobre dicho listado ya que mientras que el punto de transición es calculado mediante frecuencias, el algoritmo RAKE categoriza los N-términos a partir de la coocurrencia de las palabras que los conforman. Partiendo de esta premisa, me di a la tarea de encontrar un punto de transición de coocurrencias en el listado de los N-términos.

Encontré dos situaciones:

- Existe la posibilidad (sí, muy pequeña, pero existe) de que nos encontremos con el caso de que no existan N-términos con coocurrencia de 1 (A la postre, entendí que también podría darse el caso de no existir palabras en un texto con frecuencia unitaria para el cálculo del P.T.G[38] lo que conllevaría a un P.T.G=0 que pondría muy abajo dicha marca).
- Para encontrar un punto de transición entre las coocurrencias más y menos frecuentes dentro de la ventana sugerida por el algoritmo RAKE (una tercera parte de los N-términos totales), era necesario tomar ese tercio de N-términos como el nuevo universo y, entonces sí, encontrar los N-términos más relevantes. Esto me facilitó encontrar que cuanto más grande es el texto que se procesa por el sistema desarrollado, existe una menor probabilidad de que en el tercio superior de N-términos, se encuentren coocurrencias unitarias (incrementando la posibilidad de encontrarnos con el caso del punto anterior). Por lo que decidí realizar una modificación a la ecuación 4.3 propuesta por Boyce ($In/I_1 = 2/n(n + 1)$) y al procedimiento que plantean para la obtención del punto de transición (Boyce & Lockard, 1975)

Para tener un punto de transición más adecuado dependiendo del universo con el que se esté trabajando, se debe utilizar la ecuación 6:

Ecuación 6: $\mathbf{I_{nm}/I_m = 2/n(n + 1).}$

Donde:

1. I_{nm} es el inicio de vecindario mínimo para el universo que se está procesando. Considerando que el vecindario de palabras descriptoras no comienza cuando $I_n=1$ ya que en realidad no es la mínima posible para todos los universos debido a que puede darse el caso que $I_n=2$, $I_n=3$, etc.

[38] P. T. G. Punto de Transición de Goffman.

2. l_m es el número de palabras con la menor frecuencia (o en el caso del algoritmo RAKE, el menor valor calculado por coocurrencia) para el universo seleccionado.

Con esta ecuación, se puede calcular el punto de transición del valor calculado para la ventana seleccionada de cualquier universo de palabras, en función del registro con menor valor dentro de esa ventana. En este caso, el tercio de los N-términos arrojado por el algoritmo RAKE.

Tras el desarrollo del sistema de procesamiento inteligente de texto, se ha podido comparar el resultado de las palabras y términos multipalabra obtenidos por el sistema, contra las palabras descriptoras planteadas por el autor, indizador o editor para el texto procesado con resultados realmente buenos y que en verdad fueron esperados.

Teniendo en cuenta que el objetivo general consiste en el desarrollo de un sistema basado en métodos estadísticos para la extracción de palabras clave, el objetivo ha sido cumplido ya que se ha desarrollado la herramienta que procesa archivos de texto plano en formato digital para extraer los términos descriptores del documento.

No sólo se desarrolló la herramienta, sino, se optimiza el proceso de extracción de palabras clave y términos multipalabra procesando de 5 a 7kB por minuto, lo que representa unas 1200 palabras por minuto; similar a leer 4 páginas (en tipografía arial 12) extraer la frecuencia por cada palabra, calcular la coocurrencia de cada palabra en cada frase, ordenar los resultados y mostrar las más relevantes y todo, en un minuto.

Con esta herramienta es posible extraer de manera certera, descriptores que constan de más de una palabra, incluso frases complejas que en su momento llamamos n-términos.

Con esto, podemos dar por confirmada la hipótesis ya que esta herramienta facilita y optimiza la labor de extracción de términos multipalabra descriptores y con ello, el proceso general de indización de archivos digitales.

Partiendo del hecho que un sistema de recuperación de información toma su fortaleza a partir del listado en base de datos de palabras relevantes, el ofrecer descriptores más próximos a lo que el autor pretende transmitir, los IRS actuales se enfrentan al siguiente caso:

En el capítulo 3 se habló acerca de los tipos de consulta que reciben los IRS (palabra única, por frase y booleana). Recordando que

a) las consultas que generan resultados más exactos (el usuario escribe exactamente la palabra que espera recuperar), son aquellas en las que la palabra a buscar se encuentra en el descriptor del documento y
b) las consultas más frecuentes para el IRS son aquellas en las que el usuario no sabe exactamente cómo buscar.

Queda entonces la posibilidad de que entre las consultas que se hagan del tipo *b* encuentren un descriptor como si hubiesen intentado realizar una consulta del tipo *a* y entonces, el IRS encuentre más posibles resultados *relevantes* lo que podría llegar a resultar en un problema de precisión.

Con los ejemplos analizados en el capítulo 5, podemos determinar que el sistema cumple con los requerimientos propuestos, ya que en menos de un minuto, es capaz de procesar un archivo digital relativamente pequeño de texto plano, seleccionar las palabras más relevantes, hacer los cálculos necesarios para la generación de la lista final y presentarlos de una manera clara.

El sistema es útil para lo que fue pensado: la extracción de descriptores. Pero más allá de si describen bien o no el documento, nos encontramos ahora con una nueva ventaja colateral: Un IRS encuentra más resultados relevantes entre un universo mayor de descriptores que se asemejen más a la idea que el autor pretende transmitir.

A pesar de que tanto los objetivos, la hipótesis y el conjunto de requerimientos fueron satisfechos y corroborados, es necesario mencionar las posibles líneas de crecimiento del sistema, como la constante necesidad de actualización de los listados de palabras delimitadoras de frase, stopwords y verbos que se utilizan en

este sistema, ya que al procesar diferentes corpus [39] encontraremos cierta discrepancia entre las palabras irrelevantes de un tema, con respecto de las de otro.

Por otro lado, el sistema actual únicamente es capaz de procesar un archivo a la vez ya que sobrescribe las tablas que utiliza. Fue pensado así debido a la limitante de almacenamiento que en el ambiente de desarrollo se tuvo.

Una tercera línea de crecimiento y futuro desarrollo consiste en la capacidad de almacenar en una base de datos los resultados obtenidos, de tal manera que se pueda volver a consultar sin necesidad de volver a procesar el archivo.

[39] Conjunto de documentos de un tópico en particular.

Referencias

Academia. (2015). *Academia.* Recuperado el 3 de 03 de 2015, de https://www.academia.edu/9789746/Tipos_de_Bases_de_Datos

Academia. (2015). *Academia.* Recuperado el 10 de 12 de 2014, de https://www.academia.edu/9785466/Tipos_de_Bases_de_Datos

Acosta Mata, V. (2004). El módulo de adquisiciones en Aleph: su utilización en la UNAM. *Biblioteca Universitaria, 7*, 23-33.

Arteta, I. (s.f.). *Modelo de cascada y espiral*. Recuperado el 205, de http://modelo-cascada.blogspot.mx/

Babb, E. (15 de Junio de 1976). *Londres, Inglaterra Patente nº 3964029.*

Bañuelo Saucedo, A., & Manzanarez Gómez, N. (03 de 02 de 2012). *División de ciencias básicas.* Recuperado el 11 de 11 de 2013, de http://dcb.fi-c.unam.mx/users/angellbs/htm/GRUPO1/ARCHIVOS_VARIOS_G1/PyE_T1.pdf

Barrón Cedeño, A., Sierra, G., Drouin, P., & Ananiadou, S. (s.f.). *An Improved Automatic Term Recognition Method for Spanish.* México, España, Candá, Inglaterra: Engineering Institute, Universidad Nacional Autónoma de México.

Barrón, L. A., Sierra, G., & Villaseñor, E. (s.f.). *C-value aplicado a la extracción de términos multipalabra en documentos técnicos y científicos en español.* México: Instituto de Ingeniería, UNAM.

Berry, M., & Kogan, J. (2010). *Text mining: Applications and theory.* Estados Unidos: WILEY.

Biblioteca Central Facultad de Derecho. (s.f.). *Análisis y recuperación de información.* Chile: Universidad de Chile.

Boyce, B., & Lockard, M. (1975). Automatic and manual indexing perfformance in a small file of medical literature. En *Bull. Med. Libr. Assoc.* (págs. 378-382). Columbia, Missouri: School of library and informational science University of Missouri.

Braun, E. (1996). *CAOS, FRACTALES Y COSAS RARAS.* México: Fondo de cultura económica. Recuperado el 01 de 09 de 2014, de http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen3/ciencia3/150/htm/sec_23.htm

Churcher, C. (2007). *Beginning database design from novice to professional.* Apress.

Churcher, C. (2007). *Beginning SQL Queries From Novice to Professional.* Estados Unidos: Apress.

Churcher, C. (2008). *Beginning SQL Queries From novice to professional.* Estados Unidos: Apress.

CINDEX. (s.f.). *Cindex Indexing research*. Recuperado el 18 de 03 de 2015, de http://indexres.com/soft.php

Coronel, C., Morris, S., & Peter, R. (2011). *Bases de datos. Diseño, implementación y administración.* Cengage learning.

Dechile.net. (2001). *Origen de las palabras*. Recuperado el 10 de 10 de 2013, de http://etimologias.dechile.net/

Delgado García, J. (2015). *UT-01 Principios de Seguridad y Alta Disponibilidad - Parte 1.*

Diego Cabrera, L. (2011). *TF-IDF PARA LA OBTENCIÓN AUTOMÁTICA DE TÉRMINOS Y SU VALIDACIÓN MEDIANTE WIKIPEDIA.* México, DF.: Universidad Nacional Autónoma de México.

Dirección General de Bibliotecas. (2008). Sistema Aleph 500. El sistema bibliotecario de la UNAM en cifras. *Biblioteca Universitaria, 2*, 49-56.

Epifanio Tula, L. G., & Medeot Matías, D. (2008). *Sistema de Recuperación de Información Motor de Búsqueda: Innuendo.* Córdoba: Universidad Tecnológica Naciona.

Gama Ramírez, M. (s.f.). *Acervo histórico del cómputo en la UNAM*. (Dirección General de Servicios de Cómputo Académico (DGSCA), UNAM) Recuperado el 15 de Septiembre de 2015, de Sistemas de Automatización en las Bibliotecas de la DGSCA: http://www.historiadelcomputo.unam.mx/files/40anos/memorias_40_anos/edu/edumgr33.htm

Gil Leiva, I. (1999). *la automatización de la indización de documentos.* Trea.

Google. (2011). *Dentro de Google*. Recuperado el 15 de agosto de 2014, de http://www.google.com.mx/intl/es-419/insidesearch/howsearchworks/algorithms.html

Harmon, G. (13 de Diciembre de 2007). *Remembering William Goffman: Mathematical information science pioneer.* Recuperado el 03 de 09 de 2014, de http://garfield.library.upenn.edu/papers/goffman.pdf

History channel. (2012). Humanidad: La Historia de Todos Nosotros. Estados Unidos.

Informática Moderna. (2015). *Informática Moderna*. Recuperado el 27 de 02 de 2015, de http://www.informaticamoderna.com/Info_dat.htm

International Standard. (01 de 12 de 1985). *Documentation - Methods for examining documents, determining their subjects, and selecting indexing terms.* Recuperado el 09 de 10 de 2013, de https://courses.washington.edu/is530/win08/LIS%20530%20Readings/ISO_1985.pdf

ISO. (s.f.). *International Organization form Standarization*. Recuperado el 01 de 08 de 2015, de http://www.iso.org/iso/home.html

Kraffczyk Fuentes, J. F. (14 de mayo de 2003). *Realidad virtual aplicada al tratamiento del trastorno de lateralidad y ubicación espacial.* Cholula: Universidad de las Américas Puebla. Recuperado el 08 de enero de 2015, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/fuentes_k_jf/

Laboratorio nacional de calidad del software. (2009). *ingeniería del software: Metodologías y ciclos de vida.* España: Instituto Nacional de Tecnologías de la Comunicación.

Marciniak, M. (2009). *Aspects of Natural Language Processing.* Berlín, Alemania: Springer-Verlag.

Medina Trejo, D. (2010). Ingeniería de Software. México.

Mills, H. D. (1980). The management of software engineering, Part I: Principles of software engineering. *IBM Systems Journal, 19*(4), 414-420. Recuperado el 01 de 2015, de http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5387924&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5288519%2F5387922%2F05387924.pdf%3Farnumber%3D5387924

Parra, S. (25 de 09 de 2010). *Papel en blanco*. (Papelenblanco) Recuperado el 01 de 09 de 2014, de http://www.papelenblanco.com/metacritica/la-ley-de-zipf-la-frecuencia-con-la-que-una-palabra-aparece-en-un-texto

Pigna, F. (2015). *El historiador*. Recuperado el 07 de 2015, de http://www.elhistoriador.com.ar/aula/antigua/cuando_empezo_la_historia.php

Rauch, C., & Rauber, A. (2004). Preserving Digital Media: Towards a Preservation Solution Evaluation Metric. *Digital Libraries: International Collaboration and Cross-Fertilization. 7th International Conference on Asian Digital Libraries* (págs. 203-212). Shanghai: Springer.

Real Academia Española. (2001). *Diccionario de la lengua española*. Recuperado el 10 de 10 de 2013, de http://lema.rae.es/drae/

Rodríguez Gallardo, A. (Enero-Junio de 2001). Por qué una Biblioteca Central. *Biblioteca Universitaria, Nueva Época, 4*(1), 13-17. Recuperado el 16 de 09 de 2015, de http://www.dgbiblio.unam.mx/servicios/dgb/publicdgb/bole/fulltext/volIV12001/pgs_13-17.pdf

Rodríguez Gallardo, A. (2007). Definiendo la lectura, el alfabetismo y otros conceptos relacionados. *Investigación bibliotecológica, 21*(42), 143-175.

Sempere Galán, E. M. (03 de 10 de 2012). *Alquibla*. Recuperado el 11 de 11 de 2013, de http://www.alquiblaweb.com/2012/10/03/la-indizacion/

Sepúlveda Cuevas, S., & Cravero Leal, A. (s.f.). *Estandarización de los procesos asociados al desarrollo de proyectos informáticos: un caso de estudio.* Temuco, Chile.: Universidad de la Frontera, Departamento de Ingeniería de Sistemas, Centro de Estudios en Ingeniería de Software (CEIS).

SKY Software. (11 de 02 de 2015). *SKY Index Professional*. (SKY Software) Recuperado el 18 de 03 de 2015, de http://www.sky-software.com/

Sommerville, I. (2001). *Software Engineering (6a ed.).* Pearson.

Suryn, W. (2014). *Software Quality Engineering:A Practitioner's Approach.* Wiley-IEEE Press.

Universidad Nacional Autónoma de México, Dirección General de Bibliotecas. (2013). *Numeralia 2013*. Recuperado el 01 de 09 de 2014, de http://bibliotecas.unam.mx/index.php/numeralia-2013

Universidad Nacional de Colombia. (s.f.). *HTTP://WWW.UNALMED.EDU.CO/~MSTABARE/DBMS.HTM*. Recuperado el 22 de 01 de 2014, de HTTP://WWW.UNALMED.EDU.CO/~MSTABARE/DBMS.HTM

Urbizagástegui, R., & Restrepo, C. (2011). La ley de Zipf y el punto de transición de Goffman en la indización automática. *Investigación bibliotecológica, 25*(54), 71-92.

Vacas Sáez, F. (1992). *Complejidad y tecnologías de la información.* Madrid: Instituto tecnológico bull. Obtenido de http://www.gsi.dit.upm.es/~fsaez/intl/libro_complejidad/15-el-desarrollo-del-software.pdf

Web incertidumbre. (s.f.). Recuperado el 18 de 02 de 2014, de HTTP://BLOGINCERTIDUMBRE.WORDPRESS.COM/MEDICION/DISENOS-ESTADISTICOS/

Wiki ALEPH. (2014). *http://wikialeph.wikispaces.com/ALEPH*. Recuperado el 29 de 08 de 2014

Wikipedia. (21 de 01 de 2015). *wikipedia. La enciclopedia libre*. Recuperado el 27 de 02 de 2015, de http://es.wikipedia.org/wiki/Dato

Winkler, D., Biffl, S., & Bergsmann, J. (2013). Software Quality Increasing value in software and systems development. En Springer (Ed.), *5th international conference, SWQD 2013 .* Vienna, Australia. Recuperado el 01 de 09 de 2015

ANEXO 1. MANUAL DE USUARIO.

MANUAL DE USUARIO

Elaborado por: Christian Daniel Enciso Padilla.

Manual que permite un correcto uso del sistema para la extracción de descriptores y términos multipalabra.

Introducción:

En el presente documento, se plantean los pasos a seguir para acceder, seleccionar un archivo, cargar el archivo, leer los resultados arrojados por el sistema y navegar por las diferentes opciones y ventanas del sistema. Se incluye también una sección de los requerimientos para poder ejecutar el sistema y un glosario en el que se describen los términos que pudieran llegar a causar confusión o que una interpretación diferente pueda afectar al funcionamiento del sistema.

Análisis y requerimientos del sistema:

- Computadora con Conexión a internet.
- El sistema debe ser accedido desde un navegador web actualizado (Chrome, Internet Explorer, Mozilla Firefox, Safari, Opera, etc.).
- IP o URL de acceso al sistema proporcionada por el administrador del mismo.

Funcionamiento:

El sistema cuenta con una interfaz de dos estados.

El primer estado de la interfaz es un formulario en el que se puede seleccionar el archivo a procesar como se muestra en la ilustración 0.1.

ILUSTRACIÓN 0.1 Interfaz de usuario, primer estado.

Al presionar el elemento “Seleccionar archivo” (A), se abrirá una ventana en la cual, el usuario debe dirigirse hasta el directorio en el que se encuentre el archivo que desea procesar y seleccionar haciendo doble *clic* sobre el archivo o seleccionándolo y presionando el botón “Abrir” (B). Una vez que el usuario selecciona un archivo y presiona el botón “Enviar” (C), el sistema intentará calcular (con base en el tamaño del archivo) el *tiempo estimado* del proceso. Apareciendo una ventana mostrando el tamaño del archivo seleccionado y el tiempo estimado en terminar el proceso. Presionar el Botón “Aceptar” (D). Véase la ilustración 0.2.

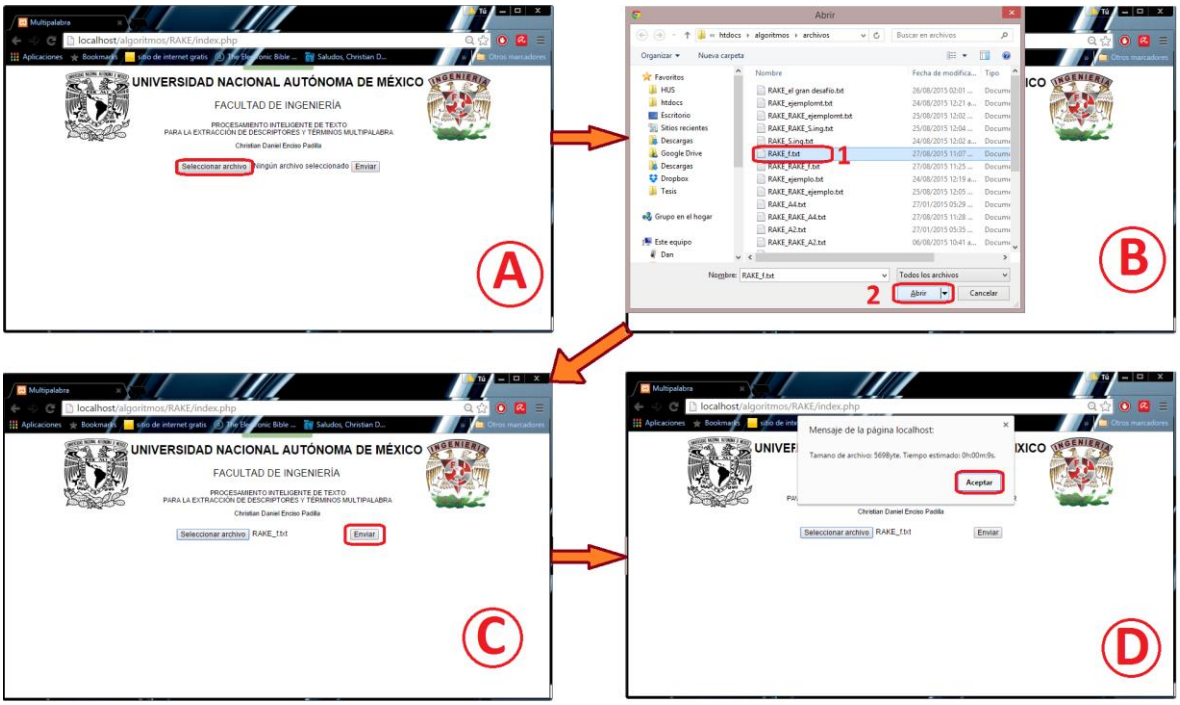


ILUSTRACIÓN 0.2 Interfaz de usuario, primer estado: selección y envío de archivo.

El segundo estado de la interfaz se presenta una vez que el usuario ha enviado el archivo y el sistema ha finalizado el procesamiento del mismo. Es en este momento cuando el sistema muestra los resultados.

En la primera pantalla, se muestra un listado con las palabras más frecuentes encontradas en el archivo, clasificadas según el valor calculado (coocurrencia) para la palabra/frase. Es la ventana de palabras con frecuencia mayor que 1 y se muestra en la Ilustración 0.3.

ILUSTRACIÓN 0.3 Listado de palabras con frecuencia mayor que 1

La segunda pantalla muestra un listado con una tercera parte del total de N-términos en la base de datos, ordenados por el valor de su coocurrencia como frase en el texto. Véase la ilustración 0.4

Consecutivo	Compuesto	veces	Valor calculado
1	MINIMAL GENERATING SETS	1	8.6667
2	LINEAR DIOPHANTINE EQUATIONS	1	8.5
3	MINIMAL SUPPORTING SET	1	7.6667
4	MINIMAL SET	1	4.6667
5	LINEAR CONSTRAINTS	1	4.5
6	NATURAL NUMBERS	1	4
7	STRICT INEQUATIONS	1	4

ILUSTRACIÓN 0.4 N-términos ordenados por valor calculado.

La tercera pantalla, muestra un listado similar al presentado en la segunda pantalla, también de N-términos ordenados según su coocurrencia, con la diferencia de que se calcula un nuevo punto de transición de Goffman, dependiendo del número de N-términos que aparece en el listado. Es notoriamente diferente cuando de documentos más extensos se trata. La ventana propuesta es de un vecindario que varía dependiendo el nuevo punto de transición, ya que en color verde mostrará aquellos N-términos cuyo valor calculado sea cercano al de punto de transición, para irse alejando y pintar en amarillo aquellos N-términos que se alejan en al menos una unidad al nuevo punto de transición. Véase la ilustración 0.5

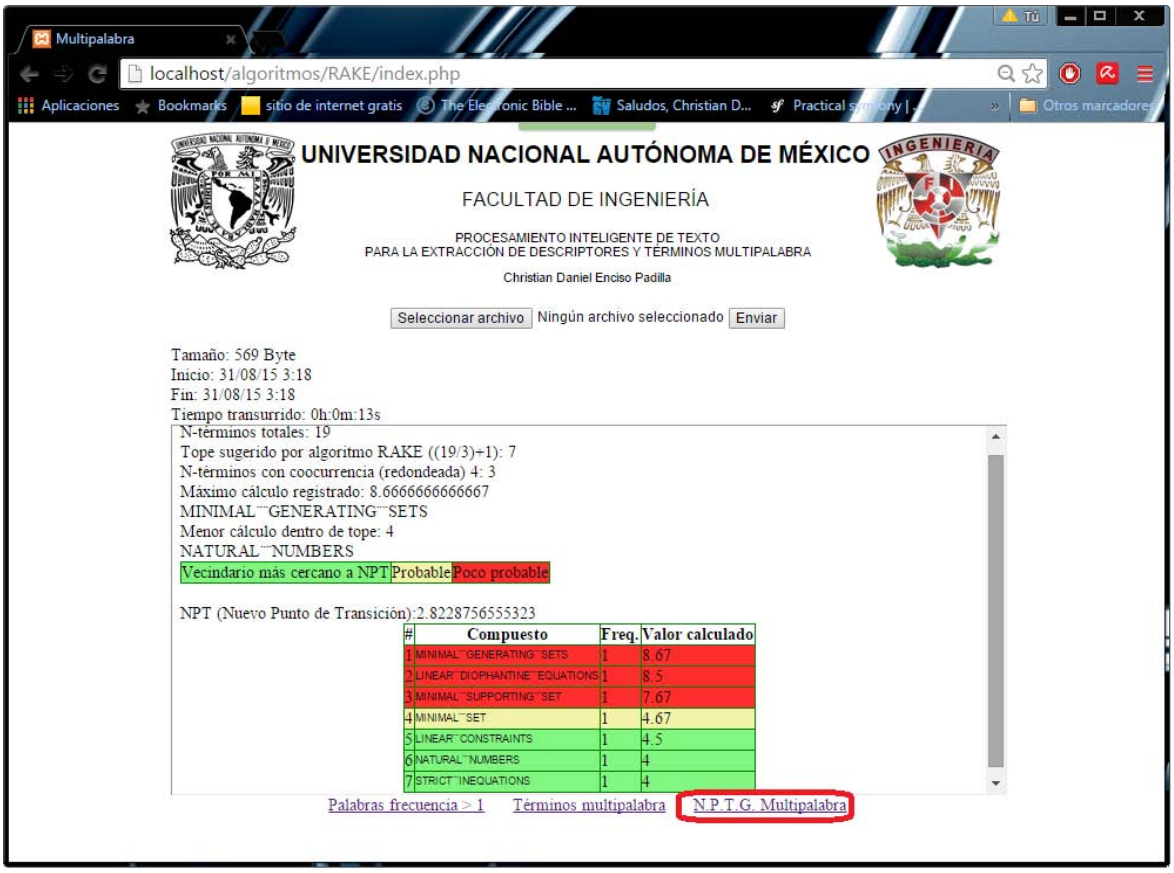


ILUSTRACIÓN 0.5 N-términos ordenados por valor calculado y marcados por cercanía al N.P.T.

ANEXO 2. MANUAL DE ADMINISTRACIÓN.

MANUAL DE ADMINISTRACIÓN

Elaborado por: Christian Daniel Enciso Padilla.

Manual que permite una correcta implementación, administración y mantenimiento del sistema para la extracción de descriptores y términos multipalabra.

Introducción:

Para la implementación del sistema, se debe tener conocimientos básicos de un servidor web, desarrollo PHP y bases de datos. MySQL en este caso.

De no ser así, es recomendable instalar XAMPP (ya sea para servidor Linux o Windows) ya que contiene los paquetes necesarios para el correcto funcionamiento del sistema.

Requerimientos:

Software:

- Sistema operativo Linux/Windows xp o superior:
- apache 2.0
- PHP 5.0.0 o superior
- MySQL 5.5 o superior

Hardware:

- Espacio suficiente para almacenar los archivos que los usuarios subirán. (Al menos 2 GB)
- Dependiendo del archivo a procesar (1 por vez) al menos 200MB de espacio para almacenamiento de la base de datos.
- Al menos 1GB de RAM

Deben crearse tres directorios dentro de la ruta para la web pública que se desee acceder (en Linux, por defecto es /var/www/ mientras que en Windows se configura en C:\directorioWeb esta ruta es modificable en el archivo de configuración httpd.conf).

El primer directorio es donde se encuentran los archivos con extensión .php (index.php, proceso.php, salida1.php, salida2.php y salida3.php), el archivo de configuración xml (config.xml).

El segundo directorio img es donde se encuentran las imágenes para la carátula del

Sistema.

El tercer directorio es el sitio donde se almacenan los archivos que se han cargado al sistema. Si se realiza instalación sobre Linux, el usuario www-data debe tener privilegios sobre este directorio.

Instalación:

Para la implementación del sistema, debe crearse una base de datos en el servidor MySQL, la cual debe poder ser accedida con el usuario y contraseña registrados en el archivo de configuración (config.xml).

El sistema fue desarrollado siguiendo el modelo de desarrollo clásico: en cascada debido a la simplicidad del mismo y a que los requerimientos del sistema permitían un desarrollo simple y eficaz del mismo.

El lenguaje de programación utilizado para el desarrollo es PHP debido a su flexibilidad y facilidad de implementación/mantenimiento.

PHP permite tanto el desarrollo estructurado como el desarrollo orientado a objetos. El sistema se desarrolló en paradigma estructurado a excepción de llamadas a ciertos objetos como lo referente a base de datos, lectura a configuración, etc.

El diseño del sistema se incluye en la tesis, ver referencia a la ilustración 5.4 para mayores referencias.

Descripción de archivos:

config.xml

Contiene la configuración de base de datos (servidor, nombre de la base de datos, usuario, contraseña) y un parámetro de precisión para el criterio de selección de keywords.

index.php

Es la página principal del sistema, interfaz que permite al usuario seleccionar un archivo que posteriormente será enviado mediante un formulario al archivo proceso.php y por último, se muestran los resultados.

proceso.php

Es el script encargado de subir al servidor el archivo, procesarlo, cargarlo en base de datos, realizar los cálculos pertinentes para las palabras/frases contenidas en el archivo seleccionado y actualización de valores por frase.

salida*n*.php

Existen tres archivos de salida (salida1.php, salida2.php, salida3.php) Estos archivos se parecen entre sí, a diferencia de que

salida1.php

Es el archivo encargado de mostrar las palabras más frecuentes en calidad de unitérmino. Es el primer archivo invocado por el index.php una vez que proceso.php ha terminado de realizar los cálculos y cargas a la base de datos pertinentes para el archivo procesado.

salida2.php

Es el archivo que muestra la salida original del algoritmo RAKE (1/3 de los N-términos totales en la base de datos).

salida3.php

Es el archivo que muestra las modificaciones realizadas al algoritmo RAKE, incluida la ventana variable de selección de palabras relevantes para el tercio propuesto en el algoritmo original.

ANEXO 3. CÓDIGO DESARROLLADO.

Nombre del archivo: Index.php

Tamaño del archivo: 4 kB

Total de líneas: 127

Descripción: Interfaz de usuario mediante el cual se puede cargar un nuevo archivo y navegar entre las diferentes opciones de salida que entrega el sistema.

En general, el código mostrado es una copia del original, con la salvedad de algunas líneas que fueron intencionalmente omitidas para proteger el código original, el sistema y los equipos con los que el equipo de desarrollo comparte red.

```
<?php
session_start();
?>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<html>
<head>
<title>Multipalabra</title>
<script>
function calc()
{
	/*calcular tamaño y tiempo de procesamiento*/
	}
}
</script>
</head>
<body>
<?PHP
date_default_timezone_set("America/Mexico_City");
	function display_filesize($filesize){

		/*Imprime PHP tam file*//
	}
?><center>
	<table border=0>
		<tr>
			<td>
				<img src="../img/UNAM.png" alt="Por mi raza hablar&aacute; el
esp&iacute;ritu" height="130" width="120">
			</td>
			<td>
				<center><span		style="font-family:Helvetica;font-
size:22px;font-style:normal;font-weight:bold;text-decoration:none;text-
transform:uppercase;color:000000;">Universidad	Nacional	Aut&oacute;noma	de
M&eacute;xico</span>
```

```
                        <br><br><span             style="font-family:Helvetica;font-
size:18px;font-style:normal;text-decoration:none;text-
transform:uppercase;color:000000;">Facultad de Ingenier&iacute;a</span>
                        <br><br><span             style="font-family:Helvetica;font-
size:12px;font-style:normal;text-decoration:none;text-
transform:uppercase;color:000000;">Procesamiento   inteligente   de   texto<br>para   la
extracci&oacute;n de descriptores y t&eacute;rminos multipalabra</span></center>
                </td>
                <td>
                        <img     src="../img/FI.png"     alt="Vino     mujeres     y
org&iacute;a.<br>Vino mujeres y org&iacute;a" height="130" width="120">
                </td>
        </tr>
        <tr>
                <td>
                </td>
                <td align="center">
                        <span      style="font-family:Helvetica;font-size:12px;font-
style:normal;color:000000;">Christian Daniel Enciso Padilla</span>
                </td>
                <td>
                </td>
        </tr>
        <tr>
        </tr>
        <tr>
                <td>
                </td>
                <td align="center">
                        <br>
                        <FORM           method="post"           action="proceso.php"
ENCTYPE="multipart/form-data" onsubmit="return calc()" />
                                <INPUT type=file name="archivo" id="file">
                                <INPUT type=submit value="Enviar" name="subir" >
                        </FORM>
                </td>
                <td>
                </td>
        </tr>
        <tr>
                <td colspan=3>
<?php
if (/*si ya se procesó un archivo*/)
{

        echo "Tama&ntilde;o: ".display_filesize($_SESSION["size"]);
        echo "<br>Inicio: ".$_SESSION["st"];
        echo "<br>Fin: ".$_SESSION["nd"];
        $transc=$_SESSION["start"]->diff($_SESSION["end"]);
        $t=$transc->format("%hh:%im:%ss");
        ?>
        <iframe width="100%" height="350px" src="salida1.php" name="iframe_a"></iframe>
        <center>
                <a href="salida1.php" target="iframe_a">Palabras frecuencia > 1</a>
                    
                <a href="salida2.php" target="iframe_a">T&eacute;rminos multipalabra</a>
                    
                <a href="salida3.php" target="iframe_a">N.P.T.G. Multipalabra</a>
        </center>
        <?php
}
```

```
?>
</td>
                </tr>
        </table>
        </center>
        </table>
</body>
</html>
```

Nombre del archivo: proceso.php

Tamaño del archivo: 15 kB

Total de líneas: 383

Descripción: Script encargado de carga de archivos, procesamiento, lógica del sistema, ingresos y actualización a base de datos.

En general, el código mostrado es una copia del original, con la salvedad de algunas líneas que fueron intencionalmente omitidas para proteger el código original, el sistema y los equipos con los que la computadora de desarrollo comparte red.

```
<?php
session_start();
?>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<?PHP
date_default_timezone_set("America/Mexico_City");
if (!$variableImportante)
        {
        ob_start();
  header("refresh: 5; url = index.php");
    echo 'Espere un momento y será redireccionado...';
ob_end_flush();
}
else
{
        if ($_FILES['archivo']['error'])
        {
                /*Mostrar error encontrado*/
                echo "<center><a href ='index.php'>Regresar</a></center>";
        }
        else
        {
                $_SESSION["st"]=date("d/m/y G:i");
                $_SESSION["start"]= new DateTime("now");
                $ruta_destino = ruta relativa
                echo"Tipo de arachivo: ".$_FILES['archivo']['type'];
                if($_FILES['archivo']['type']=="text/plain")//Tipo de archivo permitido
                {
                        move_uploaded_file($_FILES['archivo']['tmp_name'],
$ruta_destino."RAKE_".$_FILES['archivo']['name']);
                        $archivo=$ruta_destino."RAKE_".$_FILES['archivo']['name'];
                        $palabras=null;
                        $veces=null;
                        $N=0;//tamaÃ±o mayor de candidato
                        /*procesar archivo de stopwords*/

                        $xml=simplexml_load_file("config.xml")  or  die("Error:  Archivo  de
configuraci&oacute;n no encontrado");
                        $host=$xml->configuration[0]->host;
                        $user=$xml->configuration[0]->user;
                        $pass=$xml->configuration[0]->pwd;
                        $bd=$xml->configuration[0]->db;
```

```
                    $con=mysqli_connect($host,$user,$pass,$bd)or die ("Error en la base
de datos");
                    /*borrar tabla si existe*/
                    $borratabla="DROP table if exists rake;";
                    $res=$con->query($borratabla);
                    $borratabla="DROP table if exists rake_m;";
                    $res=$con->query($borratabla);
                    $con->query("commit;");

                    /*crear tabla multiterm*/
                    $creatabla="create  table  rake_m  (id_rakem  int  PRIMARY  KEY
AUTO_INCREMENT, mt text,  veces int, calc double,escapes varchar (20));";
                    $res=$con->query($creatabla);
                    $creaindex="create index idx_rakem on rake (mt)";
                    $res=$con->query($creaindex);
                    $creaindex="create index idx_rakemc on rake (calc)";
                    $res=$con->query($creaindex);
                    $con->query("commit;");

                    /*crear tabla rake (uniterminos)*/
                    $creatabla="create table rake (word varchar(20) unique, freq int, deg
double,cooc double, deeg int);";
                    $res=$con->query($creatabla);
                    $creaindex="create index idx_rake on rake (word)";
                    $res=$con->query($creaindex);
                    $con->query("commit;");

                    $a=$total=0;
                    set_time_limit(0);
                    $b=0;

                    $contenido=strtoupper(file_get_contents($archivo,true));


                    /*depuración de caracteres de escape*/
                    /*fin depuración*/

                    $limite="../verbos.palabras";
                    /*eliminar palabras sin peso*/
                    $cand1=explode("¬¬",$contenido);
                    for ($i=0;$i<sizeof($cand1);$i++)
                    {
                           $pars=explode(" ",$cand1[$i]);
                           if (sizeof($pars)>1)
                           {
                                  for($j=0;$j<sizeof($pars);$j++)//para  cada  elemento
del token de varias palabras
                                  {
                                         if ($pars[$j]=="¬¬")//si la palabra es ¬¬ se
cambia por nada
                                                unset($pars[$j]);
                                         if (/*algo importante*/)
                                         {
                                                //¬¬
                                                $pars[$j]="¬¬";
                                         }
                                  }
                                  $cand2[$i]=implode(" ",$pars);
                           }
                           else//si es un elemento único
                           {
```

```
	$cand2[$i]=(in_array($cand1[$i],$arrlim))?"¬¬":$cand1[$i];
                        }
                }
                $cand1=$i=$pars=$j=null;
                $aux=null;
                $m=0;
                /*preprocesar el archivo*/
                /*fin de preproceso*/
                $cand2=$i=$str=null;
                for ($i=0;$i<sizeof($aux);$i++)
                {
                        /*limpiar de stopwords*/
                        /*Fin de stop words*/
                        /*carga en base de datos*/
                        /*fin de carga en base de datos*/

                }
                $pals="select * from rake;";
                $res=$con->query($pals);
                //¡Felicidades! ¡¡¡Encontraste un huevo de Pascua!!!
                /*cálculo de coocurrencia*/
                /*fin de cálculo de coocurrencia*/
                $candidate="select * from rake_m;";
                $ex=$con->query($candidate);
                while($mul=mysqli_fetch_array($ex))
                {
                        /*Nueva carga de base de datos para coocurrencias*/
                        /*Fin de carga de coocurrencias en base de datos*/
                }
                $con->query("commit;");
                echo "<br>Fin: ".date("d/m/y G:i")."<br>";
        }
        else
        {
                header("refresh: 5; url = index.php");
                echo "Tipo de archivo no permitido";
        }
	}
	$_SESSION["end"]= new DateTime("now");
	$_SESSION["nd"]=date("d/m/y G:i");
	header("location:index.php");
}
?>
```

Nombre del archivo: salida1.php

Tamaño del archivo: 2 kB

Total de líneas: 33

Descripción: Script encargado de procesar los datos y enviar un listado de las palabras más frecuentes a la interfaz de usuario.

En general, el código mostrado es una copia del original, con la salvedad de algunas líneas que fueron intencionalmente omitidas para proteger el código original, el sistema y los equipos con los que la computadora de desarrollo comparte red.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<head>
<style>
</style>
</head>
<?PHP
date_default_timezone_set("America/Mexico_City");
$xml=simplexml_load_file("config.xml") or die("Error: Archivo de configuraci&oacute;n no
encontrado");
                     $host=$xml->configuration[0]->host;
                     $user=$xml->configuration[0]->user;
                     $pass=$xml->configuration[0]->pwd;
                     $bd=$xml->configuration[0]->db;
                     echo $host.":".$user.":".$pass.":".$bd."<br>";
                     $con=mysqli_connect($host,$user,$pass,$bd)or die ("Error en la base
de datos");
$mostrar="select * from rake order by cooc desc, freq where freq>1 desc";
                        echo       "<center><table       border=1><tr><td       colspan=4
align='center'><b>Palabras                con                frecuencia                >
1</b></td></tr><th>palabra</th><th>deeg</th><th>freq</th><th>coocurrencia</th>";
                        $ms=$con->query($mostrar);
                        while($muestra=mysqli_fetch_array($ms))
                        {
                                    echo
"<tr><td>".$muestra['word']."</td><td>".$muestra['deeg']."</td><td>".$muestra['freq']."</
td><td>".$muestra['cooc']."</td></tr>";
                        }
                        echo "</table></center>";

?>
```

Nombre del archivo: salida2.php

Tamaño del archivo: 5 kB

Total de líneas: 72

Archivo encargado de procesar los datos y mostrar el tercio propuesto por el algoritmo RAKE hacia la interfaz de usuario.

En general, el código mostrado es una copia del original, con la salvedad de algunas líneas que fueron intencionalmente omitidas para proteger el código original, el sistema y los equipos con los que el equipo de desarrollo comparte red.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<head>
<style>
</style>
</head>
<?PHP
date_default_timezone_set("America/Mexico_City");
$xml=simplexml_load_file("config.xml") or die("Error: Archivo de configuraci&oacute;n no
encontrado");
                    $host=$xml->configuration[0]->host;
                    $user=$xml->configuration[0]->user;
                    $pass=$xml->configuration[0]->pwd;
                    $bd=$xml->configuration[0]->db;
                    echo $host.":".$user.":".$pass.":".$bd."<br>";
                    $con=mysqli_connect($host,$user,$pass,$bd)or die ("Error en la base
de datos");

                       $T="select count(*) as pal from rake_m where veces=1";
                       $inf=$con->query($T);
                       $row=mysqli_fetch_array($inf);
                       $una=$row['pal'];
                       $T="select mt as pal,veces as veces from rake_m order by veces desc
limit 1";
                       $inf=$con->query($T);
                       $row=mysqli_fetch_array($inf);
                       $mayf=$row['veces'];
                       $mayp=$row['pal'];
                       $mostrar="select * from rake_m order by calc desc, veces desc;";
                       $ms=$con->query($mostrar);
                       $cant= mysqli_num_rows($ms);
                       $tope=round(($cant/3)+1);
                       $n=$cont=0;
                       $pttrans=round((-1+(sqrt(1+(8*$una))))/2);
                       echo "N-t&eacute;rminos totales: ".$cant;
                       echo     "<br>Tope     sugerido     por     algoritmo     RAKE
((".$cant."/".(3).")+1): ".$tope;
                       echo "<br>N-t&eacute;rminos con una aparici&oacute;n: ".$una;
                       echo  "<br>Punto  de  transici&oacute;n  de  Goffman  (<b>round((-
1+(sqrt(1+(8*".$una."))))/2)</b>):".$pttrans;
                       echo   "<br>M&aacute;xima   frecuencia   registrada:   ".$mayf."
(".$mayp.")<br>";
```

```
                        echo                                                    "<center><table
border=1><th>Consecutivo</th><th>Compuesto</th><th>veces</th><th>Valor calculado</th>";
$c=0;
                        while(($muestra=mysqli_fetch_array($ms))&&($n<$tope))
                        {
                                if ($muestra['veces']==1)
                                        $c++;

                                if
(($muestra['veces']>0)&&($muestra['calc']>0)&&(strlen($muestra['mt'])>2))
                                {
                                        $cont++;
                                        $color="bgcolor='#81F781'";

                                                                                $font="";
                                        echo  "<tr><TD  ".$color.">".$font.$cont."</TD><td
".$color.">".$font.$muestra['mt']."</td><td
".$color.">".$font.$muestra['veces']."</td><td
".$color.">".$font.round($muestra['calc'],4)."</td></tr>";
                                        $n++;
                                }
                        }
                        echo "</table></center>";
?>
```

Nombre del archivo: salida3.php

Tamaño del archivo: 6 kB

Total de líneas: 114

Descripción: Script encargado de procesar los datos y pasar una ventana de registros a la interfaz de usuario.

En general, el código mostrado es una copia del original, con la salvedad de algunas líneas que fueron intencionalmente omitidas para proteger el código original, el sistema y los equipos con los que el equipo de desarrollo comparte red.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<head>
<style>
</style>
<script>
/script>
</head>
<?PHP
date_default_timezone_set("America/Mexico_City");
$xml=simplexml_load_file("config.xml") or die("Error: Archivo de configuraci&oacute;n no
encontrado");
                     $host=$xml->configuration[0]->host;
                     $user=$xml->configuration[0]->user;
                     $pass=$xml->configuration[0]->pwd;
                     $bd=$xml->configuration[0]->db;
                     $con=mysqli_connect($host,$user,$pass,$bd)or die ("Error en la base
de datos");

                        $T="select count(*) as pal from rake_m where calc=1";
                        $inf=$con->query($T);
                        $row=mysqli_fetch_array($inf);
                        $una=$row['pal'];
                        $T="select mt as pal,calc as veces from rake_m order by calc desc
limit 1";
                        $inf=$con->query($T);
                        $row=mysqli_fetch_array($inf);
                        $mayf=$row['veces'];
                        $mayp=$row['pal'];
                        $mostrar="select count(1) from rake_m;";
                        $ms=$con->query($mostrar);
                        $cant= mysqli_fetch_array($ms);
                        $tope=round(($cant[0]/3)+1);
                         /*Seleccionar registros para cálculos de ventana variable*/
                        echo "N-t&eacute;rminos totales: ".$cant[0];
                        echo     "<br>Tope     sugerido     por     algoritmo     RAKE
((".$cant[0]."/".(3).")+1): ".$tope;
                        echo   "<br>N-t&eacute;rminos   con   coocurrencia   (redondeada)
".round($menc).": ".$cc;
                        echo     "<br>M&aacute;ximo     c&aacute;lculo     registrado:
".$mayf."<br>".$mayp."";
                                          echo "<br>Menor c&aacute;lculo dentro de tope:
".$menc."<br>".$menp."";
```

```
                                        $mm=round($menc);
                        echo   "<table><tr><td   bgcolor='#81F781'>Vecindario   m&aacute;s
cercano   a   NPT</td><td   bgcolor='#F2F5A9'>Probable</td><td   bgcolor='#FE2E2E'>Poco
probable</td></tr></table>";
                                        $c=$n=0;
                                        $ms="select * from rake_m order by calc desc,
veces desc limit ".$tope.";";
                                        $ms=$con->query($ms);


                                        $pttrans2=((-
$mm+(sqrt(($mm*$mm)+(8*$cc*$mm))))/(2*$mm))+2;
                        echo "<br>NPT (Nuevo Punto de Transici&oacute;n):".$pttrans2;
                                        $criterio=$xml->configuration[1]->param;
                                        do{

       $criterio=($pttrans2<$menc)?$criterio+0.1:$criterio;
                                        }while($criterio<($mayf/5));
                        echo                                                "<center><table
border=1><th>#</th><th>Compuesto</th><th>Freq.</th><th>Valor calculado</th>";
                        $cont=0;
                      while($muestra=mysqli_fetch_array($ms))
                              {
                                      $cont++;

       $color=((($muestra['calc']<=$pttrans2)&&($muestra['calc']>=$pttrans2-
$criterio))||(($muestra['calc']>$pttrans2)&&($muestra['calc']<=$pttrans2+$criterio)))?"bg
color='#81F781'":
                                                        ((((($muestra['calc']<($pttrans2-
$criterio))&&(($muestra['calc']>=($pttrans2-
(2*$criterio)))))||(($muestra['calc']>$pttrans2+$criterio)&&($muestra['calc']<$pttrans2+(
2*$criterio))))?"bgcolor='#F2F5A9'":

       (/*($cont<($pttrans2+($pttrans2/2)))?*/"bgcolor='#FE2E2E'"/*:"bgcolor='#FFFFFF'"*/
));
                                      echo       "<tr><td       ".$color.">".$cont."</TD><td
".$color."><span style='font-family:Helvetica;font-size:10px;'>";
                                      echo $muestra['mt'];
                                      echo                                      "</span></td><td
".$color.">".round($muestra['veces'],2)."</td><td
".$color.">".round($muestra['calc'],2)."</td></tr>";
                                      //echo      "<tr><TD      ".$color.">".$cont."</TD><td
".$color.">".utf8_decode($muestra['mt'])."</td><td
".$color.">".$muestra['veces']."</td><td ".$color.">".$muestra['calc']."</td></tr>";
                                      $n++;
                              }
?>
```

Nombre del archivo: config.xml

Tamaño del archivo: 1 kB

Total de líneas: 12

Descripción: Archivo que contiene los parámetros para el correcto funcionamiento del sistema.

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <configuration category="database">
    <host lang="en">host</host>
    <user>usuario</user>
    <pwd>password</pwd>
    <db>database</db>
  </configuration>
  <configuration category="params">
    <param lang="en-us">param</param>
  </configuration>
</config>
```

ANEXO 4 SISTEMA DE RECUPERACIÓN DE INFORMACIÓN

DIAGRAMA ORIGINAL **(LONDRES, INGLATERRA PATENTE Nº 3964029, 1976)**

Se presenta en inglés dado que es la imagen original como fue encontrado. En el capítulo 3 se muestra la traducción.