Universidad de las Ciencias Informáticas

Facultad 6

TÍTULO: Componente de software para la extracción de las palabras representativas en un documento.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Susana Gómez Mosquera

Tutor: Ing. Rafael L. Cardero Álvarez

Ciudad de La Habana, Cuba, Junio 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los __29__ días del mes de ___Junio___ del año __2011__.

Autor:	Tutor:
_____	_____
Susana Gómez Mosquera	Ing. Rafael L. Cardero Álvarez

AGRADECIMIENTOS

De manera general agradezco a toda mi familia por darme mucho cariño, por contribuir a mi educación y por apoyarme cada vez que lo necesité.

Agradezco a todos mis amigos que han convivido conmigo estos 5 años en la UCI, en especial a Lili, Rosy, Daya, Harlenia, Yaima, Yaimit, Elisa, Pedro, Frank y Liuver, que a pesar de que este es nuestro último año juntos siempre los voy a recordar.

A mi amiga Gretty que siempre ha estado a mi lado en todos los momentos de mi vida, eres una persona especial, te quiero flaqui.

A todas las personas que han contribuido de una forma u otra en mi formación profesional.

DEDICATORIA

Dedico esta investigación a mis abuelos Mimi y Papi por haberme dado todo lo que soy como persona, por su apoyo y dedicación, los quiero mucho.

A mis padres por haberme dado la vida, por hacer de mí la persona que hoy soy.

A mis hermanos que a pesar de fajarme con Yandy lo quiero mucho, al igual que a Yoander que lo quiero con el alma.

A mis tíos Roberto, René, Ricardo y mi tía querida Raquel y todos mis primos que son como si fueran mis hermanos: Yeny, Yudi, Kevin, Richard y Pedritín.

A mi abuela Caridad que siempre me guía por el mejor camino, y me da todo su amor, es que soy la nieta más linda que ella tiene (la única).

Y no se me puede olvidar a la última personita que llegó a la familia a mi sobrinita Isabella que la quiero cantidad.

A toda mi familia por confiar mucho en mí y por quererme tanto.

A mis amistades y a todas aquellas personas que siempre estuvieron ahí cuando los necesité.

DATOS DE CONTACTO

Ing. Rafael L. Cardero Álvarez: Graduado de Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas, en el año 2008. Instructor en la docencia. Trabaja en la Facultad 6 de la Universidad de las Ciencias Informáticas. Ha recibido varios cursos sobre procesamiento de imágenes digitales. Imparte en la universidad los cursos de pregrado “Procesamiento de imágenes digitales I” y “Procesamiento de imágenes digitales II”. Ha participado en varios eventos realizados en Cuba, vinculados al procesamiento de imágenes digitales. Sus intereses de investigación son el procesamiento de imágenes digitales (en general), el análisis de imágenes de documentos, la segmentación de textos en fotogramas de video, y la dirección de industrias de la informática y las comunicaciones.

Correo electrónico: rlcordero@uci.cu.

RESUMEN

En la actualidad existen millones de documentos disponibles para los usuarios, por lo cual se precisa de sofisticadas herramientas automatizadas de recuperación de documentos. Un aspecto clave de estas herramientas es la extracción de las palabras representativas de un documento.

La misión del proyecto Factoría de Software perteneciente al Departamento Señales Digitales es desarrollar componentes útiles para los demás proyectos de dicho departamento. En esta investigación se desarrolla un componente que extrae las palabras representativas de un documento. Elcomponentees capaz de leer un archivo de texto y extraerle las palabras representativas. En la investigación se abordan algunos métodos para extraer las palabras representativas y se justifica la elección del método por frecuencia de término como el adecuado para desarrollar el componente.

El componente se desarrolló utilizando el lenguaje de programación C++, la herramienta Qt Creator y el Proceso Unificado como metodología de desarrollo. Con el resultado de esta investigación se incrementa la cantidad de componentes disponibles en el Departamento y por tanto se aumenta la capacidad de respuesta del mismoante nuevos proyectos que necesiten extraer las palabras representativas de un documento.

PALABRAS CLAVES

Componente, Departamento Señales Digitales, extracción, Factoría de Software, palabra representativa.

Índice de figuras

Índice de tablas

Índice

INTRODUCCIÓN

Las nuevas Tecnologías de la Información y las Comunicaciones (TIC) han provocado un cambio profundo en los modos de trabajo de cualquier profesional y en cualquier labor que se realiza en la denominada sociedad de la información. Actualmente la demanda de información es inmensa y la generación de documentación ilimitada, haciendo que el sistema tradicional, se muestre físicamente incapaz de controlar tal cúmulo de datos.

La tendencia progresiva a la consulta, producción, almacenamiento, recuperación y difusión eminentemente digital de información, y al uso de las redes que posibilitan el acceso y gestión virtual de la documentación de cualquier ámbito profesional está provocando la aparición de una "sociedad documental". En ella se generan, comunican y localizan documentos en entornos digitales, que posibilitan en muchos casos el acceso a informaciones con contenidos.

No obstante, a más datos no necesariamente mejor información, sino más "ruido" informativo, entiéndase el ruido informativo por la acumulación de datos que no necesariamente presentan información válida en un contexto determinado. A mayor cantidad de información, mayor necesidad de búsqueda y sistematización documental, por tanto se precisa disponer de motores cada vez más inteligentes de tratamiento documental y de localización de informaciones contrastadas, y de profesionales cada vez más expertos en la localización, producción, recuperación y difusión de informaciones.

La creciente cantidad de textos disponibles y la imperiosa necesidad de información relevante por parte de los usuarios, hacen de la búsqueda de información un punto crítico en cualquier actividad investigativa. Para ayudar al usuario en esta tarea se han desarrollado poderosos motores de búsqueda que organizan la información disponible en base a categorías, títulos o contenido. Estas herramientas basan su funcionamiento en crear grandes índices de documentos, usados para resolver las consultas de los usuarios. La mayor parte de los documentos obtenidos como resultados a dichas consultas suelen ser poco relevantes para el usuario a pesar de cumplir los criterios de búsqueda especificados, por lo que hace que sea mucha información y poco tiempo para la búsqueda deseada. Actualmente existen nuevas herramientas que pueden brindar un enfoque diferente a la búsqueda de documentos, dando un mayor énfasis a las áreas de interés del usuario que requiere la información y obteniendo resultados más precisos.

El país no está exento a este desarrollo de la sociedad informativa. Hoy se trabaja en busca de una sociedad de la información que se base en el desarrollo vertiginoso de herramientas que faciliten al usuario una información más precisa a través de patrones de búsqueda. Existe en el país el Centro de Estudios de Reconocimiento de Patrones y Minería de Datos (CERPAMID), el cual está orientado hacia la investigación básica y aplicada en el área del Reconocimiento de Patrones (RP) y su aplicación a la Minería de Datos (MD) y Textos.

La Universidad de las Ciencias Informáticas (UCI), creada con el objetivo de impulsar el desarrollo de la sociedad de la información, en estos momentos no contiene proyectos que trabajen en el desarrollo de herramientas que puedan identificar las palabras representativas de un documento.

El proyecto Factoría de Software del departamento Señales Digitales surge por la necesidad de que en varios proyectos del mismo departamento utilizaban componentes diferentes que realizaban la misma funcionalidad, ejemplo de esto es el proyecto Video Vigilancia y el proyecto Sistema de Captura y Catalogación de Medias (SCCM), los cuales utilizaban el componente de captura de video. Dicho proyecto es una entidad dedicada al desarrollo de componentes de software, y tiene como objetivo poner en el mercado una infraestructura que garantice el acceso de diferentes entes a los componentes de software desarrollados por ella. Se desea perfeccionar los procesos implicados en la búsqueda de palabras representativas de un documento, para facilitar la búsqueda y así un mejor resultado de la misma.

A través de la problemática planteada anteriormente se define como **problema a resolver** ¿Cómo mejorar la situación del Departamento Señales Digitales para asumir proyectos en los que se requiera extraer las palabras representativas de un documento?

Se propone como **objetivo general** de la investigación desarrollar un componente software que extraiga las palabras representativas de un documento.

Para darle solución a este problema se define como **objeto de estudio** los métodos de extracción de rasgos.

Para el desarrollo de la investigación se necesita profundizar en el conocimiento de métodos que extraen palabras representativas de un documento, lo que hace que sea el **campo de acción** de la investigación.

Dando lugar a definir como **idea a defender** el desarrollo de un componente software que extraiga las palabras representativas de un documento, aumenta la cantidad de componentes disponibles en el Departamento Señales Digitales, y por tanto incrementa la capacidad de respuesta de dicho departamento ante nuevos proyectos.

Durante el desarrollo de esta investigación, y para dar respuesta al problema científico planteado, se aplicarán los siguientes **métodos científicos**:

Métodos teóricos:

Histórico-Lógico: mediante el cual se analizará la trayectoria y desarrollo del problema y cómo se pone de manifiesto la lógica interna de su desarrollo. Se aplica para analizar a nivel nacional e internacional, las empresas que utilizan software para el análisis de procesos de extracción de palabras de un documento y las características de otros sistemas informáticos para el análisis de procesos de extracción de palabras de un documento similares, así como investigaciones realizadas anteriormente sobre el tema.

Analítico-Sintético: se logrará desarrollar un estudio de la evolución que ha tenido el proceso de extracción de palabras de un documento a partir de su investigación en diferentes momentos históricos. Este método facilita el entendimiento del fenómeno en el que se trabaja, para de esta forma descubrir sus principales características.

Las **tareas de la investigación** que se proponen para lograr la realización de la misma son:

1. Definir el estado actual en Cuba y a nivel mundial respecto a la extracción de las palabras representativas de un documento, incluyendo tendencias y centros de investigación existentes.
2. Caracterizar las librerías que existen para leer documentos en formato TXT.
3. Caracterizar las librerías que existen para leer documentos en formato PDF.
4. Definir la arquitectura del componente para extraer las palabras representativas de un documento, incluyendo el diseño del algoritmo a utilizar.
5. Implementar la arquitectura especificada del componente para extraer las palabras representativas de un documento.
6. Validar el componente implementado.

El documento está estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica: Se realizará un estudio de las principales herramientas de extracción de palabras representativas tanto a nivel internacional como nacional. Además, se describen los lenguajes, las herramientas y metodologías a utilizar para el desarrollo del componente.

Capítulo 2: Propuesta y solución del componente: Se construirá el modelo del dominio, se realizará la especificación de los requerimientos funcionales y no funcionales, y el diagrama de casos de usos de sistema.

Capítulo 3: Implementación y Prueba: Se realizará la implementación y las pruebas del componente.

Capítulo 1: Fundamentación Teórica.

1 Introducción

El objetivo de este capítulo es abordar los aspectos investigativos para realizar el trabajo. En este capítulo se hará una investigación de los algoritmos que existen para extraer palabras representativas de un documento. Contiene además las tecnologías que se utilizarán para realizar el componente de software. Además de presentar métodos para calcular el peso de una palabra.

1.1 Conceptos asociados al dominio del problema

Palabra clave

Término significativo y representativo del documento que lo contiene. (1)

Componente de software

Un componente de software en tiempo de ejecución es un paquete dinámicamente vinculado con uno o varios programas manejados como una unidad y que son accedidos mediante interfaces bien documentadas que pueden ser descubiertos en tiempo de ejecución. (2)

Un componente de software es una parte modular de un sistema que encapsula implementación y un conjunto de interfaces.

Librería

Una librería es un conjunto de recursos (algoritmos) prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones. (3)

1.2 Algoritmos para la extracción de palabras representativas de un documento

1.2.1 Ponderado booleano y ponderado por frecuencia de término

El indexado denota la actividad de hacer el mapeo de un documento d_j en una forma compacta de su contenido. La representación más comúnmente usada para representar cada documento es un vector con términos ponderados como entradas. Es decir, un texto d_j es

representado como el vector $\overline{dj} = \langle Wij, \ldots, W|\tau|j\rangle$, donde τ es el diccionario, y w_{kj} representa cuanto el término t_k refleja el contenido del documento d_j. En ocasiones τ es el resultado de filtrar las palabras del vocabulario con respecto a una lista de palabras vacías (i.e. palabras frecuentes que no transmiten información, e.g. artículos y preposiciones) y un lematizador (i.e. obtener sus raíces morfológicas por aplicar un algoritmo de eliminación de afijos en una palabra de tal modo que aparezca sólo su raíz léxica. Con respecto al peso w_{kj} se tienen diferentes formas de calcularlo, entre las más usadas en la comunidad científica se tienen el ponderado booleano y ponderado por frecuencia de término, a continuación se presenta una breve descripción.

- **Ponderado booleano:** Consiste en asignar el peso de 1 si la palabra ocurre en el documento y 0 en otro caso

$$W_{kj} = \begin{cases} 1 & si \quad t_k \quad aparece \quad en \quad d_j \\ 0 & en \quad otro \quad caso \end{cases}$$

- **Ponderado por frecuencia de término:** Es asignar el número de veces que el término t_k ocurre en el documento d_j.(4)

$$W_{kj} = f_{kj}$$

1.2.2 Frecuencia de términos - frecuencia invertida de documentos (TF-IDF)

TF-IDF se basa en la observación de la frecuencia de aparición de los términos para el ámbito de cada uno de los documentos en particular y para el de la colección en su conjunto. La fórmula TF-IDF se puede expresar en los siguientes términos:

$$w_{ij} = f_{ij} * \log n / df_i$$

Donde w_{ij} es el peso otorgado al término $_i$ para el documento $_j$,f_{ij} es la frecuencia de aparición del término $_i$ en el documento $_j$, n es el número total de documentos y df_i es la frecuencia de aparición de $_i$ en los documentos de la colección (o lo que es lo mismo, el número de documentos que contienen dicho término).

La anterior fórmula se aplicará secuencialmente sobre cada término de cada documento de la colección, por lo que será necesario adecuar los valores **i** y **j** a todas estas situaciones.

Una vez se haya obtenido este valor se está en condiciones de representar los documentos mediante vectores, en los cuales la posición indica el término del léxico y el valor para dicha posición el peso para el elemento en cuestión. A partir de este punto se puede trabajar con las representaciones para generar las comparaciones que están en la base de los procesos que seguirán.

1.2.3 BM-25

BM-25 se basa en combinar cuatro fuentes de datos distintas: la frecuencia de aparición en la colección, la frecuencia de aparición en un documento, la longitud de un documento y dos parámetros configurables según las características específicas de cada colección.

Respecto a la frecuencia de aparición en la colección, se deriva de las mismas ideas que la frecuencia inversa de aparición en documentos de TF-IDF. Se expresa de forma ligeramente distinta, pero los efectos son similares.

$$CFW_i = log\ N - log\ n_i$$

Donde CFWi es el peso derivado de la frecuencia de aparición del término i en los documentos de la colección, N es el número total de documentos de la colección y n_i es el número de documentos que contienen el término i. Los logaritmos sirven únicamente para limitar la manera en que crecen los pesos, de manera que un término que aparece el doble que otro no reciba exactamente el doble de peso, sino sólo “algo más”.

La frecuencia de aparición en un documento es exactamente la misma que en el caso de TF-IDF.

$$TF_{ij} = nº\ de\ veces\ que\ i\ aparece\ en\ j$$

La longitud de los documentos es un dato a considerar si se estima que el hecho de que un término aparezca en un documento corto implica que éste término representa una mayor porción del conjunto de conceptos representativos de dicho documento que en el caso de que este sea largo, caso en el que la proporción que un solo término representa es menor. Para contextualizar la frecuencia de aparición de un término en un documento se puede utilizar la longitud normalizada de dicho documento.

$$NDL_j = DL_j / ADL$$

Donde NDLj es la longitud normalizada del documento j, DLj es la longitud sin normalizar del documento j y ADL la longitud media de los documentos de la colección.

Por último, existen dos parámetros que se deberían fijar tras pruebas exhaustivas con la colección. Aun así es posible hacer una idea previa de los valores aproximados que deberían tener observando las características de la colección.

El parámetro k1 permite moderar la importancia que tiene la frecuencia de aparición de un término en un documento. Dependiendo de la longitud de los documentos de la colección, puede ser aconsejable que k1=0, en el caso de las colecciones compuestas tan sólo por títulos y resúmenes, que k1=1 para colecciones con documentos cortos (como por ejemplo colecciones de noticias) o que k1=2 (o valores superiores) si se trata de colecciones con documentos largos, como colecciones de documentos técnicos y científicos. La idea detrás de este parámetro reside en que en una colección con documentos muy cortos la frecuencia de aparición de un término en un documento puede modificar de forma muy importante el peso de dicho término, mientras que en realidad los términos que aparecen en textos tan cortos normalmente tienen una importancia similar (muy alta). En los documentos largos las frecuencias de aparición de los términos no se derivarán probablemente de coincidencias accidentales, sino que reflejarán la importancia de dichos términos para representar los contenidos de los documentos.

El parámetro b sirve para introducir un factor de moderación sobre la longitud normalizada de los documentos. Se entiende que en ocasiones los documentos de una colección serán largos porque tratan sobre muchos temas diversos, mientras que en otras ocasiones son largos simplemente porque son repetitivos. Para evitar que esta repetición desvirtúe el dato de la longitud de los documentos se utiliza b=1 o un valor cercano a 1. En los casos en los que la longitud de los documentos está asociada a la diversidad de temas recogidos en el mismo se utilizará b=0 o un valor similar.

Combinando todas las fuentes de datos disponibles, se construirá una fórmula como la siguiente:

$$Wij = \frac{CFWi * TFij * (k1 + 1)}{k1 * ((1-b) + (b * NDLj)) + TFij}$$

Algoritmo seleccionado

Para el desarrollo del componente se utilizará el método por frecuencia de términoya que este se aplica a un sólo documento, y los demás métodos se utilizan cuando existe una colección de documentos. Téngase en cuenta que en el contexto relativo a esta investigación el procesamiento será en base a un documento. El interés consiste en extraer las palabras que representan a un documento de entrada de manera independiente. No es objetivo de la investigación desarrollar un componente que extraiga las palabras que mejor representan a un documento dentro de una colección de documentos.

No se escogió el método ponderado booleano ya que este no tiene en cuenta la cantidad de veces que aparece un término en el documento. Por tanto una palabra que aparezca mil veces tendrá igual relevancia que una palabra que aparezca una vez. Este comportamiento no es adecuado para desarrollar el componente.

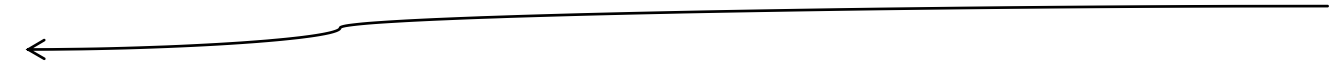
A continuación se emplea el método por frecuencia de término para un mejor entendimiento:

A manera de ejemplo, se presenta el siguiente fragmento sobre la Ingeniería de Software.

> *Ingeniería de software es la disciplina o área de la Ingeniería que ofrece métodos y técnicas para desarrollar y mantener software.*

El componente debe guardar el texto en una lista

Ingeniería	*de*	*software*	*es*	*la*	*disciplina*	*o*	*área*	*de*	*la*	*Ingeniería*	*que*	*ofrece*	*métodos*	*y*	*técnicas*



para	*desarrollar*	*y*	*mantener*	*software*	.

Luego se filtrarán las palabras vacías es decir las palabras que no trasmiten información, como son los artículos y preposiciones. Al realizar esta operación la lista quedaría de esta forma:

Ingeniería	*software*	*disciplina*	*área*	*Ingeniería*	*ofrece*	*métodos*	*técnicas*	*desarrollar*	*mantener*	*software*

Se tiene la lista con las palabras, ahora para ver cuáles son las más representativas se calcula su peso.

Se asigna el número de veces que el término t_k ocurre en el documento d_j.

Ingeniería	*software*	*disciplina*	*área*	*ingeniería*	*ofrece*	*métodos*	*técnicas*	*desarrollar*	*mantener*	*software*
1	***1***	*1*	*1*	***2***	*1*	*1*	*1*	*1*	*1*	***2***

Aquí las palabras representativas serían las de mayor peso, en este caso *Ingeniería* y *software* son las palabras representativas.

1.3 Análisis de soluciones existentes

Hoy en día en el mundo existen varias soluciones para la extracción de palabras representativas, a continuación se presentan algunas soluciones.

1.3.1 En el ámbito internacional

Keyrow

Keyrow es una herramienta muy simple, que sirve para extraer algunas palabras clave de un sitio web determinado. Aunque sólo sirve para esta tarea, arroja muchos resultados, y hace un buen trabajo analizando el contenido de una página.

Por lo regular se muestran los primeros 100 resultados (las primeras 100 keywords encontradas), sin embargo la verdadera utilidad de Keyrow viene cuando se ordena la información, ya que muestra información muy útil.(5)

Wordtracker

Wordtracker es una herramienta que sirve para sugerir palabras claves relacionadas con un tema y conocer su oferta y demanda. Está en inglés y tiene un costo por búsqueda sin embargo da la ventaja de trabajar con datos numéricos, es decir, arroja cifras de búsquedas reales, no sólo de volúmenes estimados. Además combina información de diferentes buscadores.(6)

Google Suggest

Google Suggest es una nueva herramienta que a medida que se escribe en el cajón de búsqueda va sugiriendo términos y palabras, los más buscados por los usuarios, crea diagramas que permite ver cómo se combinan términos de búsqueda. Cuando se hace una investigación de keywords, ésta no es la mejor herramienta, ya que es muy lenta y poco práctica.(7)

Ninguna de las herramientas anteriormente planteadas vale para el desarrollo del componente ya que son soluciones integrales. En otras palabras constituyen la integración de un conjunto de componentes. Uno de tales componentes es el que extrae las palabras representativas de un documento. Lamentablemente no se pudo conocer cómo es que dichas herramientas extraen las palabras claves de un documento. Se consideró importante mencionar estas herramientas ya que están relacionadas con el área de conocimiento de esta investigación.

1.4 Centros de Investigación existentes en Cuba

Centro de Estudios de Reconocimiento de Patrones y Minería de Datos

El CERPAMID está conformado por un grupo de profesores del Departamento de Computación de la Facultad de Matemática y Computación de la Universidad de Oriente con una calidad profesional asegurada con más de media década de experiencia. Realizan investigaciones que incluyen el desarrollo de algoritmos para el procesamiento y análisis de grandes volúmenes de información estructurada o textual. Imparten cursos de postgrado, y maestría en Ciencia de la Computación de la Universidad de Oriente en Reconocimiento de Patrones. (8)

Líneas de investigación

- ✓ Minería de textos.
- ✓ Algoritmos paralelos.
- ✓ Desarrollo de algoritmos de Reconocimiento de Patrones para los problemas de clasificación supervisada y no supervisada.

Centro de Aplicaciones de Tecnologías de Avanzada

El Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV) es un centro de investigaciones teóricas y aplicadas que tiene como misión fundamental asimilar, desarrollar e introducir en la práctica social los aspectos más novedosos de la Teoría y la Práctica del RP en su concepción más general, y de la MD, que permitan responder a las necesidades del progreso científico-técnico y socio-económico, así como incrementar el Patrimonio Científico Nacional. Las investigaciones incluyen en la actualidad el procesamiento digital de imágenes y señales, la teledetección, el reconocimiento lógico combinatorio de patrones, el reconocimiento sintáctico estructural, la teoría de testores, algoritmos conceptuales, análisis de texturas, la interpretación conceptual de datos espaciales, la minería de texto, la minería de datos mezclados, entre otras. Las aplicaciones están dirigidas a áreas tales como la biometría, la recuperación de información, la prospección geológica y procesamiento de información de texto.

El CENATAV se creó en el 2004 y está en proceso de consolidación. Además de su actividad de investigación teórica y aplicada, brinda servicios de información científico – técnica, consultoría, cursos de posgrado en RP y MD. (9)

Presenta tres áreas vinculadas directamente con la investigación, que se interrelacionan y complementan en cada uno de los proyectos que se desarrollan. Estas son:

- ✓ Reconocimiento de Patrones
- ✓ Minería de Datos
- ✓ Ingeniería en sistemas

Dichos centros fueron de ayuda en el desarrollo de la investigación pues se pudo contactar con personal de los centros y así solicitar información acerca de la extracción de palabras representativas de un documento, cuya información fue de mucha ayuda para la investigación.

1.5 Fundamentación de las tecnologías y herramientas propuestas a utilizar

1.5.1 Lenguajes o tecnologías de Programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. (10)

Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. (11) En la actualidad existen disímiles lenguajes de programación cada uno de ellos con características que lo distinguen, a continuación se exponen algunos de ellos como posible lenguajes a utilizar.

Java

Java es un lenguaje de programación y la primera plataforma informática creada por Sun Microsystems en 1995. Los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

Java es un lenguaje de programación orientado a objetos, de una plataforma independiente. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (12)

Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor WEB. Por lo general los applets son programas pequeños y de propósitos específicos. (13)

C Sharp

C# es un lenguaje moderno y altamente expresivo que se ajusta al paradigma de programación orientada a objetos. Su sintaxis es similar a C++ y Java. El lenguaje fue desarrollado en gran parte por Anders Hejlsberg (creador del mítico compilador Turbo Pascal y uno de los diseñadores del lenguaje de programación Delphi). (14)

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por

la ECMA [1]e ISO[2]. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes. C# fue diseñado para combinar el control de lenguajes de bajo nivel como C y la velocidad de programación de lenguajes de alto nivel como Visual Basic. (15)

C# es un lenguaje que toma las mejores características de otros lenguajes como Java o C++, y este las combina en uno solo.

C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de **C**, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.(16)C++ contiene muchas mejoras y características que sencillamente lo convierten en C mejor, independientemente de la programación orientada a objetos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel.

Lenguaje seleccionado

C++ es un lenguaje muy eficaz en cuanto a rapidez y uso de memoria en las aplicaciones que se obtienen. Es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones.

Brinda la posibilidad de trabajar con IDEs[3] de programación estandarizados y calificados en sistemas operativos libres, esto hace que se elimine el uso del lenguaje C# por ser nativo de la plataforma propietaria .NET. Por otra parte se encuentra Java, su principal inconveniente es la importación de máquinas virtuales para poder ejecutar sus sentencias, lo que implica una compilación más lenta.

[1]European Computer Manufacturers Association

[2] International Organization for Standardization

[3]Integrated Development Environment en español Entorno de Desarrollo Integrado.

Se selecciona C++ como el lenguaje a utilizar en el desarrollo del componente pues brinda la posibilidad de que algoritmos cuyo pseudocódigo se encuentra ya desarrollado en C++, se pueda utilizar y amoldarlo a la solución. Dicho IDE está difundido mundialmente y cuenta con una gran comunidad de desarrollo.

1.5.2 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos, software. Estos van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. (17)

Programación Extrema (XP)

Esta metodología fue desarrollada por Kent Beck. XP es una de las metodologías ágiles más utilizadas, centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP está guiada por una rápida programación y se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, en la reutilización de código, en la realización de pruebas a los principales procesos con el objetivo de tratar de obtener los posible errores futuros, esto conocido como pruebas unitarias, en la simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Impone un alto nivel de disciplina entre los programadores, lo cual permite mantener un mínimo nivel de documentación que a su vez se traduce en una gran velocidad de desarrollo, además de proponer que el trabajo de los programadores sea en pares de forma tal que uno realice lo que el otro no hace en ese instante. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (18)

Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (RUP) es una metodología de desarrollo de software. Junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud

y diferentes tamaños de proyecto. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. (19)

RUP es un proceso de desarrollo de software, es un conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software. Es un marco genérico que puede especializarse para una variedad de tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyectos.(20)

Sus principales características son:

Dirigido por casos de uso: Los casos de uso guían el proceso de desarrollo, no se desarrollan aisladamente. Además guían el diseño, implementación y prueba son los que indican cómo debe actuar el sistema con el usuario final o con otro sistema para conseguir su objetivo.

Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño, lo cual constituye la arquitectura del producto a desarrollar. La arquitectura es una vista del diseño completo con las características más importantes resaltadas.

Iterativo e incremental: A medida que avanza el proceso de desarrollo se producen versiones incrementales, las cuales se acercan cada vez más al producto terminado. En cada iteración los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componente y verifican que los componentes satisfacen los casos de uso.

Beneficios que aporta RUP

- ✓ Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.
- ✓ Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.
- ✓ Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros.

- ✓ Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas.
- ✓ Unifica todo el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software.
- ✓ Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.(21)

Metodología de desarrollo de software seleccionada

Se escoge RUP como la metodología de software puesto queno necesita que el cliente forme parte del equipo de desarrollo al contrario de XP y en este caso es muy beneficioso ya que no hay un cliente específico para el sistema. La gran cantidad de artefactos que se generan en RUP contribuyen a un mejor entendimiento del sistema que se desea realizar. RUP es una metodología con alta adaptabilidad a las condiciones reales del desarrollo del sistema con lo que se puede hacer más ágil según se necesite.

1.5.3 Lenguaje Unificado de Modelado (UML)

RUP utiliza como lenguaje para el modelado al UML.

El Lenguaje Unificado de Modelado (UML) es un lenguaje para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas. Representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos. (22)

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- ✓ Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- ✓ Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.4 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, del español Ingeniería de Software Asistida por Computadoras) modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía. El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones, y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía. (23)

Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Además cuenta con una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones, dándole una peculiaridad sobre el resto de las herramientas de este tipo.

Características principales

- ✓ Soporte de UML versión 2.1.
- ✓ Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- ✓ Modelado colaborativo con CVS y Subversión (control de versiones).
- ✓ Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI.
- ✓ Ingeniería de ida y vuelta.
- ✓ Ingeniería inversa - Código a modelo, código a diagrama.
- ✓ Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- ✓ Generación de código - Modelo a código, diagrama a código.

- ✓ Editor de detalles de casos de uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Diagramas EJB - Visualización de sistemas EJB.
- ✓ Generación de código y despliegue de EJB - Generación de beans para el desarrollo y despliegue de aplicaciones.
- ✓ Diagramas de flujo de datos.
- ✓ Soporte ORM - Generación de objetos Java desde la base de datos.
- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- ✓ Generador de informes.
- ✓ Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- ✓ Importación y exportación de ficheros XMI.
- ✓ Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de Microsoft Visio.
- ✓ Editor de figuras.(24)

Rational Rose

Rational Rose es una de las herramientas más potentes de modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de construirlo. Cubre todo el ciclo de vida de un proyecto: concepción y familiarización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases.Proporciona mecanismos para realizar la ingeniería inversa, es decir, que a partir del código se pueda obtener información sobre su diseño; adicionalmente permite generar código en diferentes lenguajes a partir de un diseño en UML. Brinda la posibilidad de que varias personas trabajen a la vez, permitiendo que cada desarrollador opere en un

espacio de trabajo privado que contiene el modelo completo y permite que tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.(25)

Herramienta CASE seleccionada

Se selecciona Visual Paradigm como la herramienta CASE a utilizar ya que soporta todo el ciclo de vida del desarrollo del software, además es una herramienta que ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Además de que Visual Paradigm presenta una licencia pública, descartando Rational Rose por presentar licencia privativa.

1.5.5 IDE de desarrollo

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (26)

Qt Creator

Qt Creator fue creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt. Es multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Qt Creator utiliza el lenguaje de programación C++ de forma nativa.

Qt está disponible bajo 3 diferentes licencias:

- **Qt GNU GPL v. 3.0:** Aplicación de código abierto, los cambios realizados al código fuente de Qt deben ser compartidos con la comunidad.
- **Qt GNU LGPL v. 2.1:** Es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de Qt deben ser compartidos con la comunidad.
- **Qt Commercial Developer License:** Es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de Qt pueden mantenerse cerrados.(28)

Principales características de Qt Creator:

- Posee un avanzado editor de código C++.
- Además soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee también una GUI(Graphical User Interface) integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.(27)

KDevelop

KDevelop es un entorno de desarrollo integrado para sistemas Linux y otros sistemas Unix, publicado bajo licencia GPL, orientado al uso bajo el entorno gráfico KDE (K Desktop Environment), aunque también funciona con otros entornos, como Gnome.

A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio, por lo que depende de gcc[4]para producir código binario.(29)

IDE seleccionado

Se selecciona Qt Creator como IDE de desarrollo ya que utiliza C++, lenguaje de programación seleccionado para el desarrollo del componente, además que provee un API[5]sencilla y divertida de utilizar, permite que los desarrolladores tengan una alta productividad y ofrece herramientas potentes y sencillas. Además viene acompañado de un conjunto de herramientas para facilitar su uso.

[4] GNU Compiler Collection: compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.

[5]Application Programming Interface

1.6 Formatos PDF y TXT

Formato PDF

El formato PDF (Portable Document Format) es uno de los más utilizados por sus características compuestas de mapas de bits, imágenes vectoriales y texto. Es un formato que permite la fidelidad del diseño original de un documento.

Entre las características del formato PDF es que es multiplataforma, permite integrar cualquier tipo de elementos y lo más interesante es que es una especificación abierta con alternativas de software libre.

Formato TXT

El formato TXT identifica archivos que contienen solo texto es decir son aquellos que están compuestos únicamente por texto sin formato, sólo caracteres.

Debido que para leer un archivo .txt es fácil de realizar, no se utilizará ninguna librería para poder leer dicho archivo. Existe la clase QTextStream la cual se utilizará para el desarrollo del componente a realizar ya que proporciona funciones básicas para leer y escribir texto.

1.7 Conclusiones

En este capítulo se realizó un estudio de métodos que calculan el peso de una palabra, seleccionando el método por frecuencia de término debido a que es el mejor que se ajusta a las necesidades existentes. Para la realización del sistema se decidió utilizar como metodología de desarrollo RUP, apoyándose en la herramienta CASE Visual Paradigm. También se decidió que el lenguaje más factible para la programación del sistema es el C++, y el IDE QT Creator.

Capítulo 2: Propuesta de solución.

2 Introducción

En este capítulo se hace referencia a la presentación de la solución propuesta con el objetivo de entender el problema en la investigación presente. Se presenta el modelo de dominio con la definición de cada una de sus clases, se mencionan los requerimientos funcionales y no funcionales. Posteriormente se realiza la descripción de los actores y los casos de uso del sistema para el desarrollo del diagrama de casos de uso del sistema.

2.1 Modelo del dominio

El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto. Es decir, un diagrama con los objetos que existen (reales) relacionados con el proyecto que se va a acometer y las relaciones que hay entre ellos.

Definición de las clases del Modelo del Dominio

Componente de software: Parte modular de un sistema que leerá el documento de texto y extraerá las palabras representativas.

Documento de texto: Consiste en un archivo para salvar información.

Palabra clave: Palabra representativa del documento.

Diagrama del Modelo del Dominio

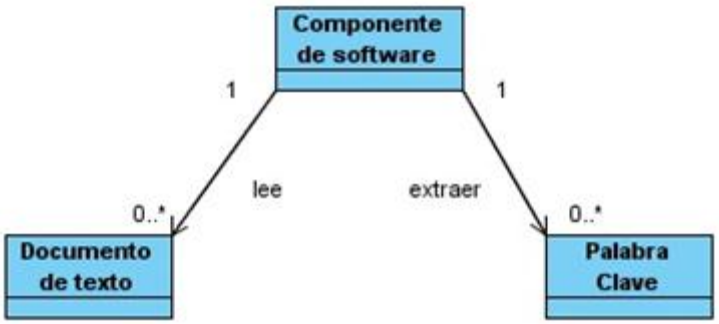


Figura 1: Diagrama de clases del modelo del dominio.

Descripción del modelo de dominio

En la figura anterior el componente debe ser capaz de leer un documento de texto para luego extraer las palabras claves del texto seleccionado.

2.2 Requerimientos funcionales

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.(30)

Un requerimiento funcional no es más que condiciones que el sistema debe cumplir.

RF1 Extraer palabras representativas

1.1 Extraer las palabras representativas de un documento de texto en formato txt.
1.2 Extraer las palabras representativas de un documento de texto en formato pdf.

2.3 Requerimientos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.(30)

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

RNF1 Portabilidad

El componente debe ser multiplataforma, siendo así compatible con cualquier tipo de sistema operativo.

2.4 Descripción del sistema propuesto. Modelos de casos de uso del Sistema.

2.4.1 Descripción de los actores

Un actor del sistema es quien interactúa o hace uso del sistema, representa a un ser humano, a un software o a una máquina que interactúa con el sistema. Participa en uno o más casos de uso y juega un papel por cada caso de uso con que colabora. Un actor inicia un caso de uso pero una vez que este ha comenzado el caso de uso puede interactuar con varios actores.

Tabla 1: Descripción de los actores del sistema.

Actor	**Descripción**
Llamador	Es el software que invocará al componente, para extraer las palabras representativas de un documento.

2.4.2 Casos de uso del sistema

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.(31)

Tabla 2: Actor de sistema.

CU-1	Extraer palabras representativas
Actor	Llamador
Descripción	El Llamador extrae las palabras

	representativas del documento.
Referencia	RF1

2.4.3 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

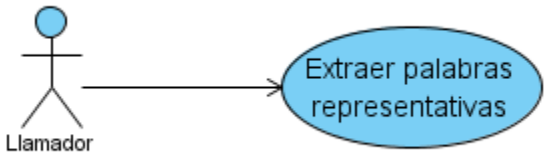


Figura 2: Diagrama de casos de uso del sistema.

2.4.4 Casos de uso expandidos

2.4.4.1 Descripción del CU Extraer palabras representativas

Tabla 3: Descripción del caso de uso del sistema.

Caso de Uso:	Extraer palabras representativas
Actor:	Llamador
Resumen:	Extrae las palabras representativas del documento seleccionado.
Precondiciones:	
Referencias	RF1
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Extraer palabras representativas"	
Acción del Actor	**Respuesta del Sistema**

1. Invoca al componente utilizando la interfaz que permite extraer las palabras representativas de un documento.	2. Comprobar que existe el archivo. 3. Lee el texto contenido en el archivo. 4. Extrae las palabras representativas del texto leído. 5. Retorna las palabras representativas del texto.

2.5 Análisis y diseño

2.5.1 Descripción de la arquitectura

La arquitectura representa la estructura de los componentes de un programa o sistema, sus interrelaciones, los principios y reglas que gobiernan su diseño y evolución en el tiempo.

Para el desarrollo del componente se decidió utilizar la arquitectura en capas, definiéndose la utilización de dos capas.

Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.
Para un mejor diseño del componente se tuvo en cuenta el estudio de varios patrones como:

Los patrones GRASP (Patrones de Software para la asignación General de Responsabilidad), dichos patrones son parejas de problema - solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describe los principios de la asignación de responsabilidades de un objeto.

Patrón Creador
El cual plantea la necesidad de asignarle a una clase la responsabilidad de crear una instancia de otra clase siempre y cuando agregue los objetos de la clase, los contenga, registre las instancias de estos objetos y los utilice específicamente.

Patrón Bajo acoplamiento

Acoplamiento bajo significa que una clase no depende de muchas clases, ya que uno de los principios para protegerse frente a los cambios es mantener bajo el acoplamiento entre variedades. Resulta evidente que, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios.

Patrón Experto

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

Alta Cohesión

Expresa que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta, la cohesión no es más que la medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. El patrón de Alta Cohesión, más que un diseño directamente implementable en código, se trata de un principio director que guiará el diseño. Una clase estará más cohesionada cuanto más enfocado sea su comportamiento. Es decir, al asignar responsabilidades en el diseño, se buscará soluciones que asignen los métodos a las clases de forma coherente, completa y relacionada. De esta forma, se obtendrá clases cohesionadas. (32)

Inyección de dependencia

Permite crear instancias de objetos y configurarlas, aislando las dependencias entre los mismos mediante la inyección de las instancias de otros objetos como parámetros de los objetos.

2.6 Análisis y diseño del caso de uso “Extraer palabras representativas”

En esta sección se presentan los diagramas de análisis y diseño correspondientes al caso de uso “Extraer palabras representativas”.

2.6.1 Diagrama de clases de análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. (33)

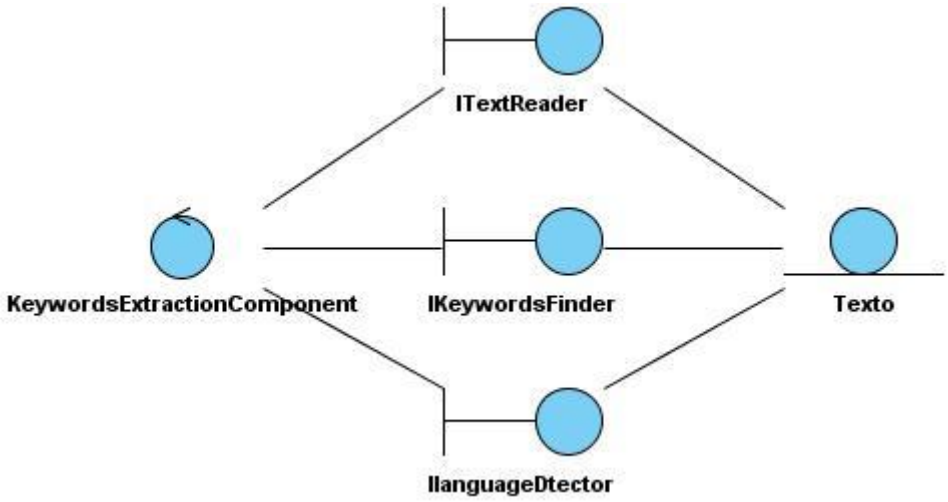


Figura 3: Diagrama de clases de análisis.

En la Figura 3 se muestra el diagrama de clases de análisis para el caso de uso Extraer palabras representativas, en él se aprecian las clases interfaces IKeywordsFinder, ITextReader e ILanguageDetector, además de la clase control KeywordsExtractionComponent la cual es la que maneja todos los pasos que se van a ejecutar para extraer las palabras representativas de un documento.

2.6.2 Diagrama de clases del diseño

Un diagrama de clases del diseño se utiliza para que el programador en cuestión pueda realizar una correcta implementación del sistema a desarrollar.

Definición de las clases del Diagrama del diseño

KeywordsExtractionComponent: Es la clase controladora que interactúa con las clases interfaces ITextReader y la clase IKeywordsFinder, es la encargada de retornar las palabras representativas del documento.

ITextReader: Con esta clase interfaz se podrá leer el documento según el tipo de formato que presente el texto, se retornará una lista con todas las palabras del texto. Esta clase presenta dos clases hijas que heredan de ella, la clase pdfTextReader y txtTextReader, que según el tipo de formato que presente el texto una de estas clases será la encargada de retornar la lista.

IKeywordsFinder: Esta clase será la encargada de retornar una lista con las palabras más significativas, es decir eliminará las palabras que no tengan un significado como son las preposiciones, las conjunciones. Presenta una clase hija nombrada FrecuencyBasedKeywordsFinder que será la clase que implementará la extracción de las palabras representativas utilizando las frecuencias de aparición de los términos.

ILanguageDetector: Es la encargada de verificar que tipo de idioma presenta el texto, es decir; español, inglés, francés, etc.

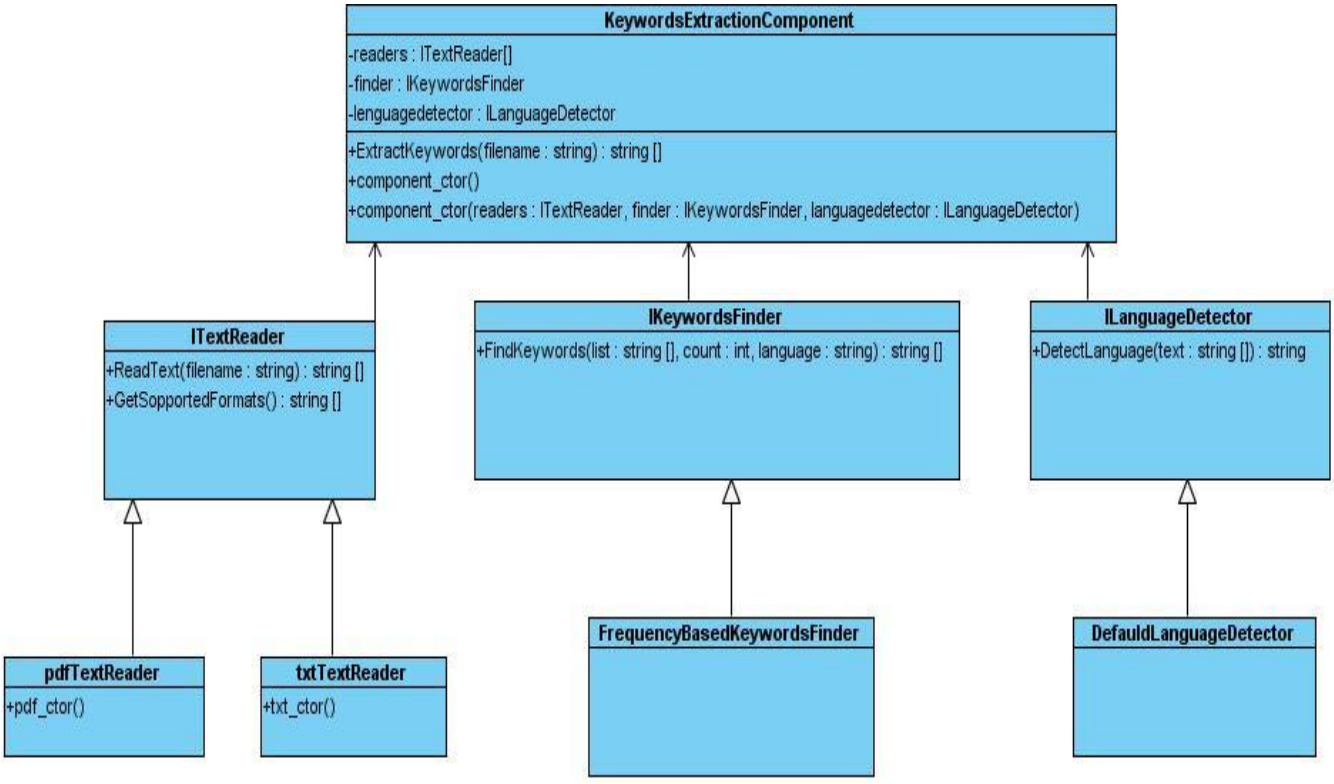


Figura 4: Diagrama de clases del diseño.

En la Figura 4 se muestra el diagrama de clases del diseño para el caso de uso Extraer palabras representativas. La clase KeywordsExtractionComponent es la encargada de interactuar con el Llamador y brindarle a este las funcionalidades del componente. Presenta los métodos necesarios para cumplir con los requisitos funcionales.

2.6.3 Diagrama de interacción entre las clases del diseño

2.6.3.1 Diagrama de secuencia

Los diagramas de secuencia muestran la secuencia cronológica de mensajes entre objetos durante un escenario concreto, la vida de cada objeto viene dada por una barra vertical y el tiempo transcurre de arriba abajo.

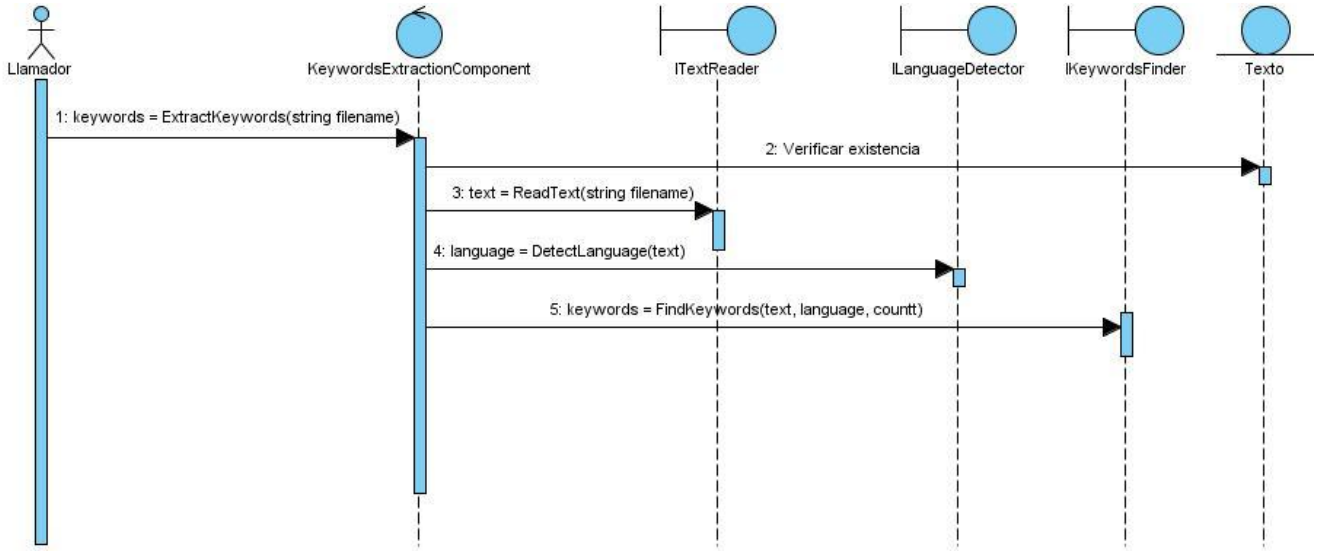


Figura 5: Diagrama de secuencia.

En la figura 5 se muestra el diagrama de secuencia del caso de uso Extraer palabras representativas donde se muestra la interacción en el tiempo entre los objetos a través de mensajes. El Llamador interactúa con la clase KeywordsExtractionComponent la cual le ofrece las funcionalidades para extraer las palabras representativas del documento.

2.6.3.2 Diagrama de colaboración

Los diagramas de colaboración ilustran las interacciones entre objetos en un formato de grafo o red, en el cual los objetos se pueden colocar en cualquier lugar del diagrama.

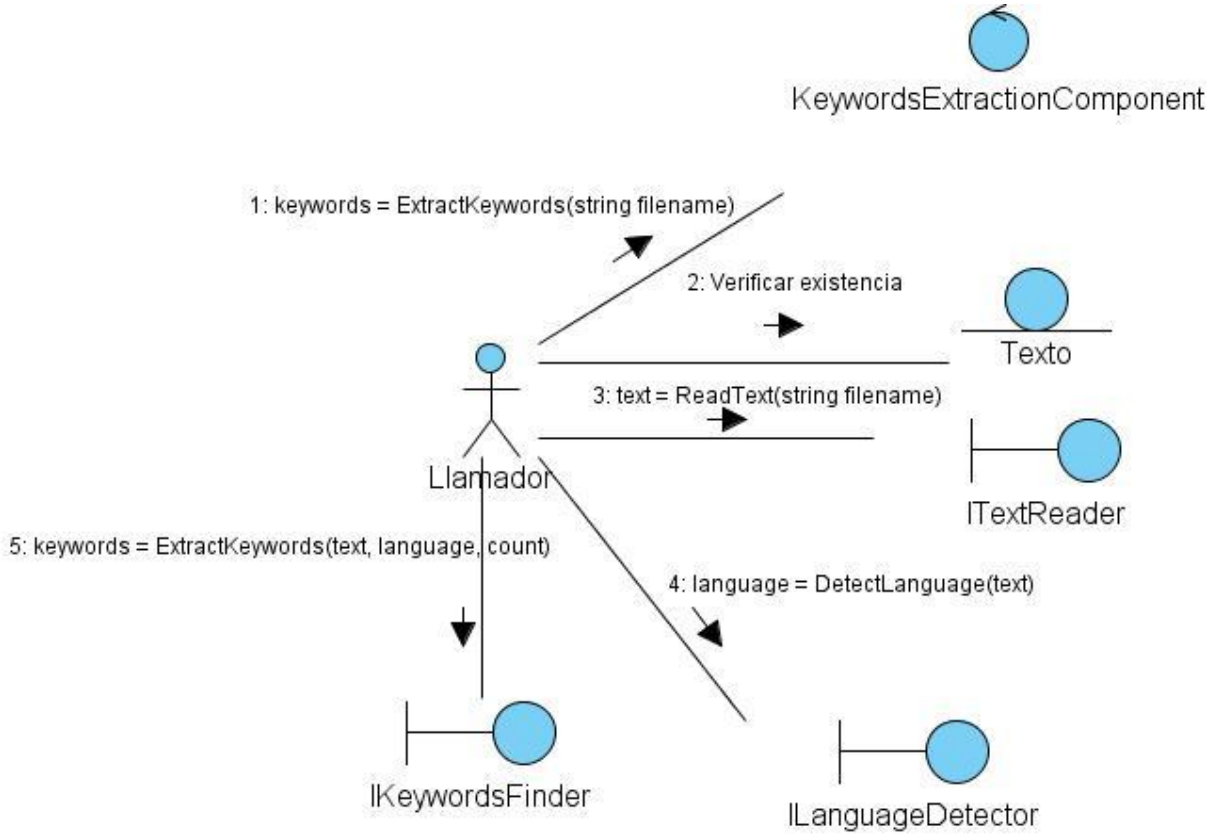


Figura 6: Diagrama de colaboración.

En la Figura 6 se muestra el diagrama de colaboración del caso de uso Extraer palabras representativas, en el cual se muestran los principales mensajes intercambiados entre las clases para realizar el caso de uso, se obtiene como resultado las palabras representativas del documento.

2.7 Conclusiones

En este capítulo se realizó el diagrama del modelo de dominio, con él se permite ver los objetos que existen relacionados con el proyecto. El diagrama de clases de análisis permitirá ver un dominio del problema buscando una solución ideal. Con el diagrama del diseño el programador podrá realizar una correcta implementación del sistema a desarrollar. Además se realizaron los diagramas de interacción donde muestran cómo se comunican los objetos. Estos diagramas representan de forma precisa las interacciones entre los mismos, en esencia, su misión es localizar el comportamiento de los objetos.

Capítulo 3: Implementación y prueba.

3 Introducción

Se comenzará la implementación con el resultado del modelo de diseño donde se implementará en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables. En este capítulo se realizarán las pruebas al componente permitiendo verificar y revelar la calidad del componente. Se abordan temas como las pruebas realizadas al software, en específico las pruebas de caja blanca.

3.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas.

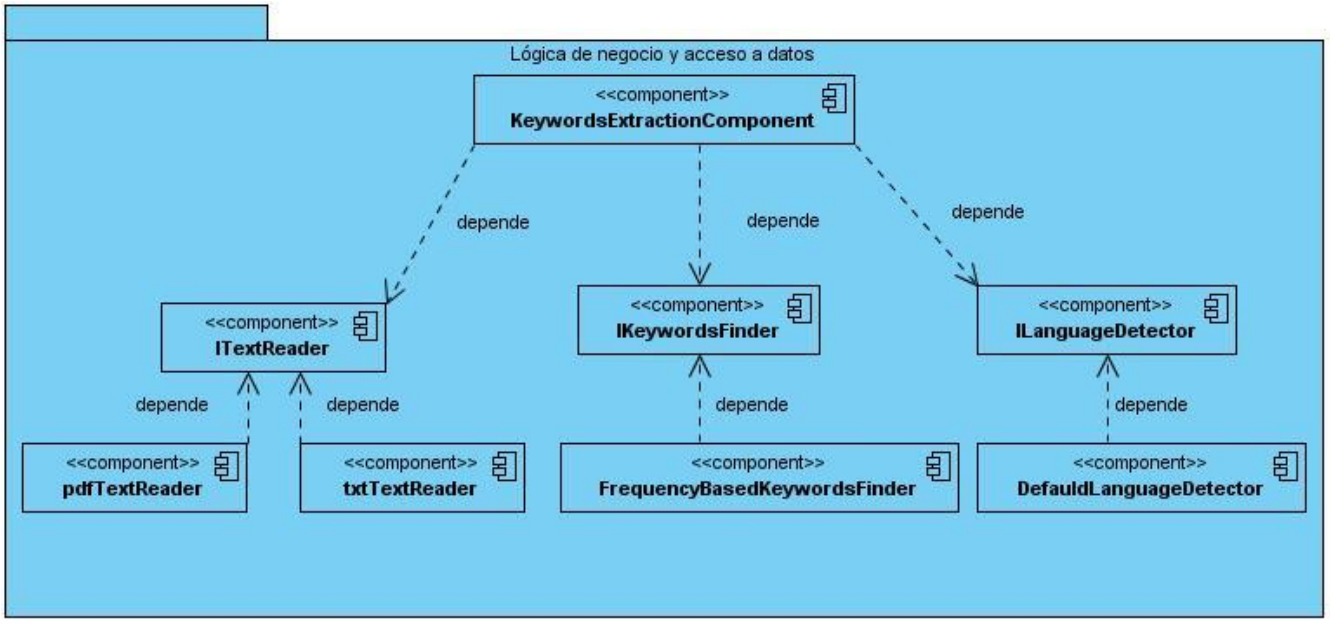


Figura 7: Diagrama de componentes

3.2 Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.(34)

Estas pruebas son técnicas de validación o procesos de ejecución de un programa con la intención de descubrir errores de una aplicación que antes no se habían descubierto.

A la hora de realizarle una prueba a un producto de software se debe escoger los tipos de prueba que se adapte mejor al sistema que se va a probar. Se debe tener en cuenta el lenguaje de programación utilizado, así como el proceso de desarrollo, las características de los desarrolladores y los errores más importantes.

3.2.1 Prueba de caja blanca

Las pruebas de Caja Blanca se nombran de esta forma porque a diferencia de las pruebas de Caja Negra que actúan sobre la interfaz, estas revisan la parte interna del software, específicamente sobre el código fuente. Se basan en el examen minucioso de los detalles procedimentales. Se comprueban los caminos lógicos del sistema generando casos de prueba que ejerciten las estructuras condicionales y los bucles. Existen varios métodos que analizan diferentes partes del programa y se complementan entre sí para garantizar la calidad del sistema.

La técnica del camino básico se utiliza para comprobar la complejidad lógica de un diseño procedimental, permite diseñar casos de prueba para cubrir todas las sentencias de un programa a partir de la obtención de un conjunto de caminos independientes.

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual cada nodo del grafo corresponde a una o más sentencias de código fuente, todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo, se calcula la complejidad ciclomática del grafo.

Un grafo de flujo está formado por 3 componentes fundamentales (nodos, aristas, regiones) que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente.(35)

Para realizar la técnica de prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar.La complejidad ciclomática, como resultado fundamental de estas pruebas, acota la cantidad mínima de casos de prueba que se deben ejecutar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método **ReadText()** el cual se encarga de leer el texto.

```
QList<QString>* TxtTextReader::ReadText()
{
    QList<QString>* list = new QList<QString>();1
    QFile* file = new QFile(this->filename);1

    if(!file->open(QIODevice::ReadOnly | QIODevice::Text))2
    {
        throw "Error abriendo el fichero";3
    }
    else
    {
        QTextStream in(file);4
        while(!in.atEnd())5
        {
            QString line = in.readLine();6
            QStringList words = line.split(" ", QString::SkipEmptyParts);6
            for(int i = 0; i < words.count(); i++)7
                list->append(words.at(i));8
        }
        file->close();9
    }
    return list;0
}
```

Figura 8: Representación del algoritmo del método ReadText()

A continuación se representa el Grafo de flujo asociado al algoritmo anteriormente descrito.

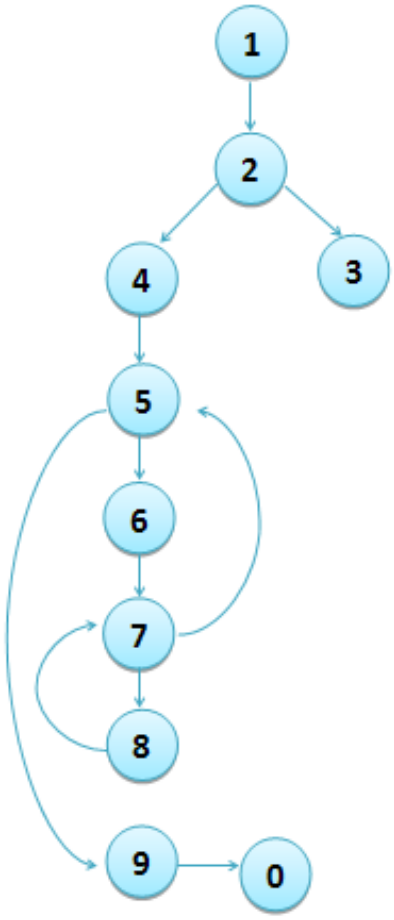


Figura 9: Grafo de flujo asociado al algoritmo ReadText().

Cálculo de la complejidad ciclomática a partir de un segmento de código

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A – N) + 2

V (G) = (11 - 10) + 2

V (G) = 1 + 2

V (G) = 3

Siendo "**A**" la cantidad total de aristas y "**N**" la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

Siendo "**P**" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 3

Siendo "**R**" la cantidad total de regiones, para cada formula "**V (G)**" representa el valor del cálculo.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 3, lo que significa que existen tres posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 4: Caminos básicos del flujo

Número	Camino básico
1	1 – 2 – 3
2	1 – 2 – 4 – 5 – 6 – 7 – 8 – 7 – 5 – 9 – 0
3	1 – 2 – 4 – 5 – 9 – 0

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Caso de prueba para el camino básico 1

Camino 1: [1 – 2 – 3]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro file no existe.

Entrada

Valor de file: no existe

Resultados esperados del camino básico 1:

Se espera que si no se pudo abrir el fichero retorne un error de abriendo el fichero.

Caso de prueba para el camino básico 2

Camino 2: [1 – 2 – 4 – 5 – 6 – 7 – 8 – 7 – 5 – 9 – 0]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro file existe.

Entrada:

Valor de file: file.txt

El texto que se encuentra en file es:

Ingeniería de software es la disciplina o área de la Ingeniería que ofrece métodos y técnicas para desarrollar y mantener software.

Resultados esperados del camino básico 2:

Se espera que se abra el fichero y mientras que este no llegue al final line = *Ingeniería de software es la disciplina o área de la Ingeniería que ofrece métodos y técnicas*, words={Ingeniería | de | software | es | la | disciplina | o |área | de | la | Ingeniería | que | ofrece | métodos | y | técnicas}

Luego de agregar la línea de texto a la lista de words, se verifica si file ha llegado al final, como no ha llegado al final line = *para desarrollar y mantener software.*, words={Ingeniería | de | software | es | la | disciplina | o | área | de | la | Ingeniería | que | ofrece | métodos | y | técnicas| para | desarrollar | y | mantener | software | **.**}

Ya en este caso file.txt llegó al final por lo que se cierra file y retorna la lista de words representada anteriormente.

Caso de prueba para el camino básico 3

Camino 3: [1 – 2 – 4 – 5 – 9 – 0]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro file exista.

Entrada

Valor de file: file.txt

El archivo file.txt no contiene ninguna palabra.

Resultados esperados del camino básico 3:

Se espera que se abra el fichero, luego este llegue al final del texto por lo que se cierra file.txt y después retorne la lista, que en este caso estará vacía.

Tabla 5: Resultados de la prueba de caja blanca.

Camino	Caso de prueba	Función del código	Resultado
1	Probando la función ReadText()	Se comprueba que la primera condición no se cumpla.	Satisfactorio
2	Probando la función ReadText()	Se comprueba que la primera condición se cumpla, luego se verifica que la siguiente condición se cumpla y que esta retorne la lista de palabras.	Satisfactorio
3	Probando la función ReadText()	Se comprueba que la primera condición se cumpla, luego se verifica que la siguiente condición no se cumpla, por lo que se retornaría la lista vacía.	Satisfactorio

Resultados esperados de la prueba:

Luego de haberse aplicado pruebas de caja blanca, en específico las pruebas del camino básico, seleccionando diferentes caminos a través del cálculo de la complejidad ciclomática a partir de un segmento de código, se ha arribado a la conclusión de que los resultados obtenidos fueron aceptables pues se pudo comprobar que el flujo de trabajo de la función esta correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.3 Conclusiones

En este capítulo se realizaron las pruebas de caja blanca, en específico la prueba del camino básico en las que se seleccionan diferentes caminos a través del cálculo de la complejidad ciclomática. En general se obtuvieron resultados favorables pues se logró probar que las funcionalidades fueron implementadas correctamente y que cumplen con los requisitos planteados.

CONCLUSIONES GENERALES

Con la culminación de la presente investigaciónse llegó a las siguientes conclusiones:

- ✓ El estudio realizado sobre los métodos para extraer las palabras representativas de un documento, demostró que el método por frecuencia de término es elmejor para desarrollar el componente.
- ✓ El sistema implementado aumenta la cantidad de componentes disponibles en el departamento Señales Digitales, lo cual mejora la capacidad de respuesta del departamento ante nuevos proyectos que necesiten extraer las palabras representativas de un documento.
- ✓ Se utilizó una técnica de caja blanca, específicamente la técnica del camino básico, para probar el componente. Los resultados arrojados durante la prueba indicaron que el componente cumple con los requisitos especificados.

RECOMENDACIONES

Con el objetivo de mejorar la solución planteada se propone como recomendaciones:

- ✓ Implementar las funcionalidades para que el componente sea capaz de leer otro tipo de ficheros como .RTF y .DOC.
- ✓ Implementar la detección automática del lenguaje del texto.

BIBLIOGRAFÍA

1. Diccionario de Organización y Representación del Conocimiento. [En línea] [Citado el: 20 de Octubre de 2010.] http://www.eubca.edu.uy/diccionario/letra_p.htm.

2. Los patrones de diseño. [En línea] [Citado el: 20 de Octubre de 2010.] http://www.scribd.com/doc/23941988/Los-patrones-de-diseno.

3. Librería estandar C++. [En línea] 1990. [Citado el: 20 de Octubre de 2010.] http://www.zator.com/Cpp/E5.htm.

4. Téllez Valero, Alberto.*Extracción de Información con Algoritmos de Clasificación.* Tonantzintla, Pue : s.n., 2005.

5. Keyrow, una herramienta SEO para obtener las palabras clave de un sitio. [En línea] [Citado el: 25 de Octubre de 2010.] http://www.incubaweb.com/keyrow-una-herramienta-seo-para-obtener-las-palabras-clave-de-un-sitio/.

6. Emprendedores-de-negocios.com. [En línea] [Citado el: 11 de Noviembre de 2010.] http://www.emprendedores-de-negocios.com/nicho.html.

7. adseokSEO. [En línea] [Citado el: 12 de Noviembre de 2010.] http://www.adseok.com/herramientas-seo/un-google-suggest-visual/.

8. CERPAMID. [En línea] 2004. [Citado el: 29 de Octubre de 2010.] http://www.cerpamid.co.cu.

9. [En línea] 2004. [Citado el: 1 de Noviembre de 2010.] http://www.cenatav.co.cu/es/.

10. Lenguajes de Programaciñon. [En línea] Agosto de 2009. [Citado el: 20 de Octubre de 2010.] http://es.kioskea.net/contents/langages/langages.php3.

11. Lenguajes de Programación. [En línea] 2009. [Citado el: 22 de Octubre de 2010.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

12. slideshare. [En línea] [Citado el: 22 de Octubre de 2010.] http://wapedia.mobi/es/Lenguaje_de_programaci%C3%B3n_Java.

13. Lenguajes de Programación. [En línea] 2009. [Citado el: 23 de Octubre de 2010.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

14. freedesigns. [En línea] 11 de Noviembre de 2007. [Citado el: 23 de Octubre de 2010.] http://freedesigns.foroactivo.com/t71-aprendiendo-a-programas-c.

15. casidiablo.net|Java, Linux y Programación. [En línea] 2006. [Citado el: 23 de Octubre de 2010.] http://casidiablo.net/c-y-net/.

16. Curso C++. [En línea] 1990. [Citado el: 26 de Octubre de 2010.] http://www.zator.com/Cpp/E1_2.htm.

17. Metodología de Desarrollo de Software. [En línea] [Citado el: 27 de Octubre de 2010.] http://www.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software.

18. Willy Dev. [En línea] 2003. [Citado el: 27 de Octubre de 2010.] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

19. EcuRed. [En línea] 21 de Febrero de 2011. [Citado el: 17 de Marzo de 2011.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo.

20. Buenas Tareas. [En línea] [Citado el: 27 de Octubre de 2010.] http://www.buenastareas.com/ensayos/Ventajas-Del-Proceso-Unificado-De-Desarrollo/573875.html.

21. Empresa de Proyectos de Arquitectura e Ingeniería. [En línea] [Citado el: 27 de Octubre de 2010.] http://www.empai-matanzas.co.cu/revista/Rev_PDF/2008/Vol.2%20%20No.3%20%202008.pdf.

22. Jacobson, Ivar, Booch, Grady y Rumbaugh, James.*El Proceso Unificado de Desarrollo de Software.*

23. Herramientas CASE. [En línea] 2010. [Citado el: 4 de Noviembre de 2010.] http://www.slideshare.net/guestf131a9/herramientas-case.

24. Visual Paradigm for UML. [En línea] 2004. [Citado el: 2 de Noviembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

25. Estrada Velasco, Aylin y Jean, Suarez Michel.*Sistema de Captura e Indexación de Video.* Ciudad de La Habana : s.n., 2009.

26. IDEs. [En línea] 2009. [Citado el: 8 de Noviembre de 2010.] http://www.softwarelibre.ec/site/index.php?option=com_content&view=article&id=198&Itemid=165.

27. Qt. [En línea] 8 de Noviembre de 2010. http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.

28. Zona Qt. [En línea] [Citado el: 9 de Mayo de 2011.] http://www.zonaqt.com/tutoriales/tutorial-de-qt-4-por-zona-qt-0.

29. KDevelop. [En línea] 2009. [Citado el: 8 de Noviembre de 2010.] http://www.softwarelibre.ec/site/index.php?option=com_content&view=article&id=197%3Akdevelop&catid=40%3Aides&Itemid=165.

30. Sommerville, Ian.*Ingenieria del software.* España : Pearson Educación, S.A., 2005.

31. Scribd. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.scribd.com/doc/32854812/Documento-de-Requerimiento-de-software.

32. Larman, Craig.*Una introduccion al analisis y diseño orientado a objetos y al proceso unificado.*

33. *Ingeniería de Software 1. Conferencia #10. Fase de Elaboración. Flujo de trabajo de Análisis y Diseño.* 2010-2011.

34. Ingeniería de Software II. Conferencia 7: Disciplina de Prueba. Materiales Básicos. [En línea] [Citado el: 5 de Mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14103.

35. Ingeniería de Software II. Conferencia 7: Disciplina de Prueba. Materiales Complementarios. EVA. [En línea] [Citado el: 13 de Mayo de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Complementarios/Material_de_caja_b_y_caja_n.pdf.

36. Wapedia. [En línea] [Citado el: 22 de Octubre de 2010.] http://wapedia.mobi/es/Lenguaje_de_programaci%C3%B3n_Java.

37. BlogTopSItes. [En línea] 2010. [Citado el: 11 de Noviembre de 2010.] http://www.blogtopsites.com/post/palabras+clave.

38. [En línea] 30 de Septiembre de 2004. [Citado el: 20 de Octubre de 2010.] http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf.

39. Qt. [En línea] http://www.softwarelibre.ec/site/index.php?option=com_content&view=article&id=198&Itemid=165.

40. Sistema Gestor de Base de Datos. [En línea] 2004. [Citado el: 9 de Noviembre de 2010.] http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.php.

41. Que es Oracle. [En línea] [Citado el: 9 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/840.php.

42. MySQL. [En línea] 2010. [Citado el: 9 de Noviembre de 2010.] http://dev.mysql.com/doc/refman/5.0/es/introduction.html.

43. Librerías para generar archivos .PDF. [En línea] [Citado el: 11 de Noviembre de 2010.] http://www.maestrosdelweb.com/editorial/librerias-para-generar-archivos-pdf/.

44. Categorización de Textos. [En línea] 8 de Marzo de 2006. [Citado el: 26 de Enero de 2011.] http://legutier.blogspot.com/2006/03/algoritmo-de-porter-enespaol.html.

45. Sierra, María.*INGENIERÍA DEL SOFTWARE I.*

GLOSARIO

Datos se caracterizan por no contener ninguna información. Un dato puede significar un número, una letra, un signo ortográfico o cualquier símbolo que represente una cantidad, una medida, una palabra o una descripción.

Información es un conjunto de datos significativos y pertinentes que describan sucesos o entidades.

Indización es el proceso de describir o representar el contenido temático de un recurso de información. Este proceso da como resultado un índice de términos de indización que será utilizado como herramienta de búsqueda y acceso al contenido de recursos en sistemas de recuperación de información.