Sistema de categorización de documentos para el buscador cubano Orión

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:
Jorge Ignacio Martínez Ravelo

Tutores:
Ing. Juan Manuel Alvarez Tur
Ing. Miguel Ángel Chávez Alfonso
Ing. Hunyaris Mariela Marín Fonticoba

La Habana, junio, 2017

Declaratoria de autoría

Declaro por este medio que yo, Jorge Ignacio Martínez Ravelo, con carné de identidad 93111806265 soy el autor principal del trabajo titulado Sistema de categorización de documentos para el buscador Orión y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos el presente documento a los ____ días del mes de junio del año 2017.

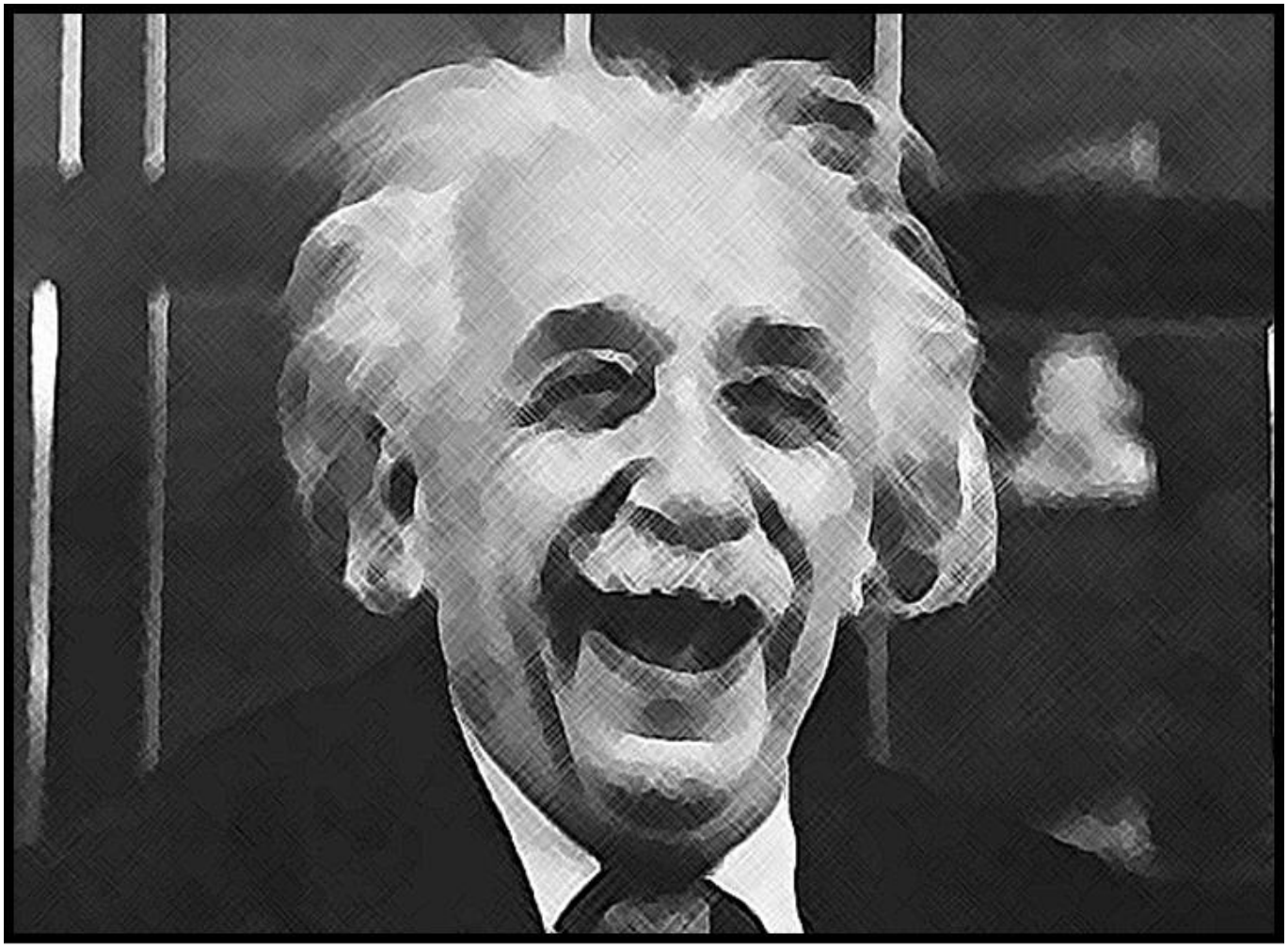
Jorge Ignacio Martínez Ravelo

_____

Firma del autor

Juan Manuel Alvarez Tur	Miguel Ángel Chávez Alfonso	Hunyaris Mariela Marín Fonticoba
_____	_____	_____
Firma del tutor	Firma del tutor	Firma del tutor

Frase



"Una persona que nunca ha cometido un error nunca intenta nada nuevo"

Albert Einstein

Dedicatoria:

Dedico este trabajo de diploma a mis padres, por su apoyo incondicional en esta travesía de convertirme en ingeniero informático, a mis abuelos y mi hermana por su apoyo y su cariño, a mi novia y mis amigos por compartir tantos momentos inolvidables en el transcurso de estos cinco años de universidad.

Agradecimientos:

A mi madre

Por ser la voz que me impulsa en la vida y el faro que alumbra mi camino, por ser mí guía, mi consejera, mi amiga, por ser todo lo que un hijo desearía tener.

A mi padre

Por ser siempre mí amigo y brindarme su apoyo en todas las decisiones que tomo, por hacerme reír y reflexionar en cada paso que doy.

A mis abuelos y mi hermana

Por brindarme su amor y su cariño, por haber sido piezas claves en mi formación y por su constante preocupación por mi bienestar.

A mi novia

Por su cariño, apoyo y comprensión durante el transcurso de este difícil proceso.

A mis amigos

En especial a Reinaldo y Lázaro que han sido mis compañeros desde el primer día, con lo que he compartido los malos y buenos momentos. Por ser mis confidentes y mis hermanos.

A mis Tutores

Por el apoyo y el tiempo que han dedicado para la tutoría de este trabajo.

Resumen

La presente investigación tiene como objetivo proponer un sistema de categorización de documentos en español para el buscador cubano Orión, que permita categorizar los documentos indexados por el buscador, para contribuir así, en el proceso de recuperación de información en la red cubana. Durante la investigación se analizaron sistemas homólogos existentes en el ámbito internacional y nacional identificando su funcionamiento y los principales algoritmos para la categorización de documentos, así como sus aspectos positivos y negativos. Para darle cumplimiento al objetivo planteado, en la investigación se realizó el estudio del estado del arte sobre el proceso de categorización de documentos y recuperación de información. El proceso de desarrollo estuvo guiado por la metodología de desarrollo de *software*[1] AUP-UCI. Se seleccionaron como principales tecnologías: entorno de desarrollo integrado (IDE) Netbeans 8.0, para la programación el lenguaje Java 8, como herramienta para la minería de datos Weka 3.7.12, como indexador la herramienta Solr 4.10.3 y Visual Paradigm 8.0 como herramienta para el modelado. A partir de la aplicación de las pruebas de software de tipo unitaria, integración y funcionalidad se verificó un correcto funcionamiento del sistema implementado para contribuir en la optimización del proceso de recuperación de la información en el buscador Orión.

Palabras clave: algoritmos, buscador, categorización, documentos, Orión.

[1] Según la Real Academia Española (RAE), el software es un conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

Índice

Declaratoria de autoría II

Frase III

Dedicatoria: IV

Agradecimientos: V

Resumen VI

Introducción 1

Capítulo 1: “Fundamentación teórica sobre los sistemas de categorización de documentos” 7

1.1 Conceptos asociados a la investigación. 7

1.2 Estudio sobre Sistemas Homólogos. 11

1.3 Algoritmos para la categorización automática de documentos. 14

1.4 Ambiente de Desarrollo. 17

1.5 Metodologías de desarrollo. 22

Conclusiones del capítulo. 26

El Capítulo 2: “Análisis y diseño del sistema de categorización de documentos” 27

2.1 Diagrama de Modelo de Dominio. 27

2.2 Propuesta de Solución. 28

2.3 Artefactos. 29

2.4 Requisitos del Sistema. 29

2.5 Historias de Usuario. 31

2.6 Arquitectura del sistema. 34

2.7 Diagrama de clases. 36

2.8 Patrones de diseño. 38

2.9 Modelo de despliegue. 40

Conclusiones del capítulo. 40

El Capítulo 3: “Implementación y prueba del sistema de categorización de documentos” 41

3.1 Estándar de codificación. 41

3.2 Diagrama de componentes. 43
3.3 Colección de entrenamiento. 44
3.4 Estrategia de pruebas. 45
3.5 Validación del sistema. 53
Conclusiones de capítulo. 55
Conclusiones generales 56
Recomendaciones 57
Bibliografía 58

Introducción

Para localizar y procesar la información de forma rápida y automática en Internet son utilizados los motores de búsqueda o buscadores como sistemas recolectores de información, que son capaces de encontrar contenido existente en la Web. En la actualidad resulta fácil para los usuarios con conexión a Internet buscar en la Web las canciones de sus artistas favoritos, los videos más seguidos, realizar llamadas, comunicarse con personas al otro lado del planeta o encontrar información de cualquier tipo entre cientos de aplicaciones. Esto ha convertido a las tecnologías de la información y las comunicaciones (en lo adelante TIC), y su constante desarrollo en una parte importante de la vida de las personas. Desde la creación de la Internet (también conocida como la Red de Redes) en el año 1969, que posteriormente le da paso a la llamada sociedad de la información, en la cual las tecnologías facilitan la creación, distribución y manipulación de la información, juega un papel esencial en las actividades sociales, culturales y económicas a nivel mundial.

Los volúmenes de documentación existentes en la Web han ido alcanzando niveles extraordinarios y continúan creciendo diariamente, el promedio de datos transferidos por minuto a nivel global es de casi 639 800 Gigabytes (GB), lo que implica millones de fotografías, correos electrónicos y búsquedas en sólo 60 segundos, reveló un estudio de Intel[2] en el año 2015, por lo que encontrar la información correcta resulta de suma importancia para los millones de usuarios que interactúan con la Internet.

Los seres humanos siempre han buscado la manera de organizar los grandes cúmulos de información que han recopilado a lo largo de su historia. La tarea de encontrar grupos de documentos con características comunes no sólo es compleja, sino que además consume tiempo. Un bibliotecario que tratara de categorizar un documento tendría que leerlo para comprender su significado y luego asignarle una categoría utilizando su experiencia y sentido común. El costo y el tiempo asociados a la categorización de documentos han llevado a la investigación de técnicas que permitan automatizar la tarea (Rüger, et al., 2000).

La categorización automática puede contemplarse como un proceso de aprendizaje, durante el cual el programa capta las características que distinguen cada categoría o clase de las demás, es decir, aquellas que deben poseer los documentos para pertenecer a esa categoría. Estas características no tienen por qué indicar de forma absoluta la pertenencia a una clase o categoría, sino que más bien lo hacen en

[2] ***Int****egrated* ***El****ectronics Corporation* es el mayor fabricante de circuitos integrados del mundo (Intel, 2017).

función de una escala o graduación. Ejemplo: documentos que posean ciertas características tendrán un factor de posibilidades de pertenecer a determinada clase (Figuerola, et al., 2000).

Desde la aparición de los primeros catálogos en línea se han realizado experimentos para demostrar el potencial de los principales sistemas de categorización y explotar sus posibilidades como instrumentos para mejorar el acceso a la información. El primer proyecto importante que intentaba evaluar las posibilidades de mejorar la recuperación en línea utilizando una categorización fue el desarrollado en 1968 por Atherton (Atherton, 1968). Las investigaciones arrojaron como resultado que los sistemas de categorización eran buenos para recuperar información en entornos automatizados. Se demostró la capacidad del sistema de clasificación decimal universal para la recuperación de información aunque sus conclusiones no tuvieron una repercusión directa en la evolución de este sistema, entre otros motivos, por las dificultades que planteaba su gestión en el seno de la Federación Internacional de Información y Documentación (FID) (Castro, 1998).

Cuba desde hace algunos años atraviesa un proceso de informatización, donde la Universidad de las Ciencias Informáticas (UCI) tiene la misión de producir aplicaciones y brindar servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria nacional de la informática. Dentro de la infraestructura docente-productiva de la UCI, se encuentra el Centro de Ideoinformática (CIDI), dentro de sus objetivos de trabajo tiene a cargo el desarrollo del buscador web Orión. El objetivo de este proyecto es apoyar el proceso de informatización del país ofreciendo una ventana a los contenidos alojados en los servidores nacionales.

Actualmente el buscador Orión se encuentra en un proceso para mejorar sus prestaciones y optimizar su proceso de recuperación de información para brindar mejores resultados a los usuarios. Las búsquedas en Orión están orientadas a todo tipo de usuarios, entre ellos los que tienen poca experiencia en las búsquedas en la web. Las técnicas de representación de las preferencias o recomendaciones de los usuarios constituyen un mecanismo para ayudar a encontrar mejores resultados. Estas técnicas trabajan con perfiles de usuarios que utilizan entre otros aspectos, la categorización de los resultados en diferentes categorías para recomendar información en búsquedas posteriores.

Con el objetivo de mostrar resultados personalizados y así aumentar la fidelización de clientes, se encuentra en implementación un componente que permite realizar búsquedas utilizando datos históricos y estadísticos del usuario para el cálculo de una relevancia personalizada. Este nuevo componente requiere

como precondición la existencia de un campo categoría asociado a cada documento almacenado en el buscador Orión.

La implementación de un módulo para el análisis estadístico y el apoyo a la toma de decisiones permitirá mostrar información estadística en formato de gráficos o tablas, perfiles de búsquedas de un usuario y de publicación de un sitio web. Este componente tiene un carácter político, económico, social y organizativo. Uno de los requisitos de su implementación es mostrar el comportamiento de publicación de un sitio web específico en un período de tiempo por cada categoría.

La recomendación de la información para los usuarios, identificación de tendencias, la creación de perfiles de usuarios y la optimización del cálculo de la relevancia de la información constituyen elementos fundamentales en la optimización del proceso de recuperación de la información que se lleva a cabo el sistema Orión. Todos estos nuevos desarrollos demandan la categorización de los documentos. Realizar el proceso de categorización de los documentos indexados en el buscador Orión de forma manual necesitaría muchos recursos humanos y un tiempo prolongado teniendo en cuenta que la web cubana posee aproximadamente 5000 dominios registrados según el portal Cubanic (http://nic.cuba.cu/estadisticas.php), con más 711 507 páginas web indexadas según estadísticas de los desarrolladores de la plataforma cubana C.U.B.A. hasta el 16 de mayo del 2017.

Teniendo en cuenta la situación problemática expuesta anteriormente se enuncia el siguiente **problema de la investigación:** ¿Cómo contribuir a la optimización en el proceso de recuperación de información del buscador Orión?

Para llevar adelante la investigación se plantea el siguiente **objeto de estudio:** categorización de documentos; y el **campo de acción** se define como: categorización de documentos almacenados en el buscador Orión.

Para darle solución al problema descrito, se ha planteado el siguiente **objetivo general:**

- Desarrollar un sistema para la categorización de documentos a partir de la información indexada en el buscador Orión para contribuir a la optimización de su proceso de recuperación de información, utilizando herramientas para el procesamiento del lenguaje natural.

Objetivos específicos:

1. Construir el marco teórico conceptual y el estado del arte respecto a las tecnologías y funcionalidades de los sistemas de categorización de documentos.
2. Diseñar el sistema de categorización de documentos para el buscador Orión.

3. Implementar un sistema que permita vincular los documentos indexados en el Buscador Orión a diferentes categorías.
4. Validar el correcto funcionamiento del sistema desarrollado.

Para darle cumplimiento al objetivo planteado se responderán las siguientes preguntas científicas:

1. ¿Cuáles son los referentes teóricos que sustentan el proceso de categorización automática de documentos para buscadores web?
2. ¿Qué características tienen las herramientas existentes, que permiten el desarrollo de un sistema de categorización automática de documentos?
3. ¿Qué aspectos deben tenerse en cuenta para realizar el análisis y diseño del sistema de categorización de documentos para el buscador Orión?
4. ¿Cómo llevar a cabo el desarrollo del sistema de categorización de documentos para el buscador Orión?
5. ¿Qué resultados se obtendrán al validar el sistema de categorización automática de documentos para el buscador Orión?

Para lograr los objetivos descritos anteriormente se plantean las siguientes **tareas de investigación**:

1. Estudio de los conceptos asociados al marco teórico de la investigación.
2. Caracterización de los algoritmos utilizados para la categorización de documentos.
3. Caracterización de los sistemas de categorización de documentos.
4. Determinación de los cambios necesarios en el índice de documentos mediante la tecnología Solr utilizado por el buscador Orión para que en las consultas realizadas por los usuarios se puedan incluir el término categoría.
5. Determinación de las herramientas idóneas para realizar una categorización a varios documentos a partir de una colección de entrenamiento.
6. Identificación de las colecciones de entrenamiento que pueden ser obtenidas de diferentes fuentes.
7. Definición de la arquitectura del sistema.
8. Definición de los requisitos funcionales y no funcionales.
9. Diseño e implementación de las clases y métodos que den solución a los requisitos definidos.
10. Diseño y ejecución de casos de pruebas para las funcionalidades implementadas.

11. Validación del sistema desarrollado teniendo en cuenta su funcionamiento.

Para dar solución a las tareas de investigación se utilizan los siguientes métodos de investigación científica:

Métodos Teóricos:

Histórico-Lógico: Utilizado para estudiar el desarrollo evolutivo de los sistemas de categorización automática de documentos así como los algoritmos más utilizados para una mayor comprensión de su funcionamiento, el surgimiento y desarrollo de los sistemas de recuperación de la información, en especial los buscadores web.

Analítico-Sintético: Empleado para descomponer en sus partes el objeto de estudio, permitiendo luego combinar los elementos en una propuesta de solución para Orión.

Modelado: Utilizado para realizar una reproducción simplificada de la realidad, mostrando las relaciones internas y características del sistema a través de diagramas.

Métodos Empíricos:

Entrevista: A través de preguntas a los principales especialistas del proyecto Orión, permitió conocer las características y problemas que genera en el buscador la ausencia de un proceso para la categorización de documentos.

El presente trabajo de diploma consta de la siguiente estructura: Introducción, tres capítulos, Conclusiones, Recomendaciones, Bibliografía y Anexos.

El Capítulo 1: "Fundamentación teórica sobre los sistemas de categorización de documentos"

Se brinda una visión general de los aspectos relacionados con el proceso de categorización de documentos. Se efectúa el estudio del estado del arte y un análisis sobre sistemas homólogos, se seleccionan los lenguajes y herramientas utilizadas para confeccionar el marco de trabajo.

El Capítulo 2: "Análisis y diseño del sistema de categorización de documentos"

Enfocándose en las características de los sistemas de categorización de documentos y los componentes que plantea la metodología, se expone la propuesta de diseño del sistema a desarrollar. Se especifican las funcionalidades del sistema con la descripción de los requisitos funcionales, no funcionales, las historias de usuario, la arquitectura a utilizar y los patrones de diseño que se emplean.

El Capítulo 3: “Implementación y prueba del sistema de categorización de documentos”

Se observa el diagrama de componentes y su descripción, con el fin de entender los elementos de software del sistema y sus relaciones. Se exponen los prototipos de interfaz principales del sistema desarrollado, donde se evidencian las funcionalidades implementadas. También se muestra la validación del diseño realizado mediante las estrategias de pruebas.

Capítulo 1: "Fundamentación teórica sobre los sistemas de categorización de documentos"

En el presente capítulo se brinda una visión general de los aspectos relacionados con la categorización automática de documentos. Se describen los principales conceptos asociados a la investigación, necesarios para entender el negocio y la propuesta de solución. Se analizan las herramientas y tecnologías utilizadas para concretar una propuesta, definiendo los elementos que las hacen más recomendables para la implementación de un categorizador automático de documentos para el buscador cubano Orión.

1.1 Conceptos asociados a la investigación.

Según la RAE una investigación es un proceso sistemático, organizado y objetivo, cuyo propósito es responder a una pregunta o hipótesis y así aumentar el conocimiento y la información sobre algo desconocido. Asimismo, la investigación es una actividad sistemática dirigida a obtener, mediante observación, la experimentación, nuevas informaciones y conocimientos que necesitan para ampliar los diversos campos de la ciencia y la tecnología. A continuación se presentan los conceptos más importantes de la investigación.

1. **Recuperación de Información.**

El proceso de recuperación de información (en lo adelante RI) se lleva a cabo mediante consultas a la base de datos donde se almacena la información estructurada, mediante un lenguaje de interrogación adecuado. Es necesario tener en cuenta los elementos claves que permiten hacer la búsqueda, determinando un mayor grado de pertinencia y precisión, como son: los índices, palabras clave, tesauros[3] y los fenómenos que se pueden dar en el proceso como son el ruido y silencio documental. Uno de los problemas que surgen en la búsqueda de información es ¿Qué tan buena es la RI? Es decir, dependiendo del tipo de búsqueda se pueden recuperar multitud de documentos o simplemente un número muy reducido. A este fenómeno se denomina silencio o ruido documental (Pinto, 2012).

- **Silencio documental:** Son aquellos documentos almacenados en la base de datos pero no han sido recuperados, debido a que la estrategia de búsqueda ha sido demasiado específica o que las palabras clave utilizadas no son las adecuadas para definir la búsqueda.

[3] Según la RAE, diccionario ideológico o de sinónimos.

- **Ruido documental:** Son aquellos documentos recuperados por el sistema, pero que no son relevantes. Esto suele ocurrir cuando la estrategia de búsqueda se ha definido demasiado genérica.

2. Motor de Búsqueda.

Un motor de búsqueda o buscador, es una pieza de software que permite encontrar y visitar los sitios relacionados con una palabra clave introducida al sistema por el mismo usuario. Básicamente, están compuestos por bases de datos con información sobre el contenido de los sitios que integran la Web.

Los motores de búsqueda están compuestos por tres partes:

1. Los robots o arañas que se encargan de recorrer la red escrutándola[4].
2. La base de datos que construyen estos robots.
3. El motor de búsqueda que facilita la consulta a la base de datos.

Los robots son programas que buscan o rastrean continuamente todos los servidores de la WWW[5], en *Gopher* (uno de los sistemas de Internet para RI) y FTP (permite el intercambio de datos entre diferentes servidores/ordenadores), que alimentan una base de datos. Los robots actualizan estas bases y añaden nuevas páginas o referencias cuando han cambiado o ya no existen. Cuando se ejecuta la página de algún buscador, se encuentra un formulario para definir la búsqueda y sus posibles opciones. Se ingresa la palabra o palabras clave que son las que describen los conceptos, ideas o términos buscados. El motor devuelve los resultados en función de cómo se haya definido la búsqueda (Yates, 2010).

La arquitectura del buscador Orión desarrollado en la UCI está sustentada por tres componentes (Leyva, 2016):

1. Como componente de recolección o araña se utiliza la herramienta Nutch debido a las múltiples ventajas que ofrece a lo largo de todo el proceso de rastreo y almacenamiento de la información.
2. Como componente de indexación la herramienta Solr, por las múltiples ventajas que ofrece este software.
3. Para el desarrollo del componente de visualización se utiliza el marco de trabajo *Symfony*, debido a la potencia que brinda en el desarrollo web de sistemas.

---

[4] Según la RAE, indagar, examinar cuidadosamente, explorar.

[5] *World Wide Web* (traducido en español como Red Global Mundial), es un sistema de documentos de hipertexto o hipermedios enlazados y accesibles a través de Internet.

3. Categorización Automática de Documentos.

El objetivo de las técnicas de RI es poder resolver consultas en forma eficaz y eficiente. Según Raghavan (Raghavan, et al., 1989) el criterio para evaluar la eficacia se basa en la relevancia de los resultados encontrados, y la eficiencia viene dada por la rapidez con la cual se resuelve la consulta. En el contexto de la RI (Van Rijsbergen, 1979), formula la denominada **"Hipótesis del Agrupamiento"**. Básicamente la hipótesis del agrupamiento sostiene que los documentos fuertemente asociados tienden a ser relevantes para la misma consulta, lo cual ha sido verificado experimentalmente ((Cutting, et al., 1992), (Schütze, et al., 1997), (Zamir, et al., 1999)). Basándose en dicha hipótesis, la agrupación automática tiene en cuenta el contenido de los documentos para agruparlos, ya que documentos similares contendrán palabras o términos similares.

Debido al incremento en los volúmenes de información disponibles en forma electrónica y a la necesidad cada vez mayor de encontrar la información buscada en un tiempo mínimo, estas técnicas de automatización mencionadas han estado recibiendo creciente atención ((Maarek, et al., 2000), (Suzuki, 1995), (Berry, et al., 2008), (Olson, 2008)).

Sus aplicaciones más importantes son:

- Mejorar el rendimiento de los motores de búsqueda de información, mediante la agrupación previa de todos los documentos disponibles (Faloutsos, et al., 1996). Antes de comenzar a resolver las consultas, el conjunto de documentos es separado en grupos. Luego, al resolver una consulta, no se examinan todos los documentos, sino que se busca el "grupo representante" que mejor responda a la consulta.
- Facilitar la revisión de resultados por parte del usuario final, agrupando los resultados luego de realizar la búsqueda (Allen, et al., 1993), cuando se realizan búsquedas sobre un gran volumen de documentos, la cantidad de resultados puede ser muy grande. Si la lista se presenta sin ningún tipo de procesamiento previo, el usuario del sistema se verá obligado a revisar todos los documentos, descartando aquellos que no son relevantes para él. Si los resultados se agrupan, los documentos del mismo grupo serán relevantes para una sub-consulta más específica que la original; de esta forma, los documentos quedarán separados en grupos temáticos. Así, con sólo revisar uno o dos documentos de cada grupo, el usuario podrá determinar cuál es el subtema al

que responden todos los documentos del grupo, pudiendo descartar rápidamente los documentos irrelevantes para su búsqueda.

- Aplicaciones al campo de la Minería de Datos, que permita extraer conocimiento de los documentos procesados, a través del análisis de la información obtenida en los agrupamientos.

4. Categorización supervisada y no supervisada.

Cuando se habla de categorización automática se distingue entre dos escenarios diferentes y requieren soluciones distintas. Estos escenarios reciben diversos nombres, pero básicamente consisten en lo siguiente:

- **Supervisada:** Se parte de una situación con una serie de clases o categorías conceptuales prediseñadas con anterioridad, donde la labor del categorizador implementado (manual o automático) es asignar cada documento a la clase o categoría que le corresponda según su contenido y sus características.
- **No supervisada:** No hay categorías previas, esquemas o cuadros de categorización establecidos a priori. Los documentos se agrupan en función de sí mismo, de su contenido; se puede plantear que se auto-organizan mutuamente. Es lo que se conoce como categorización automática no supervisada (*clustering*). Su nombre se debe a que se efectúa de forma totalmente automática, sin supervisión o asistencia manual (Figuerola, et al., 2004).

La categorización no supervisada de documentos no define categorías, organiza los documentos atendiendo a su contenido a través de comparaciones por lo que se convierte en un escenario no relevante para la investigación. El escenario supervisado tiene en cuenta categorías o clases conceptuales definidas a priori, además de bases documentales categorizadas de interés para un marco de trabajo en específico, asistido por la experiencia humana, como es el caso del buscador Orión.

5. Documento.

Un documento es un conjunto de datos de naturaleza textual, pero la evolución tecnológica ha propiciado la proliferación de documentos multimedia, añadiéndose al texto fotografías, ilustraciones gráficas, videos

animados, audio, etc. Aunque la variedad, en cuanto a documentos se refiere, está aumentando tanto en soportes como en el carácter de su contenido (Castillo, 2010).

6. Minería de Datos (*Data Mining*).

La minería de datos es una tecnología compuesta por etapas que integra varias áreas y que no se debe confundir con un gran software. Durante el desarrollo de un proyecto se usan diferentes aplicaciones software en cada etapa que pueden ser estadísticas, de visualización de datos o de inteligencia artificial. Actualmente existen aplicaciones o herramientas comerciales de minería de datos muy poderosas que contienen un sinfín de utilidades que facilitan el desarrollo de un proyecto. Sin embargo, casi siempre acaban complementándose con otra herramienta. Las técnicas de minería de datos son el resultado de un largo proceso de investigación y desarrollo de productos. Esta evolución comienza cuando los datos de negocios fueron almacenados por primera vez en computadoras, y continuó con mejoras en el acceso a los datos, y más recientemente con tecnologías generadas para permitir a los usuarios navegar a través de los datos en tiempo real. La minería de datos toma este proceso de evolución más allá del acceso y navegación retrospectiva de los datos, hacia la entrega de información prospectiva y proactiva*. Data Mining* está listo para su aplicación en la comunidad de negocios porque está soportado por tres tecnologías que ya están suficientemente maduras (Vallejos, 2006):

- Recolección masiva de datos.
- Potentes computadoras con multiprocesadores.
- Algoritmos de minería de datos.

1.2 Estudio sobre Sistemas Homólogos.

Según la RAE, homología, es un tipo de relación que puede darse en diversos ámbitos y que implica una paridad o semejanza. La categorización de documentos es una tarea compleja y subjetiva incluso para diferentes personas, que podrían asignar el mismo documento a grupos diferentes, cada una aplicando un criterio válido. A continuación se presenta el estudio realizado a los sistemas del ámbito internacional y nacional en el área de la categorización automática de documentos, para la comprensión del funcionamiento de estos sistemas en específico.

Internacionales:

Aspire:

Aspire Content Processing es un marco innovador y potente diseñado específicamente para datos no estructurados. Forma parte de la colección de recursos de tecnología independiente de *Search Technologies* que ayudan a las organizaciones a optimizar sus arquitecturas de búsqueda y grandes cúmulos de datos. Esta herramienta apoya los enfoques primarios de categorización: basado en vocabulario, utilizando una taxonomía (opcionalmente con sinónimos y reglas de retención) u otros recursos semánticos y la categorización basada en ejemplos, utilizando documentos de semillas. En algunos casos, una combinación de estos métodos proporciona los mejores resultados.

La versión corporativa de Aspire, normalmente entregada con los servicios de implementación, proporciona una solución completa, flexible y totalmente compatible con las necesidades de categorización de Solr (Search Technologies, 2016).

Proyecto Scorpion:

Desarrollado por OCLC (*Online Computer Library Center*, Centro de Biblioteca de Computadoras en Línea en español), cuyo principal objetivo es demostrar la efectividad de la utilización de esquemas de clasificación y encabezamientos de materia para la indización automatizada. Actualmente su centro de atención primordial es la categorización de recursos electrónicos utilizando la DDC (*Dewey Decimal Classification*, Clasificación Decimal Dewey en español, es un sistema de clasificación de bibliotecas creado en 1876, y desde ese momento ha sido enormemente modificado y ampliado).

La base de datos inicial utilizada en este proyecto está formada por los registros empleados en la Edición Electrónica de la DDC para el sistema operativo *Windows*. En esta base de datos se tiene en cuenta la naturaleza jerárquica de las relaciones conceptuales de la DDC de manera que, a la información sobre el significado de cada notación concreta se asocia la información sobre la clase a la que pertenece consiguiendo así contextualizar cualquier término. A partir de esta base de datos se han utilizado dos para ponderar y calcular la similaridad entre los términos extraídos de los documentos para representar su contenido y los términos utilizados para representar los conceptos de los registro. Comienza con una definición de los registros o *cluster*, y trata los documentos como una búsqueda, intentando ubicarlos en alguno de ellos". Para mejorar los resultados se han desarrollado diferentes filtros que permitan eliminar

los registros demasiado genéricos o demasiado específicos dentro de una clase determinada (OCLC, 2015).

El proyecto GERHARD:

La Biblioteca de la Universidad de Oldenburg en Alemania lleva a cabo el proyecto GERHARD (*German Harvest Automates Retrieval and Directory*), cuya financiación corre a cargo del Deutsche Forschugsgemeinschaft. GERHARD utiliza un sistema similar al del Nordic WAIS/WWW pero con métodos lingüísticos más avanzados y un grupo de documentos más heterogéneos. El proceso automatizado de categorización tiene dos partes: el análisis lingüístico automatizado y la comparación con una versión de la Clasificación Decimal Universal (CDU) específicamente preparada. El lenguaje natural de las páginas HTML se segmenta en entidades, palabras y frases, y se compara con un diccionario creado a partir de las definiciones de las notaciones de la CDU. El resultado es un grupo de notaciones para cada documento, que se pondera y ordena automáticamente en función de la frecuencia de aparición y de la estructura del documento. Las partes más relevantes del documento se indizan junto con los códigos de clasificación, de forma que es posible utilizar conjuntamente un índice de términos de búsqueda y la estructura jerárquica de la CDU (Carstensenl, et al., 2000).

Conclusiones parciales.

Los sistemas estudiados para la categorización automática de documentos del ámbito internacional son de carácter privativo, financiado por poderosas instituciones, por lo que el acceso a estas herramientas se ve limitado por su costo. Sus algoritmos internos se basan en los esquemas de la CDU y la DDC. De carácter positivo los sistemas cuentan con gran cantidad de bases documentales categorizadas, además de realizar el proceso de categorización automatizada a través de la información de los campos palabras-clave y el contenido de los documentos.

<u>Nacionales</u>:

La investigación realizada en sistemas y plataformas para la web cubana (dominio .cu), (Lupa[6], C.U.B.A.[7]) permitió arribar a la conclusión de que no existe constancia de la implementación de un sistema para la

[6] http://lupa.upr.edu.cu/

categorización automática de documentos que permita optimizar el proceso de recuperación de información en dichos buscadores.

1.3 Algoritmos para la categorización automática de documentos.

Existen algoritmos propuestos para la categorización de documentos en ambientes supervisados, aunque la mayor parte de estos algoritmos no son específicamente para la categorización de documentos, sino para la categorización en general. Algunos de estos algoritmos se han utilizado con adaptaciones pertinentes para la categorización automática de documentos en ambientes supervisados.

Algoritmos más utilizados:

- Algoritmos probabilísticos
- Algoritmo de Rocchio
- Algoritmo del vecino más próximo y variantes
- Algoritmos basados en redes neuronales

1. Algoritmos Probabilísticos:

Se basan en la teoría probabilística, en especial en el teorema de Bayes. Éste permite estimar la probabilidad de un suceso a partir de la probabilidad de que ocurra otro suceso, del cual depende el primero. El algoritmo de este tipo más conocido, y también el más simple, es el denominado Naive Bayes. Básicamente, se trata de estimar la probabilidad de que un documento pertenezca a una categoría. Dicha pertenencia depende de la posesión de una serie de características, de cada una de las cuales se conoce la probabilidad de que aparezcan en los documentos que pertenecen a la categoría en cuestión (Figuerola, et al., 2004).

2. Algoritmo de Rocchio:

El llamado algoritmo de Rocchio (Rocchio, 1971) es bien conocido y aplicado en la realimentación de consultas. En este ámbito, la idea es simple: formulada y ejecutada una primera consulta, el usuario examina los documentos devueltos y determina cuáles le resultan relevantes y cuáles no. Con estos datos, el sistema genera automáticamente una nueva consulta, basándose en los documentos que el

[7] http://www.redcuba.cu/

usuario señaló como relevantes o no relevantes. En este contexto, el algoritmo de Rocchio proporciona un sistema para construir el vector de la nueva consulta, recalculando los pesos de los términos de ésta y aplicando un coeficiente al peso de la consulta inicial, otro a los documentos relevantes y otro distinto a los que no son relevantes. En el ámbito de la categorización, el mismo algoritmo de Rocchio proporciona un sistema para construir los patrones de cada una de las clases o categorías de documentos. Así, partiendo de una colección de entrenamiento, categorizada manualmente de antemano, y aplicando el modelo vectorial, se construyen vectores patrón para cada una de las clases, considerando como ejemplos positivos los documentos de entrenamiento de esa categoría, y como ejemplos negativos los de las restantes categorías.

3. Algoritmo del vecino más próximo y variantes:

El algoritmo del vecino más próximo (*Nearest Neighbour, NN*) es uno de los más sencillos de implementar. La idea básica es como sigue: se calcula la similitud entre el documento a clasificar y cada uno de los documentos de entrenamiento, el más parecido estará indicando a qué clase o categoría debe asignarse el documento que se desea categorizar.

Desde un punto de vista más práctico, el vecino más próximo puede aplicarse con cualquier programa de recuperación de tipo mejor coincidencia (*Best Match*). Lo más frecuente es utilizar alguno basado en el modelo vectorial, pero, en puridad, esto no es imprescindible. Lo necesario es que sea *best match* y no de comparación exacta, como pueda ser el caso de los *booleanos*; el algoritmo se basa en localizar el documento más similar o parecido al que se desea clasificar. Para esto no hay más que utilizar ese documento como si fuera una consulta sobre la colección de entrenamiento. Una vez localizado el documento de entrenamiento más similar, dado que éstos han sido previamente categorizados manualmente, se conoce a qué categoría pertenece y, por ende, a qué categoría se debe asignar el documento que se está clasificando. Una de las variantes más conocidas de este algoritmo es la del vecino más próximo o KNN (*K-Nearest Neighbour*) que consiste en tomar los k documentos más parecidos, en lugar de sólo el primero. Como en esos k documentos los habrá, presumiblente, de varias categorías, se suman los coeficientes de los de cada una de ellas. La que más puntos acumule, será la candidata idónea (Figuerola, et al., 2004).

4. Redes Neuronales:

Las redes neuronales en general han sido propuestas en numerosas ocasiones como instrumentos útiles para la Recuperación de Información y también para la categorización automática. Básicamente, una red neuronal consta de varias capas de unidades de procesamiento o neuronas interconectadas; en el ámbito que ocupa la capa de entrada recibe términos, mientras que las unidades o neuronas de la capa de salida mapea clases o categorías. Las interconexiones tienen pesos, es decir, un coeficiente que expresa la mayor o menor fuerza de la conexión. Es posible entrenar una red para que, dada una entrada determinada (los términos de un documento), produzca la salida deseada (la clase que corresponde a ese documento). El proceso de entrenamiento consta de un ajuste de los pesos de las interconexiones, a fin de que la salida sea la deseada (Figuerola, et al., 2004).

Selección del Algoritmo:

El estudio de los sistemas homólogos dictamina la necesidad de implementar un categorizador automático de documentos para el buscador Orión por la ausencia de esta herramienta en los sistemas nacionales y el costo adquisitivo de los sistemas internacionales.

La categorización automática de documentos puede conseguirse mediante diversas técnicas. Se han revisado algunos de los algoritmos más frecuentes en categorización supervisada de documentos; al margen de mayor o menor complejidad de cada uno de ellos, se trata de técnicas suficientemente maduras, cuyos resultados son equiparables a los conseguidos por clasificadores inteligentes, es decir, personas.

Teniendo en cuenta la investigación de los algoritmos más utilizados para la categorización de documentos y consultar sus resultados experimentales (Tarragó, 2014), se arribó a la conclusión de que todos los algorimos cumplen satifactoriamente el proceso de categorización automática de documentos variando en su complejidad computacional. Se decide elegir el algoritmo probabilístico más conocido y fundamentado **Naive Bayes** en su variante **Multinominal.**

Naive Bayes es uno de los modelos probabilistas más simples y usados en categorización de texto porque produce resultados precisos como otros modelos más sofisticados. Se basa en la aplicación de la Regla de Bayes para predecir la probabilidad condicional de que un documento pertenezca a una clase a partir de la probabilidad de los documentos dada la clase y la probabilidad a priori de la clase en el conjunto de entrenamiento. Adicionalmente, el modelo Naive Bayes Multinominal considera la frecuencia de aparición de cada término en los documentos en vez de una ocurrencia binaria a diferencia del algoritmo clásico

Naive Bayes. Sus condiciones de uso son conjuntos de entrenamientos bien estructuradas y atributos razonablemente independientes.

1.4 Ambiente de Desarrollo.

Para el desarrollo del sistema para la categorización de documentos en el buscador Orión se definió el uso de las siguientes tecnologías y herramientas, las cuales constituyen un requisito para desarrollar las nuevas funcionalidades propuestas en el siguiente capítulo de la investigación.

Lenguaje de programación.

Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que puedan ser llevados a cabo por equipos electrónicos o computadoras. Están formados por conjuntos de símbolos, reglas sintácticas y semánticas que definen su estructura. Son usados para crear programas capaces de controlar el comportamiento físico y lógico de una máquina así como expresar algoritmos con precisión (Wilson, 1993). Para el desarrollo del sistema propuesto se utilizó lenguaje de programación Java en su versión 8.

- **Java 8:**

Java es un lenguaje de programación orientado a objetos, 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. Los programas escritos en el lenguaje Java pueden ejecutarse en cualquier tipo de hardware[8]. La recolección de basura de Java es un proceso prácticamente invisible al desarrollador, es decir, el programador no tiene conciencia de cuándo la recolección de basura tendrá lugar, ya que ésta no tiene necesariamente que guardar relación con las acciones que realiza el código fuente (Java, 2009).

Bibliotecas.

Biblioteca (*library*) en la rama de la informática es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación con una interfaz bien definida para la funcionalidad que se le

[8] La RAE define al hardware como el conjunto de los componentes que conforman la parte material (física) de una computadora.

invoca. Su fin es ser utilizada por otros programas, independientes y de forma simultánea. A continuación se describen las bibliotecas utilizadas en el desarrollo del sistema propuesto (Alegsa, 2010).

- **SolrJ 4.10.3:**

SolrJ es una biblioteca que permite la comunicación entre las aplicaciones Java y Solr. SolrJ engloba una gran cantidad de detalles referentes a la conexión a Solr y permite su aplicación para interactuar con Solr con métodos simples de alto nivel. El centro de SolrJ es el paquete org.apache.solr.client.solrj, que contiene sólo cinco clases principales (Confluence, 2015). La biblioteca es utilizada fundamentalmente para la interacción y comunicación con los índices generados en Solr en el desarrollo del sistema para la categorización de documentos.

Entorno de desarrollo integrado

Un entorno de desarrollo integrado o IDE (acrónimo en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. 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. A continuación se exponen las características principales del IDE a utilizar.

- **Netbeans 8.0:**

NetBeans IDE es un entorno de desarrollo integrado, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java cumpliendo con el lenguaje propuesto para el desarrollo del sistema, aunque puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Otros aspectos considerados son que NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Herramienta de modelado.

Las herramientas CASE (*Computer Aided Software Engineering* por sus siglas en inglés, Ingeniería de software asistidas por computadoras) son un grupo de diversas aplicaciones informáticas o programas informáticos destinados a elevar la productividad del desarrollo de software reduciendo el costo del mismo

en cuanto a tiempo y dinero, para el desarrollo del sistema propuesto se definió la herramienta Visual Paradigm en su versión 8.0 y el lenguaje de modelado UML en su versión 2.4.1.

- **Visual Paradigm 8.0:**

Visual Paradigm es una herramienta de software diseñada para que los equipos de desarrollo de software puedan modelar el sistema de información empresarial y gestionar procesos de desarrollo. Visual Paradigm soporta los principales lenguajes y estándares de modelado de la industria, como UML, SysML, SoaML, BPMN, XMI, etc. Ofrece herramientas completas de software para la captura de requisitos, análisis de procesos, diseño de sistemas, diseño de bases de datos, y otras muchas actividades (Visual Paradigm, 2015). La herramienta será empleada para el diseño metodológico documentado y generando información sobre el sistema a desarrollar.

- **Lenguaje de Modelado UML 2.4.1:**

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. Este lenguaje se centra en la representación gráfica de un sistema. Este lenguaje indica cómo crear y leer los modelos. Esto último es el objetivo de las metodologías de desarrollo. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones como visualizar, especificar, construir y documentar información de forma gráfica para los sistemas en desarrollo.

Herramienta para indexar información.

En Orión se utiliza Solr en su versión 4.10.3 como componente de indexación por las múltiples ventajas que ofrece este software, a continuación se exponen las principales características de esta herramienta.

- **Solr 4.10.3:**

Apache Solr es un popular motor de búsquedas de código abierto que proporciona potentes funcionalidades de búsqueda y navegación por facetas, es decir, explorar la información desde diversas perspectivas. Tales funcionalidades pueden ser difíciles de implementar sobre una base de datos relacional. Solr es capaz de manejar complejos criterios de búsqueda, corrige ortografía en las consultas, permite realzar resultados, configurar la relevancia de términos, entre otras funcionalidades además de contar con bibliografía referente para el estudio de su funcionamiento (Estrada, 2012).

Solr está escrito en Java y se ejecuta como un servidor de búsqueda de texto completo independiente dentro de un servidor web. Solr Lucene utiliza la biblioteca Java de búsqueda en su base para la indexación de texto completo y de búsqueda, y tiene como REST HTTP / XML y JSON, APIs[9] que hacen que sea fácil de utilizar desde prácticamente cualquier lenguaje de programación (Apache, 2015).

Herramientas para la Minería de Datos.

Para el desarrollo del sistema de categorización de documentos se realizó un estudio sobre tres herramientas para el trabajo enfocado en la Minería de Datos (Knime, RapidMiner, Weka), basándose en sus características, funcionamientos y posibilidades. Son herramientas de software de código abierto para minería de datos que pueden ser obtenidas de forma gratuita.

- **Knime.**

Está desarrollado sobre la plataforma Eclipse y programado en Java, su uso se basa en el diseño de un flujo de ejecución que plasme las distintas etapas de un proyecto de minería de datos y predecir posibles resultados. Es una plataforma de código abierto de fácil uso y comprensible para integración de datos, procesamiento, análisis, y exploración. Ofrece a los usuarios la capacidad de crear de forma visual flujos de datos, ejecutar selectivamente algunos o todos los pasos de análisis, y luego estudiar los resultados, modelos y vistas interactivas (KNIME, 2017).

- **RapidMiner.**

Es un ambiente de experimentos en aprendizaje automático y minería de datos que se utiliza para tareas de minería de datos tanto en investigación como en el mundo real. Permite a los experimentos componerse de un gran número de operadores anidables arbitrariamente, que se detallan en archivos XML y se hacen con la interfaz gráfica de usuario de RapidMiner. Ofrece más de 500 operadores para todos los principales procedimientos de máquina de aprendizaje, y también combina esquemas de aprendizaje y evaluadores de atributos del entorno de aprendizaje Weka. Está disponible como una herramienta autónoma para el análisis de datos y como motor para minería de datos que puede integrarse en sus propios productos (Rapidminer, 2016).

[9] Es un conjunto de subrutinas, funciones y procedimientos en la programación orientada a objetos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- **Weka.**

Entorno Waikato para el Análisis del Conocimiento (Weka) es una conocida suite de software para máquinas de aprendizaje que soporta varias tareas típicas de minería de datos, especialmente pre procesamiento de datos, agrupamiento, clasificación, regresión, visualización y características de selección. Sus técnicas se basan en la hipótesis de que los datos están disponibles en un único archivo plano o relación, donde cada punto marcado es etiquetado por un número fijo de atributos. WEKA proporciona acceso a bases de datos SQL[10] utilizando conectividad de bases de datos Java y puede procesar el resultado devuelto como una consulta de base de datos. Su interfaz de usuario principal es el *explorer*, pero la misma funcionalidad puede ser accedida desde la línea de comandos o a través de la interfaz de flujo de conocimientos basada en componentes (Weka, 2016).

Tabla 1: *Comparativa de las herramientas para la minería de datos* (Jaramillo, et al., 2015).

Características	RapidMiner	Weka	Knime
Licencia Libre	Si	Si	Si
Multiplataforma	Si	Si	Si
Modificar modelos	Si	Si	**No**
Técnicas Descriptivas (agrupación)	Si	Si	Si
Técnicas Predictivas (clasificación)	Si	Si	**No**
Visualización de datos	Si	Si	Si
Filtros	Si	Si	**No**
Conversión de datos	Si	Si	**No**
Procesamiento de datos	Si	Si	Si

Dado que las herramientas seleccionadas en su mayoría comparten características similares se ha optado por realizar la comparativa solamente de los algoritmos de minería de datos que integran, dejando a un lado las opciones de pre-procesamiento, post-procesado y visualización en la siguiente tabla comparativa.

[10] **SQL** (S*tructured Query Language*; en español lenguaje de consulta estructurada)

Tabla 1.1: *Comparativa de los algoritmos en las herramientas para la minería de datos* (Dataprix, 2010).

Datos	RapidMiner	Weka	Knime
Algoritmos implementados de forma nativa	34	**168**	9
Algoritmos exportados desde Weka	101	**0**	102
Total de algoritmos implementados	135	**168**	111
% de algoritmos nativos	25.2%	**100%**	8.1%

Selección de la herramienta.

Con el estudio realizado sobre las herramientas para la minería de datos, se descarta el uso de la herramienta Knime por la ausencia de técnicas predictivas, lo que imposibilita la tarea de categorizar información. Las herramientas RapidMIner y Weka comparten características comunes y cumplen con el marco de la investigación ya que están desarrolladas en lenguaje Java y son software de código abierto. Se selecciona la herramienta Weka por su vasta colección de algoritmos nativos relevantes para el desarrollo de sistema a proponer.

1.5 Metodologías de desarrollo.

Particularmente en ingeniería de software la metodología de desarrollo es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Las metodologías de desarrollo se clasifican dentro de dos grandes categorías: las metodologías tradicionales las cuales están diseñadas para proyectos de gran envergadura y las metodologías ágiles, utilizadas para proyectos de poca envergadura y dimensiones, como es el caso del sistema propuesto para el desarrollo del categorizador automático de documentos. Cada metodología engloba características que las diferencia de las otras para ser empleada en proyectos de desarrollo. Para la construcción del sistema de categorización de documentos se realizó un estudio sobre tres metodologías de desarrollo ágiles (XP, Open UP, AUP-UCI) para definir cuál responde a las características del sistema a desarrollar.

eXtreme Programming (XP)

La programación extrema o *eXtreme Programming* (XP) es una metodología de desarrollo de la ingeniería de software más destacado de los procesos ágiles. La programación extrema se diferencia de las

metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad (Bustamante, et al., 2014).

Los pasos fundamentales inmersos en las fases del método son:

- **Desarrollo iterativo e incremental:** Pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas:** Son frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Programación en parejas:** Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que se obtenga mayor calidad del código escrito de esta manera, ya que el código es revisado y discutido mientras se escribe, es más importante que la posible pérdida de productividad inmediata.
- **Frecuente integración del equipo de programación con el cliente o usuario:** Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo. Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- **Refactorización del código:** Reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartido:** En vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad del código:** Cuando todo funcione se podrá añadir alguna funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Open Unified Process (OpenUP)

El OpenUP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto, no provee lineamientos para todos los elementos que se manejan en un proyecto pero tiene los componentes básicos que pueden servir de base a procesos específicos. La

mayoría de los elementos de OpenUP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances (Eclipse Process Framework, 2016).

Principios del OpenUP:

- Colaborar para sincronizar intereses y compartir conocimiento. Este principio promueve prácticas que impulsan un ambiente de equipo saludable, facilitan la colaboración y desarrollan un conocimiento compartido del proyecto.
- Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto. Este principio promueve prácticas que permiten a los participantes de los proyectos desarrollar una solución que maximice los beneficios obtenidos por los participantes y que cumple con los requisitos y restricciones del proyecto.
- Centrarse en la arquitectura de forma temprana para minimizar el riesgo y organizar el desarrollo.
- Desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo. Este principio promueve prácticas que permiten a los equipos de desarrollo obtener retroalimentación temprana y continua de los participantes del proyecto, permitiendo demostrarles incrementos progresivos en la funcionalidad.

Agile Unified Process (AUP)

El proceso unificado (Unified Process UP) es un marco de desarrollo de software iterativo e incremental. A menudo, es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software. Dado que es un marco de procesos, puede ser adaptado y la más conocida es RUP (Rational Unified Process) de IBM. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas identificando los riesgos desde etapas iniciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos clave del producto y que determinan los riesgos técnicos (Flores, 2015).

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados:

- **Concepción**: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- **Elaboración**: El objetivo es que el equipo de desarrollo profundice en la comprensión de revisar los requisitos del sistema y en validar la arquitectura.
- **Construcción**: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- **Transición**: el sistema se lleva a los entornos de pre-producción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Selección de la metodología de desarrollo de software a utilizar.

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para el desarrollo del sistema propuesto. Se seleccionó la metodología AUP en una versión desarrollada por la UCI.

Las fases de AUP-UCI son: inicio, ejecución y cierre. Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación. En la segunda fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto, considerando los requisitos y la arquitectura. En la fase de cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP-UCI define cuatro escenarios para modelar el sistema dependiendo de la complejidad y características del proceso de desarrollo. El escenario número 4 de esta metodología se adapta al desarrollo del sistema propuesto: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y

validarlos. Se recomienda en proyectos no muy extensos, ya que una Historia de Usuario (HU) no debe poseer demasiada información (Universidad de las Ciencias Informáticas, 2015).

Conclusiones del capítulo.

La investigación de los principales elementos teóricos sobre los categorizadores automáticos de documentos, así como la importancia de estos para la RI brindan los materiales necesarios para el desarrollo de una propuesta de solución y el arribo a las siguientes conclusiones:

1. El estudio realizado sobre los sistemas homólogos del ámbito internacional brindan resultados concretos sobre sus funcionamientos y las principales técnicas empleadas, son sistemas privativos lo que dificulta su usabilidad, en cambio, en el ámbito nacional no existe ningún motor de búsqueda web que contenga implementado un sistema de categorización automática de documentos.
2. Mediante la investigación sobre los algoritmos más utilizados a escala mundial para la categorización de documentos el autor seleccionó el algoritmo probabilístico **Naive Bayes** por sus características y su funcionamiento práctico, se define un ambiente supervisado con categorías definidas para el desarrollo del sistema de categorización de documentos.
3. Entre las distintas metodologías estudiadas se arribó que la metodología **AUP-UCI** es la más conveniente para el desarrollo del presente trabajo de diploma ajustándose al escenario 4 que plantea dicha metodología por ser un proyecto pequeño y contar con el cliente para convenir los detalles de los requisitos a desarrollar.
4. Fue confeccionado el ambiente de desarrollo, seleccionadas las herramientas y lenguajes a utilizar para el desarrollo del sistema basándose en sus rendimientos y acoplamiento para el sistema propuesto con el buscador Orión.

El Capítulo 2: “Análisis y diseño del sistema de categorización de documentos”

Enfocándose en las características de los sistemas de categorización de documentos y los componentes que plantea la metodología AUP-UCI, se expone la propuesta de diseño del sistema a desarrollar. Se especifican las funcionalidades del sistema con la descripción de los requisitos funcionales, no funcionales y las historias de usuario como principal artefacto generado. Se define la arquitectura a utilizar, el diagrama de clases con los patrones de diseños utilizados y el diagrama de despliegue del sistema.

2.1 Diagrama de Modelo de Dominio.

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software (Larman, 2003). A continuación se muestra el diagrama de modelo de dominio del sistema de categorización de documentos para el buscador Orión:

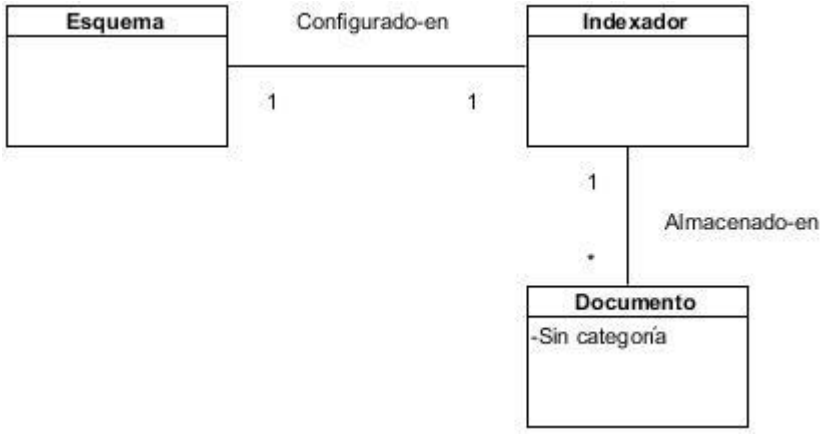


Figura 1: *Diagrama de modelo de dominio del Sistema de categorización de documentos. Elaboración propia.*

Descripción de los objetos del modelo de dominio:

- **Esquema:** Archivo de configuración por donde se rige el rastreador e indexador para generar el índice de los documentos indexados.
- **Indexador:** Herramienta encargada de indexar los documentos encontrados por el rastreador.
- **Documento:** Contiene los campos definidos por el esquema.

- **Sin categoría:** No contiene la categoría de los documentos indexados.

2.2 Propuesta de Solución.

En la presente investigación se propone el desarrollo de un sistema con funcionalidades que permitan, consultar documentos indexados en Solr, adicionándole el campo categoría al cuerpo de los documentos y categorizarlos atendiendo a la información del campo contenido consumiendo el API de Weka y Solr. El desarrollo del sistema cuenta con 2 etapas:

Etapa 1: Se propone una nueva propuesta de diseño para el archivo ***schema.xml*** adicionándole el campo "**`my_category`**" a la estructura de los documentos a rastrear e indexar ya que ambas herramientas comparten el mismo archivo. La estructura del nuevo campo queda compuesta de la siguiente manera:

```
<field name= "my_category" type="string" indexed="true" stored= "true" default= "none">
```

`field name= "my_category":` Nombre del campo ("mi categoría", en español).
`type="string":` Tipo de dato que devuelve el campo creado (de tipo "palabra", en español).
`indexed="true":` Define si se desea que el campo sea indexado.
`stored= "true":` Define si se desea que el campo sea almacenado.
`default= "none":` Valor por defecto que devuelve el campo en los documentos indexados ("nada", en español).

Etapa 2: En el capítulo anterior se referencia la categorización supervisada de documentos, seleccionando este escenario para desarrollar el sistema propuesto, y se ofrecieron elementos técnicos para la comprensión y desarrollo del mismo. El sistema propuesto permite categorizar los documentos con información solo de naturaleza textual y en español ya que el buscador Orión responde al dominio nacional donde la mayor parte de los sitios indexados son en español. En caso de categorizar documentos en otro idioma sería necesario utilizar un archivo de *Stopwords*[11] y un *Stemmer*[12] de ese idioma en

[11] Palabras no relevantes o superfluas que no aportan significado para la categorización (artículos, preposiciones, conjunciones, entre otras.)
[12] Es el proceso de reducir las palabras inflexas (o algunas veces derivadas) a su forma de palabra, base o raíz (Frakes, 1992).

específico. Para ello una vez que se modifique el archivo de configuración **schema.xml** y los documentos sean indexados con el nuevo campo "*my_category*", el índice generado por Solr será consultado a través de la biblioteca **solrj,** obteniendo así de los documentos los campos **id** (identificador), ***keyword*** (Palabras Claves) y ***content*** (contenido). Luego se realizará el procesamiento de los datos, comienza eliminando las palabras vacías de cada documento referenciado en un archivo nombrado *StopwordsES*, consumido por el sistema, creando así una cadena de palabras relevantes para la categorización, separada por espacio y en letra minúscula. Consumiendo el API de la herramienta Weka la cadena de palabras relevantes son convertidas a su raíz a través de un *Stemmer* para palabras en español. Para categorizar los documentos se utiliza el algoritmo ***Naive Bayes*** en su versión ***Multinominal*** ya que este tiene en cuenta las repeticiones de las palabras; para esto el sistema consume la colección de entrenamiento y las categorías o clases definidas que pueden poseer los documentos. El sistema culminaría cuando el campo "*my_category*" es actualizado con los resultados arrojados por el sistema en el servidor de indexación Solr, a través del campo **Id** que es único para cada documento.

2.3 Artefactos.

El proceso es dirigido por la metodología de desarrollo de software AUP-UCI, según esta metodología los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido se ajustan al escenario 4 que rige esta metodología, además de ser utilizado este escenario para proyectos pequeños. Cada requisito es descrito mediante Historia de Usuario, por lo que se convierte en el principal artefacto generado durante el diseño del sistema.

2.4 Requisitos del Sistema.

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software (Pressman, 2006). Sus clasificaciones son:

- **Requisitos funcionales (RF):** Describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema.

- **Requisitos no funcionales (RNF):** Describen atributos sólo del sistema o del ambiente del sistema que no están relacionados directamente con los requisitos funcionales. Los requisitos no funcionales incluyen restricciones cuantitativas, como el tiempo de respuesta o precisión, tipo de plataforma (lenguajes de programación y/o sistemas operativos).

Con el objetivo de establecer un entendimiento común entre el usuario y el proyecto de software la tabla a continuación muestra el listado de los requisitos funcionales según su prioridad (alta, media y baja).

Tabla 2: *Listado de requisitos funcionales. Elaboración propia.*

No. RF	Nombre	Descripción	Prioridad
1	Cargar colección de entrenamiento.	Entrenar categorizador y cargar fichero con la colección de entrenamiento.	Alta
2	Obtener documentos de Solr.	Establecer flujo de comunicación con el servidor Solr.	Alta
3	Pre-procesamiento de los documentos.	Pre-procesamiento de los documentos obtenidos del servidor de indexación Solr.	Baja
4	Categorizar documentos.	Proceso de categorizar los documentos atendiendo al algoritmo Naive Bayes Multinominal.	Alta
5	Actualizar campo categoría en el servidor Solr.	Actualizar campo categoría con los resultados arrojados por el categorizador.	Media

Tabla 3: *Listado de requisitos no funcionales. Elaboración propia.*

No.	Nombre del requisito no funcional	Atributo
RNF1	Emplear como lenguaje de programación Java.	Diseño e Implementación
RNF2	El IDE que se debe utilizar es Netbeans 8.0.	Diseño e Implementación

RNF3	La aplicación se ejecutará en cualquier sistema operativo que soporte las librerías Java 8.	Funcionalidad
RNF4	Utilizar Solr como motor de búsqueda e indexador.	Funcionalidad
RNF5	Para un correcto funcionamiento del sistema de categorización de documentos se requieren requisitos mínimos en el servidor: 2Gb de memoria RAM, CPU con 2 *cores* de procesamiento a velocidad de 2.5 GHz y 5 Gb de espacio libre en disco.	Hardware

2.5 Historias de Usuario.

Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad (Izaurralde, 2013).

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

Tabla 4: *Historia de usuario #1. Elaboración propia.*

Número: 1	Nombre del Requisitos: Cargar colección de entrenamientos
Programador: Jorge Ignacio Martínez Ravelo	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 15 días
Riesgo: Alto	Tiempo Real: 15 días
Para la puesta en marcha del categorizador automático de documentos es necesario cargar por el sistema la colección de entrenamiento almacenado en un fichero (dataFile.arff) desde el directorio raíz. Cuando se realizan modificaciones a este fichero o es utilizado otro fichero (pasado por parámetro al categorizador), es necesario realizar un proceso de entrenamiento en el categorizador y salvar las nuevas configuraciones para su posterior uso.	
Observaciones: Se define la etiqueta **–l** para cargar la colección de entrenamiento por defecto.	

Se define la etiqueta **–t** para entrenar el categorizador.
Se define la etiqueta **–s** para salvar el categorizador entrenado.
Ejemplo: Base documental por defecto en el directorio raíz.
`java –jar categorizer-1.0-SNAPSHOT.jar –t –s -l –cat /*`
Ejemplo: Base documental en el directorio raíz con nombre y extensión **File.arff.**
`java –jar categorizer-1.0-SNAPSHOT.jar –t File.arff –s –l –cat /*`

Prototipo de Interface:

Tabla 5: *Historia de usuario #2. Elaboración propia.*

Número: 2	Nombre del Requisitos: Obtener documentos de Solr.
Programador: Jorge Ignacio Martínez Ravelo	Iteración Asignada:1
Prioridad: Alta	Tiempo Estimado: 5 días
Riesgo: Alto	Tiempo Real: 5 días
Establecer flujo de información entre el sistema de categorización de documentos y el servidor de indexación Solr. Permite consultar los documentos indexados en el servidor obteniendo los valores de los campos id, palabras clave y contenido.	
Observaciones: Se define la etiqueta **–solr** para establecer comunicación con el servidor de indexación Solr.	
Prototipo de Interface:	

Tabla 6: *Historia de usuario #3. Elaboración propia.*

Número: 3	Nombre del Requisitos: Pre-procesamiento de los documentos.
Programador: Jorge Ignacio Martínez Ravelo	Iteración Asignada:1
Prioridad: Baja	Tiempo Estimado: 15 días
Riesgo: Baja	Tiempo Real: 15 días
El pre-procesamiento de los documentos comienza con la eliminación de todos los caracteres innecesarios como los números, luego procede la eliminación de las palabras superfluas,	

vacías o *"**Stopword**"*, no son más que las palabras que no aportan información para la categorización de documentos como los artículos, preposiciones, conjunciones entre otras palabras a través de un archivo (***stopwordsES***) consumido por el sistema en su directorio raíz. A continuación, se almacenan en una cadena las palabras transformadas en minúscula.
Observaciones:
Prototipo de Interface:

Tabla 7: *Historia de usuario #4. Elaboración propia.*

Número: 4	Nombre del Requisitos: Categorizar documentos.
Programador: Jorge Ignacio Martínez Ravelo	Iteración Asignada:1
Prioridad: Alta	Tiempo Estimado: 21 días
Riesgo: Alta	Tiempo Real: 21 días
Con los datos pre-procesados el sistema consume los servicios de la biblioteca Weka para convertir las palabras a su raíz a través de **`SnowballStemmer ("spanish"),`** para documentos en español. La representación de los documentos en vectores, se consigue a través del filtro **`StringToWordVector().`** El algoritmo Naive Bayes Multinominal es el encargado de procesar los datos y compararlos con la colección de entrenamiento para definir la o las categorías a la que pertenece cada documento conjuntamente con la probabilidad que cuenta el documento de pertenecer a cada clase.	
Observaciones: Se define la etiqueta **–cat** para categorizar los documentos indexados. **Ejemplo:** Categorizar todos los documentos indexados en el servidor. `java –jar categorizer-1.0-SNAPSHOT.jar -l` **`–cat /*`** **Ejemplo:** Categorizar documentos de subdominios de interés. (cubadebate) `java –jar categorizer-1.0-SNAPSHOT.jar -l` **`–cat cubadebate`**	
Prototipo de Interface:	

Tabla 8: *Historia de usuario #5. Elaboración propia.*

Número: 5	Nombre del Requisitos: Actualizar campo categoría en el servidor Solr.
Programador: Jorge Ignacio Martínez Ravelo	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 5 días
Riesgo: Media	Tiempo Real: 5 días
Actualizar el nuevo campo *"my_category"* en los documentos indexados en el servidor Solr luego del proceso de categorización atendiendo al identificador (id) único que presenta cada documento.	
Observaciones:	
Prototipo de Interface:	

2.6 Arquitectura del sistema.

La arquitectura de software demuestra la organización, funcionamiento y conexión entre las partes de un software. Su objetivo principal es garantizar un mejor desempeño en el desarrollo de las aplicaciones. Proporciona robustez, portabilidad y flexibilidad a la aplicación. Es considerado el elemento de enlace entre los requerimientos y la implementación del sistema (Gamma, et al., 2010).

Para el desarrollo del sistema de categorización de documentos fue seleccionada la arquitectura basada en componente. Esta arquitectura consiste en una rama de la Ingeniería de software en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software. El principal elemento de software dentro de una arquitectura basada en componentes son precisamente los componentes de software. Existen 5 principios diseñados (Messerschmitt, et al., 2003), que definen a un componente de software como elemento de la arquitectura:

- **Múltiple uso:** se refiere al hecho de que un componente es escrito dentro de un contexto que permita que su funcionalidad sea útil en la creación de distintas piezas de software.
- **Contexto no específico**: en relación con la orientación conceptual de la especificación de un componente, debe estar planteada de una forma general que permita su adaptación en distintos sistemas, sin que el contexto tenga prioridad.

- **Encapsulación:** se refiere a la especificación interna oculta o no investigable a través de la interface. Así se protege que el resto de componentes y piezas de software dentro de un sistema, no se vean afectados por cambios en el diseño de uno de los componentes.
- **Una unidad independiente de desarrollo con su propio control de versiones:** este principio muy relacionado con la encapsulación, permite que un componente pueda ser desarrollado de manera independiente, cambiando el diseño o agregando nuevas funcionalidades, sin afectar significativamente el resto del sistema.

La estructura de la arquitectura basada en componentes contempla 3 partes:

1. **El nombre de los componentes**: el nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como software. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura.
2. **La interface de los componentes:** es el área de intercambio (input-output) entre el interior y el exterior de un componente de software. La interface es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación).
3. **Cuerpo y código de implementación:** es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

A continuación se muestra el diagrama de arquitectura del sistema de categorización de documentos.

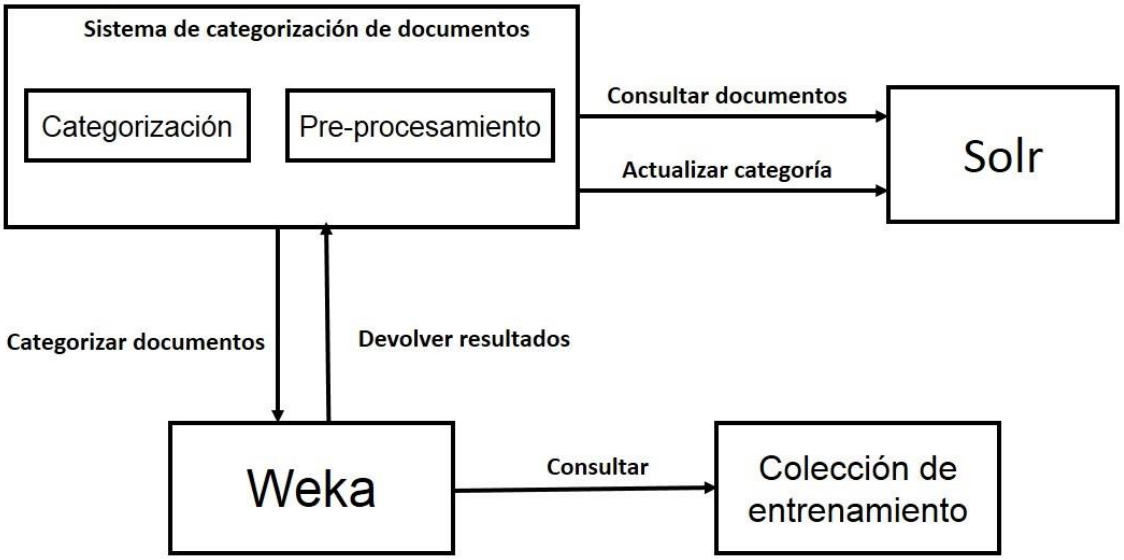


Figura 2: *Diagrama de Arquitectura del sistema. Elaboración propia.*

El **componente Solr** es el encargado del ejecutar el proceso de indexado de los documentos para ser almacenados y a su vez generar información. El sistema de **categorización automática de documentos**, es el encargado de consultar los documentos indexados para el **pre-procesamiento** de los datos y su posterior **categorización** con la ayuda de la herramienta **Weka**, que a su vez se apoya en la **colección de entrenamiento** para devolver resultados**.**

2.7 Diagrama de clases.

El diagrama de clases describe gráficamente las especificaciones de las clases de software (Larman, 2003). En otras palabras, se evidencian las clases con sus relaciones. En el diagrama de clases correspondiente al sistema a desarrollar (ver figura 3) se evidencia la pertenencia de las clases con sus relaciones.

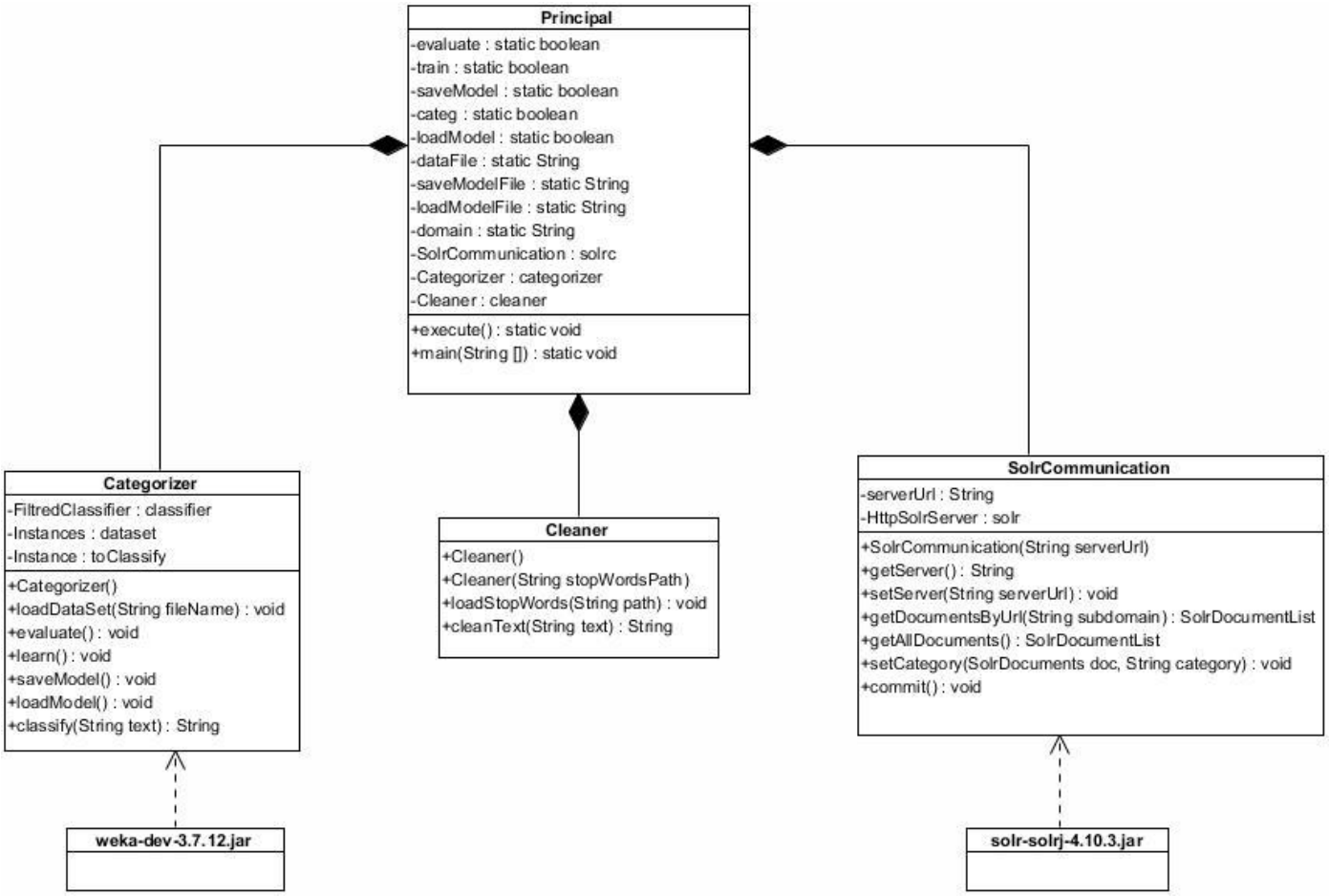


Figura 3: *Diagrama de Clases de Diseño del Sistema de categorización de documentos. Elaboración propia.*

- **`Cleaner`:** Es la clase encargada de consultar el archivo ***"stopwordsES"*** con las palabras irrelevantes para realizar el pre-procesamiento de los documentos consultados.
- **`SolrCommunication`:** Es la clase encargada del flujo de comunicación con la herramienta Solr apoyado por la biblioteca solrj, realiza consultas a subdominios de interés o todas las URL indexadas.
- **`Categorize`:** Es la clase encargada de categorizar los documentos, consume el API de la herramienta Weka.

- **`Principal`:** La clase principal es el encargado de establecer el flujo de datos entre las 3 clases creando los objetos de las clases anteriores y realizar las funcionalidades propuestas.

2.8 Patrones de diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Gamma, et al., 2010). En el desarrollo del sistema de categorización de documentos se utilizan algunos patrones generales de software para asignación de responsabilidades (GRASP, por sus siglas en ingles) y los patrones pandilla de los cuatro (GoF, por sus siglas en ingles).

GRASP:

Según Larman (Larman, 2003), los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades. En el desarrollo del sistema propuesto se puede apreciar el uso de los siguientes patrones de diseño GRASP:

- **Experto:** El patrón Experto en Información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. El Experto no pretende ser una idea oscura o extravagante; expresa la "intuición" común de que los objetos hacen las cosas relacionadas con la información que tienen (Larman, 2003). Las clases que intervienen en el sistema de categorización de documentos cumplen con este patrón ya que todas son expertas en su función.

- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se

favorece el bajo acoplamiento (Larman, 2003). En el sistema de categorización de documentos en la clase **`Principal`** se evidencia este patrón ya que este es el que se encarga de crear los objetos necesarios de las otras clases existentes.

- **Bajo Acoplamiento:** El patrón Bajo Acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve los resultados negativos que pueden producir un acoplamiento alto. El Bajo Acoplamiento soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio. No se puede considerar de manera aislada a otros patrones como el Experto o el de Alta Cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección al asignar una responsabilidad (Larman, 2003). Las clases **`SolrCommunication`**, **`Categorizer`**, **`Cleaner`** no tienen relaciones entre ellas, solo se comunican con la clase **`Principal`**, pueden ser reutilizadas como componentes para otros sistemas sin que ocurran grandes impactos por los cambios.

- **Alta Cohesión:**
  La información que almacena una clase debe de ser coherente y debe estar relacionada con la clase (Larman, 2003). Las clases **`SolrCommunication`**, **`Categorizer`**, **`Cleaner`** utilizan dicho patrón ya que cada una de las clases es totalmente coherente con sus funciones.

GoF:

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro "*Design Patterns—Elements of Reusable Software*" de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro. A continuación se muestran los patrones identificados para el desarrollo del sistema propuesto.

- **Iterator (Iterador):** Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. Proporciona un acceso secuencial a una colección de objetos sin que los clientes se preocupen de la implementación de esta colección.

- **Memento (Recuerdo):** Permite volver a estados anteriores del sistema. Salvaguarda y restaura el estado de un objeto. El categorizador se guarda y restaura en un fichero como un objeto serializado.

2.9 Modelo de despliegue.

El modelo de despliegue es un tipo de UML que se utiliza para modelar la disposición física de los artefactos de *software* en nodos (usualmente plataforma de *hardware*). Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema (OMG-UML, 2007).

Como se puede apreciar en la ***Figura 4***, Se encuentra un servidor de indexación con la herramienta **Solr**, el cual a través de consultas se comunica con el servidor con el Sistema de categorización de documentos con la herramienta Weka, devolviendo los resultados al servidor de indexación por el protocolo HTTP.

Figura 4*: Diagrama de Despliegue del Sistema. Elaboración propia.*

Conclusiones del capítulo.

En el presente capítulo a partir del estudio realizado en el Capítulo 1 sobre los sistemas de categorización de documentos, se definieron un total de cinco funcionalidades a implementar, conformando la propuesta de solución, las cuales están organizadas en cinco Historias de Usuario. Se realizó la descripción de las Historias de Usuario lo que permitió un mejor entendimiento entre el programador y el cliente. Se explicó la arquitectura del sistema tomándose la arquitectura por componentes como base para el desarrollo del sistema y crear el diseño del diagrama de clases de la aplicación, lo que permitió describir los patrones de diseño utilizados, así como conocer la relación entre las clases.

El Capítulo 3: “Implementación y prueba del sistema de categorización de documentos”

En el presente capítulo se observa el diagrama de componentes y su descripción, con el fin de entender los elementos de software del sistema y sus relaciones. También se muestran los estándares de codificación y la validación del diseño realizado mediante distintas pruebas.

3.1 Estándar de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. A continuación se muestra el estándar de codificación utilizado para el desarrollo del sistema.

- Un archivo fuente contiene una única clase.
- Se usa el margen adicional de cuatro espacios.

```
public Cleaner() throws FileNotFoundException, IOException {
    loadStopWords("stopwordsES");
}
```

- Las variables se declaran independientes en líneas nuevas, no en las mismas sentencias.

```
public void loadStopWords(String path) throws FileNotFoundException, IOException{
    stopWordsSet = new HashSet();

    FileReader fr = new FileReader(path);
    BufferedReader br = new BufferedReader(fr);

    String line;
    while ((line = br.readLine()) != null){
        stopWordsSet.add(line);
    }
```

- La implementación de las clases tendrán el siguiente orden:
  1. Atributos de la clase.
  2. Constructores.
  3. Métodos de acceso.

4. Métodos.

```
public final class Cleaner {
    private Set<String> stopWordsSet;

    public Cleaner() throws FileNotFoundException, IOException {
        loadStopWords("stopwordsES");
    }

    public Cleaner(String stopWordsPath) throws FileNotFoundException, IOException{
        loadStopWords(stopWordsPath);
    }

    public void loadStopWords(String path) throws FileNotFoundException, IOException{
        stopWordsSet = new HashSet();

        FileReader fr = new FileReader(path);
        BufferedReader br = new BufferedReader(fr);

        String line;
        while ((line = br.readLine()) != null){
            stopWordsSet.add(line);
        }
    }
```

- Los atributos relacionados se declaran en una misma línea. El resto se declara en líneas separadas.

```
public class Principal {
    private static boolean evaluate, train, saveModel, categ, loadModel;
    private static String dataFile, saveModelFile, loadModelFile, domain;
    private static SolrCommunication solrc;
    private static Categorizer categorizer;
    private static Cleaner cleaner;
```

- Los comentarios están declarados en el mismo idioma, gramaticalmente correctos y contienen los signos de puntuación apropiados.

```
public static void execute() throws Exception{

    if(train){
        System.out.println("En Principal: llamando al metodo loadDataset de Categorizer.");
        categorizer.loadDataset(dataFile);
    }

    /*As recommended by WEKA documentation, the classifier is defined but
      not trained yet. Evaluation of previously trained classifiers can
      lead to unexpected results.*/
    if(evaluate){
        System.out.println("En Principal: llamando al metodo evaluate de Categorizer.");
        categorizer.evaluate();
    }
```

3.2 Diagrama de componentes.

Un diagrama de componentes proporciona una visión física de la construcción del sistema de información. Muestra la organización de los componentes de software, sus interfaces y las dependencias entre ellos (Cillero, 2016). A continuación se muestra el diagrama de componentes del sistema de categorización de documentos para el buscador Orión.

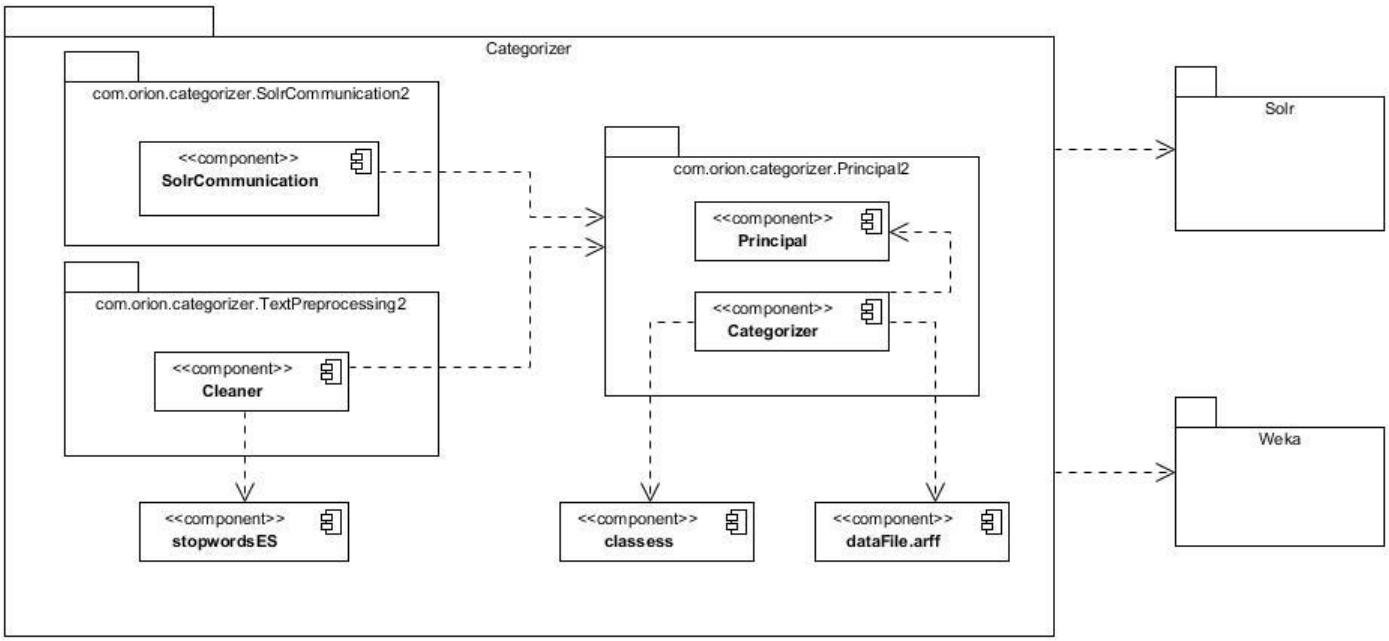


***Figura 5**: Diagrama de Componente del sistema de categorización de documentos. Elaboración propia.*

3.3 Colección de entrenamiento.

Las múltiples búsquedas realizadas en Internet no arrojaron resultados concretos sobre bases documentales categorizadas en idioma español para entrenar el sistema de categorización de documentos confeccionado. Debido a la situación antes planteada, el equipo de trabajo del Centro de Investigación y Desarrollo de Internet (CIDI) elaboró una base documental para el entrenamiento del categorizador, enfocado en documentos categorizados de uno de los sitios web informativos más importantes de Cuba, como es el caso de cubadebate.cu, un medio de expresión de periodistas cubanos y de otras nacionalidades agrupados en torno al círculo de periodistas contra el terrorismo, comprometidos con un espacio para el intercambio sobre el terrorismo y las campañas de difamación organizadas contra Cuba. Fuero seleccionadas un total de 7 de las 10 categorías que presenta el sitio: Política, Economía, Deporte, Salud, Cultura, Ciencia y tecnología y Medio Ambiente, exceptuando la categoría Medios ya que su contenido se basa en imágenes, la categoría Militar e Inteligencia cuyo contenido se asocia a la clase Política y la clase Sociedad la cual engloba contenidos muy similares a los de las categorías seleccionadas. El equipo de proyecto del buscador Orión desarrolló un algoritmo para descargar documentos de categorías específicas del sitio web www.cubadebate.cu (ver Tabla 9). Se realizó un procesamiento de los datos, eliminando las *Stopwords* y los caracteres innecesarios como signos de puntuación. El texto fue transformado a letras minúsculas obteniendo una base documental práctica, almacenada en un archivo de extensión arff para el consumo del sistema.

Tabla 9: *Cantidad de archivos por categorías definidas. Elaboración propia.*

Categorías	**Cantidad de documentos por categorías**
Cultura	109
Deporte	109
Medio Ambiente	109
Economía	109
Política	109
Salud	109
Ciencia y tecnología	109

3.4 Estrategia de pruebas.

La estrategia clásica de pruebas de software comienza probando las partes pequeñas del software y se enfocan luego en las partes más grandes, probando finalmente el sistema como un todo (Pressman, 2010). En este caso se continuará la estrategia clásica cumpliendo con los siguientes niveles de pruebas.

- Pruebas Unitarias.
- Pruebas de Integración.
- Prueba Funcionales.

Prueba de Integración.

Las Pruebas de Integración se centran en comprobar que los módulos probados por separado funcionen en conjunto, con el objetivo de verificar que interactúan correctamente a través de sus interfaces y cubren las funcionalidades establecidas en los requisitos. Integrar operaciones una a la vez en una clase (este es el enfoque tradicional de integración) es comúnmente imposible por las relaciones directas e indirectas de los componentes que conforman una clase (Pressman, 2010). Para la realización de esta prueba se utilizó la estrategia de prueba basada en uso recomendada por Pressman. Esta estrategia comienza la construcción del sistema por probar aquellas clases (independientes) que usan muy pocas o ninguna clase. Después que esas clases son probadas, se prueba la próxima capa llamadas clases dependientes que usan a las independientes. Esta secuencia de capas de pruebas de clases dependientes continúa hasta que el sistema completo es construido.

A continuación se describen las capas de clases que se seleccionaron durante el proceso.

1. `Cleaner, SolrCommunication, Categorizer.`
2. `Principal`

1	`En Principal: instanciando la clase Categorizer.`
2	`En Principal: instanciando la clase SolrCommunication.`
3	`En Principal: instanciando la clase Cleaner.`
4	`En Principal: llamando al método processArgs de la clase Principal.`

5	En Principal: llamando al método execute de la clase Principal.
6	En Principal: llamando al método loadDataset de Categorizer.
7	En Principal: llamando al método evaluate de Categorizer.
8	En Principal: llamando al método learn de Categorizer.
9	En Principal: llamando al método saveModel de Categorizer.
10	En Principal: llamando al método loadModel de Categorizer.
11	En Principal: llamando al método getDocumentsByUrl de SolrCommunication.
12	En Principal: llamando al método cleanText de Cleaner.
13	En Principal: llamando al método classify de Categorizer.
14	En Principal: llamando al método commit de SolrCommunication.

A continuación se presentan los resultados arrojados por la prueba de integración aplicada al sistema de categorización de documentos.

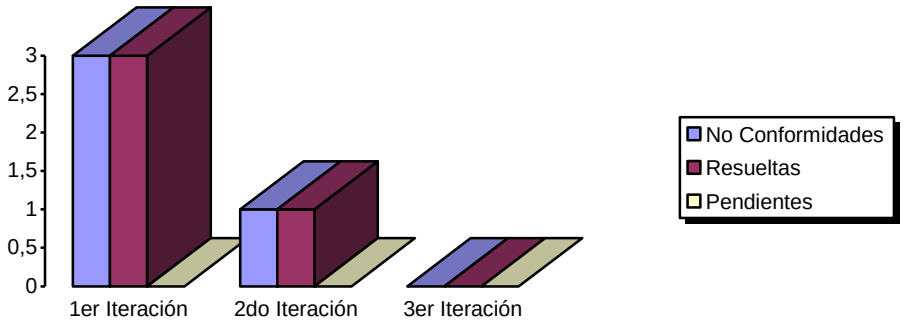


Figura 6 *Gráfico de resultados de las pruebas de integración. Elaboración propia.*

En la primera iteración se encontraron tres no conformidades relacionadas con la integración de las clases `Categorizer, Principal y SolrCommunication`, donde existía una mala referencia desde la primera y tercera clase hacia la segunda. Fue ejecutada una segunda iteración donde se encontró una no conformidad en la llamada del método `classify` de la clase Principal. Se realizó una tercera iteración donde no fueron encontradas deficiencias. En total se detectaron cuatro no conformidades que fueron corregidas.

Prueba de Unidad.

La prueba de unidad que se realiza hace uso del método de caja blanca y de la técnica del camino básico. El método del camino básico permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Pressman, 2006). Las unidades de prueba más pequeñas son las operaciones dentro de la clase. A continuación se realiza la técnica del camino mínimo al método *cleanText* de la clase *Cleaner* que permite obtener el texto procesado para la posterior categorización.

```
1    public String cleanText(String text){
2           text = text.replaceAll("[^A-Za-z\\sñáéíóú]", "").toLowerCase();
            text = text.replaceAll("^ +| +$|( )+", "$1");
            String[] words = text.split(" ");
            StringBuilder builder = new StringBuilder();
3           for (String word : words) {
4               if (!stopWordsSet.contains(word)) {
5                   if(builder.length() > 0)
6                       builder.append(" ");
7                   builder.append(word);
8                }
9           }
10          return builder.toString();
11    }
```

Figura 7 *Implementación del método* ***`cleanText`*** *de la clase* ***`Cleaner.`*** *Elaboración propia.*

Luego de enumerar el código, se diseña la gráfica del método que describe el flujo de control lógico empleando nodos y aristas.

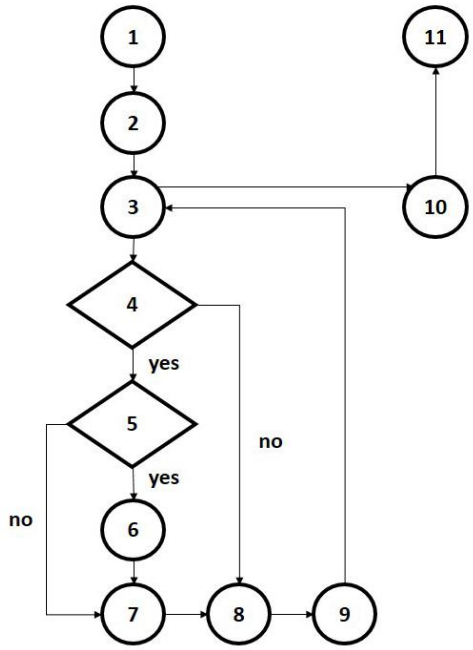


Figura 8 *Grafo de flujo. Elaboración propia.*

A partir del grafo obtenido con 11 nodos y 13 aristas se calcula la complejidad ciclomática (V(G)). La complejidad ciclomática se basa en el recuento del número de caminos lógicos individuales contenidos en un programa. Para hallar la complejidad ciclomática, el programa se representa como un grafo, y cada instrucción que contiene, un nodo del grafo. Las posibles vías de ejecución a partir de una instrucción (nodo) se representan en el grafo como aristas.

V(G) = cantidad_aristas – cantidad_nodos + 2

V(G) = 13 – 11 + 2 = 4

Una ruta independiente es cualquier ruta del programa que ingrese al menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición (Pressman, 2006). La cantidad de rutas independientes se establecen por la complejidad ciclomática, por tanto se identifican 4 rutas, tal y como se muestra en la siguiente tabla:

Tabla 10: *Listado de caminos independientes. Elaboración propia.*

No. Ruta	**Camino**

1.	1-2-3-10-11
2.	1-2-3-4-5-6-7-8-9-3-10-11
3.	1-2-3-4-5-7-8-9-3-10-11
4.	1-2-3-4-8-9-3-10-11

El valor de V(G) ofrece además un límite superior del número de pruebas que debe diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones (Pressman, 2006). A continuación se diseñan casos de pruebas para ser aplicados a cada ruta independiente.

Caso de Prueba de Unidad	
No. Ruta: 1	**Ruta:** 1-2-3-10-11
Descripción de la prueba: Obtener cadena de palabras relevantes para la categorización.	
Entrada: Se envía por parámetro un texto.	
Resultado esperado: Devolver cadena de palabras.	
Evaluación de la Prueba: Satisfactorio. Se obtiene el listado con las palabras	

Caso de Prueba de Unidad	
No. Ruta: 2	**Ruta:** 1-2-3-4-5-6-7-8-9-3-10-11
Descripción de la prueba: Obtener cadena de palabras relevantes para la categorización.	
Entrada: Se envía por parámetro un texto con palabras no relevantes.	
Resultado esperado: Eliminar palabras no relevantes en el texto	
Evaluación de la Prueba: Satisfactorio. Se obtiene un listado con las palabras relevantes	

Caso de Prueba de Unidad	
No. Ruta: 3	**Ruta:** 1-2-3-4-5-7-8-9-3-10-11
Descripción de la prueba: Obtener cadena de palabras relevantes para la categorización.	
Entrada: Se envía por parámetro un texto	
Resultado esperado: Dejar un espacio después de la última palabra de la cadena conformada	

Evaluación de la Prueba: Satisfactorio. El espacio fue generado.

Caso de Prueba de Unidad	
No. Ruta: 4	**Ruta:** 1-2-3-4-8-9-3-10-11
Descripción de la prueba: Obtener cadena de palabras relevantes para la categorización.	
Entrada: Se envía por parámetro un texto sin stopowords.	
Resultado esperado: Devolver cadena de palabras	
Evaluación de la Prueba: Satisfactorio. Se obtuvo la cadena de palabras.	

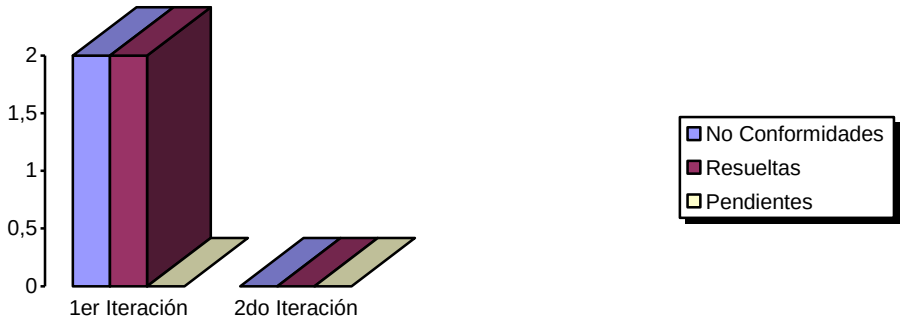


Figura 9 *Gráfico de resultados de las pruebas de unidad. Elaboración propia.*

En la primera iteración fueron detectadas dos no conformidades que fueron corregidas. Se realizó una segunda iteración donde no fueron detectadas no conformidades. Las deficiencias detectadas estaban asociadas a errores de validación y al tratamiento de las excepciones. En general fueron detectadas un total de dos no conformidades que fueron corregidas en su totalidad.

Prueba Funcional.

Las pruebas funcionales tienen como objetivo general verificar y validar un software, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo.

Para la validación del correcto funcionamiento del sistema fueron diseñados a continuación, casos de pruebas para cada categoría con el objetivo de consultar el nivel de precisión del sistema. Para esto

fueron indexados artículos de www.cubadebate.cu de los cuales se conoce sus categorías para comparar con los resultados mostrados por el sistema.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Putin y Jinping muestran preocupación por crisis en península coreana.
URL: http://www.cubadebate.cu/noticias/2017/05/14/904225/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría Política.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Modelo Socialista Cubano.
URL: http://www.cubadebate.cu/serie/modelo-socialista-cubano/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría Economía.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Poesía y poetas cubanos pasean en las Romerías de Mayo de Holguín.
URL: http://www.cubadebate.cu/noticias/2017/05/05/poesia-y-poetas-cubanos-pasean-en-las-romerias-de-mayo-de-holguin/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría Cultura.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Despaigne empata en liderazgos de jonrones e impulsadas en temporada japonesa.

URL: http://www.cubadebate.cu/noticias/2017/05/12/despaigne-empata-en-liderazgos-de-jonrones-e-impulsadas-en-temporada-japonesa/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría Deportes.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Artículos de Salud.
URL: http://www.cubadebate.cu/categoria/temas/salud-medicina/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría Salud.
Evaluación de la Prueba: Satisfactoria.

Caso de Prueba de Funcionalidad
Nombre del Artículo: La 3G llegó para quedarse
URL: http://www.cubadebate.cu/noticias/2017/04/26/la-3g-llego-para-quedarse/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría “Ciencia y Tecnología”
Evaluación de la Prueba: Satisfactorio.

Caso de Prueba de Funcionalidad
Nombre del Artículo: Expertos buscan reducir impacto del cambio climático en embalses cubanos.
URL: http://www.cubadebate.cu/noticias/2017/04/24/expertos-buscan-reducir-impacto-del-cambio-climatico-en-embalses-cubanos/
Descripción de la prueba: Obtener categoría del artículo a través del sistema.
Resultado esperado: Devolver la categoría “Medio Ambiente”.
Evaluación de la Prueba: Satisfactorio.

Los resultados arrojados por las pruebas fueron satisfactorios, se obtuvo correspondencia de las categorías arrojadas por el sistema con las categorías definidas por el sitio www.cubadebate.cu, comprobándose así las funcionalidades desarrolladas.

3.5 Validación del sistema.

El porcentaje de clasificaciones correctas no es la única métrica de evaluación utilizado para la categorización de documentos. WEKA incluye otras estadísticas de evaluación por cada clase que se utilizan con frecuencia para evaluar los sistemas de recuperación de información como los motores de búsqueda. Estos son tabulados en *Detailed Accuracy By Class* en el área de texto *Classifier output*. Se basan en el número de verdaderos positivos (VP), el número de falsos positivos (FP), el número de verdaderos negativos (VN), y el número de falsos negativos (FN) en los datos de prueba. Los positivos verdaderos son un ejemplo de prueba que está clasificando correctamente como pertenecientes a la clase de destino en cuestión, mientras que un falso positivo es un ejemplo (negativo) que está mal asignado a la clase de destino. Los falsos valores negativos (FN) y los verdaderos valores negativos (VN) se definen de manera similar. La salida de las estadísticas por WEKA se calcula de la siguiente manera:

TP Precio: VP / (VP + FN)
FP Precio: FP / (FP + VN)
Precisión: VP / (VP + FP)
Recuperación: VP / (VP + FN)
Frecuencia (F)-Medida: (2/F – 1/precisión + 1/recuperación)

En la ***Tabla 11*** también se brinda la **ROC área**, que difiere de las estadísticas de otros porque se basa en el ranking de los ejemplos de los datos de prueba de acuerdo a la probabilidad que existe de pertenecer a la clase positiva. La posibilidad está dada por la probabilidad de clase que el clasificador predice (Hall, 2011). A continuación se muestran los resultados obtenidos para la colección de entrenamiento actual del sistema a través de la herramienta Weka.

Tabla 11: *Resultados obtenidos sobre colección de entrenamiento. Elaborado por la herramienta Weka.*

VP Precio	FP Precio	Precisión	Recuperación	F-Media	ROC área	Clases
0,870	0,037	0,797	0,870	0,832	0,976	Política
0,817	0,047	0,742	0,817	0,777	0,948	Economía
0,890	0,018	0,890	0,890	0,890	0,990	Cultura
0,945	0,005	0,972	0,945	0,950	0,994	Deporte
0,807	0,023	0,854	0,807	0,830	0,977	Salud
0,727	0,040	0,755	0,727	0,741	0,950	Ciencia y tecnología
0,752	0,029	0,812	0,752	0,781	0,959	Medio ambiente
0,830	**0,028**	**0,831**	**0,830**	**0,830**	**0,970**	**Promedio**

Total de instancias:	763	
Instancias clasificadas correctamente:	633	**82,962%**
Instancias clasificadas incorrectamente:	130	17,038%
Error relativo absoluto:		21,0158%

Para validar los resultados del categorizador automático de documentos fue seleccionada una muestra de 302 documentos de distintas categorías indexados en el servidor Solr, sometiéndolos al proceso de categorización automática obteniendo los siguientes resultados.

Tabla 12: *Resultados arrojados por el categorizador. Elaboración propia.*

Clases	Muestra	Bien	%Bien	Mal	%Mal
Política	139	123	88,489	16	11,511
Economía	27	22	81,481	5	18,519
Cultura	42	36	85,714	6	14,286
Deporte	38	34	89,473	4	10,527
Salud	12	10	83,333	2	16,667
Ciencia y tecnología	35	31	88,571	4	11,429
Medio ambiente	9	8	88,888	1	11,112
Total	**302**	**264**	**87,417**	**38**	**13,583**

Conclusiones de capítulo.

El empleo de los estándares de codificación definidos para el sistema de categorización de documentos permitió desarrollar un código reutilizable, de fácil comprensión por los desarrolladores. La selección de una estrategia de prueba con un enfoque incremental permitió comprobar todas las partes del sistema y las relaciones entre estas. Mediante las pruebas realizadas se verificó que todas las instrucciones del sistema se ejecutan al menos una vez, que los componentes se integran correctamente y se validó que el sistema se ajusta de forma satisfactoria a los requisitos propuestos.

Conclusiones generales

El presente trabajo de diploma da cumplimiento a las tareas de investigación propuestas inicialmente. Se logró que los documentos indexados en el motor de búsqueda Orión fueran categorizados. De esta manera se puede concluir que:

1. Con la valoración de estado del arte de los sistemas de categorización de documentos, se garantizó una mejor comprensión del proceso, permitiendo identificar funcionalidades para dar solución al problema planteado.
2. La conformación del ambiente de desarrollo a través de un conjunto de tecnologías y herramientas identificadas en el análisis para la propuesta de solución permitió el desarrollo de un sistema estable.
3. La metodología utilizada, permitió guiar todo el proceso de desarrollo del software, generando los principales artefactos que permitieron obtener una arquitectura sólida para el sistema.
4. Con el diseño e implementación de cuatro clases permitió dar cumplimiento a los requisitos funcionales asociadas a las historias de usuarios documentadas para la solución propuesta.
5. Las pruebas realizadas facilitaron la validación de las historia de usuarios permitiendo identificar no conformidades para su posterior solución y así una mejor calidad del producto final.
6. El sistema desarrollado brinda la categoría de los documentos indexados a través del motor de búsqueda Orión, permitiendo así, consumir este servicio para mejorar el proceso de recuperación de la información del buscador.

Recomendaciones

Para futuros trabajos se tuvo en cuenta las sugerencias realizadas por los expertos por lo que se recomienda:

1. Categorizar documentos en otros idiomas haciendo uso del sistema con la utilización de **Stemmer** y **Stopwords** correspondientes.
2. Enriquecer la colección de entrenamiento del sistema de categorización de documentos con nuevas categorías con información técnica y especializada, para continuar mejorando la precisión del sistema.

Bibliografía

Alegsa, Leandro. 2010. ALEGSA. [En línea] 5 de 12 de 2010. [Citado el: 11 de 2 de 2017.] http://www.alegsa.com.ar/Dic/biblioteca.php.

Allen, R. B., Obry, P y Littleman, M. 1993. *"An interface for navigating clustered document sets returned by queries".* s.l. : Proceedings of the ACM Conference on Organizational Computing Systems, 1993.

Apache, Solr. 2015. Apache Software Fundation. [En línea] Apache, 2015. [Citado el: 2017 de 1 de 20.] http://httpd.apache.org.

Atherton, Pauline. 1968. *Manual para sistemas y servicios de información.* s.l. : UNESCO, 1968.

Bannister, T. 2012. *What is Apache?* 2012.

Berry, Michael y Castellano, Malu. 2008. *"Survey of Text Mining II ".* 2008. ISBN: 978-1-84800-045-2.

Bustamante, Dayana y Rodríguez, Jean. 2014. *Metodología Actual ¨Metodología XP¨.* UNIVERSIDAD NACIONAL EXPERIMENTAL DE LOS LLANOS. Barinas : s.n., 2014.

Carstensenl, K., Diekmann, B. y Mollera, G. 2000. *GERHARD (German Harvest Automated Retrieval and Directory).* 2000.

Castillo, Sequera, Luis, José. 2010. *NUEVA PROPUESTA EVOLUTIVA PARA EL AGRUPAMIENTO DE DOCUMENTOS EN SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN.* Alcalá : Universidad de Alcalá, 2010.

Castro, Caro, Carmen. 1998. *Sistemas de clasificación y organización de la información en Internet.* Salamanca : Facultad de Traducción y Documentación, 1998. ISBN 84-331-4609-2.

Cillero, Manuel. 2016. manuel.cillero.es. [En línea] 1 de 10 de 2016. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

Confluence. 2015. Apache Software Fundation. [En línea] 5 de 10 de 2015. [Citado el: 16 de 1 de 2017.] https://cwiki.apache.org/confluence/display/solr/Using+SolrJ.

Cutting, Karger D, Pedersen, D y Tukey, J. W. 1992. *"A cluster- based approach to browsing large document collections".* s.l. : Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992.

Dataprix. 2010. Dataprix Knowledge is The Goal. [En línea] 30 de Agosto de 2010. http://www.dataprix.com/empresa/recursos/comparativa-algoritmos-herramientas-data-mining.

Eclipse Process Framework. 2016. Eclipse. [En línea] 2016. [Citado el: 2016 de 12 de 18.] http://www.eclipse.org/epf/.

Eguiluz, Javier. 2013. *Desarrollo web ágil con Synfony2.* 2013.

Estrada, Luis Miguel. 2012. *Apache Solr - Un motor de código abierto.* México : Revista Digital Universitaria, 2012. Vol. 13. ISSN:1067-6072.

Faloutsos, Christos y Oard, D. W. 1996. *"A survey of Information Retrieval and Filtering Methods".* Maryland. : Technical Report CS- TR3514, 1996.

Figuerola, Carlos G., Zazo, Angel F. y Barrocal, Jose Luis. 2000. *Categorización automática de documentos en español: algunos resultados experimentales.* Salamanca (España) : Universidad de Salamanca, 2000.

Figuerola, Carlos, G, y otros. 2004. *"Algunas Técnicas de Clasificación Automática".* Salamanca : Universidad de Salamanca, 2004.

Flores, Ervin. 2015. *METODOLOGIAS AGILES PROCESO UNIFICADO AGIL (AUP).* La Paz : s.n., 2015.

Frakes, W. B. 1992. *Stemming algorithms, Information retrieval: data structures and algorithms.* 1992.

Gamma, Erich, y otros. 2010. *Elements of Reusable Object-Oriented Software.* 2010.

Gómez, Felipe, Laureano. 2009. *Memorias de prácticas en el uso del indizador Swish-e,.* s.l. : Sistema de indizacion y recuperación de la información digital, 2009.

Google. 2016. Google. [En línea] 2016. [Citado el: 2016 de 11 de 18.] https://www.google.com/intl/es/insidesearch/howsearchworks/algorithms.html.

Hall, Mark. 2011. *Practical Data Mining. Introduction to the WEKA Explore.* s.l. : University of Waikato, 2011.

Intel. 2017. [En línea] Intel, 2017. [Citado el: 16 de 2 de 2017.] http://www.intel.com.

Izaurralde, Maria Paula. 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias.* Universidad Tecnológica Nacional : s.n., 2013.

Jaramillo, Angélica y Paz, Henry. 2015. *Aplicación de Técnicas de Minería de Datos para Determinar las Interacciones de los Estudiantes en un Entorno Virtual de Aprendizaje.* s.l. : Revista Tecnológica ESPOL, 2015.

Java, Team. 2009. Conozca más sobre la tecnología Java. [En línea] 2009. [Citado el: 4 de 12 de 2016.] http://www.java.com/es/about/..

KNIME. 2017. KANIME. [En línea] 2017. http://kanime.org/.

Larman, Craig. 2003. *UML y Patrones. 2da Edición.* s.l. : Prentice Hall, 2003.

Leyva, Rodríguez,Paúl. 2016. *Componente y funcionalidades de un sistema de recuperación de la información.* La Habana : Revista Cubana de Ciencias Informáticas, 2016. ISSN: 2227.

Maarek, Y. S, y otros. 2000. *"Ephemeral Document Clustering for Web Applications".* s.l. : IBM Research Report RJ 10186., 2000.

Macskassy, S. A, y otros. 2001. *"Human performance on clustering web pages".* New Jersey : [Macskassy et al, 2001] Macskassy, S. A, Banerjee, A, Davison, B. D, Hirsh, H. 2001. "Human performance on clusterinDepartment of computer Science , 2001. [Macskassy et al, 2001] Macskassy, S. A, Banerjee, A, Davison, B. D, Hirsh, H. 2001. "Human performance on clusterinDCS- TR- 355.

Messerschmitt, David C y Szyperski, Clemens. 2003. *Software Ecosystem: Understanding an Indispensable Technology and Industry.* Cambridge : MIT Press, 2003. SBN 0-262-13432-2.

Microsoft. 2016. onlinehelp. [En línea] Microsoft, 2016. [Citado el: 2016 de 12 de 1.] http://onlinehelp.microsoft.com/es-es/bing/ff808447.aspx.

NetBeans. 2016. NetBeans. [En línea] 2016. [Citado el: 18 de 11 de 2016.] https://netbeans.org/index_es.html.

OCLC. 2015. [En línea] 2015. http://www.oclc.org/research/activities/scorpion.html.

Olson, D. 2008. *"Advanced Data Mining Techniques".* 2008. ISBN:978-3-540-76916-3.

OMG-UML. 2007. *Superstructure.* 2007. V2.1.2.

Pinto, Maria. 2012. *mariapinto.* [En línea] 6 de 18 de 2012. [Citado el: 15 de 10 de 2016.] http://www.mariapinto.es/e-coms/busqueda-y-recuperacion-de-informacion/.

Pressman, Roger S. 2006. 5ta Edición. [En línea] 2006. [Citado el: 2017 de 1 de 12.] http://bibliodoc.uci.cu/pdf/reg02689.pdf.

Pressman, Roger, S. 2010. Software Engineering. A Practitioner's Approach. *Software Engineering. A Practitioner's Approach.* Boston : Mc Graw Hill, 2010.

Raghavan, V, Bollmann, P y Jung, G. 1989. *"A critical investigation of recall and precision as measures of retrieval system performance".* s.l. : ACM Transactions on Information Systems, 1989. 7(3: 205- 229.

Rapidminer. 2016. RapidMiner: Data Science Platform. [En línea] 2016. [Citado el: 2016 de 12 de 2.] https://rapidminer.com.

Rochio, J.J. 1971. *"Relevance feedback in information retrieva".* Englewoods Cliff : The SMART , 1971.

Rüger, S.M.R. y Gauch, S.E. 2000. *Feature reduction for document clustering and classification.* London : Imperial Collage, 2000.

Schütze, Hinrich y Silverstein, Craig. 1997. *"Projections for Efficient Document Clustering".* s.l. : in Proceedings of ACM/ SIGIR' 97, 1997.

Search Technologies. 2016. [En línea] 2016. [Citado el: 2017 de 4 de 13.] https://www.searchtechnologies.com/solr-categorization.

Suzuki, J. 1995. *"Analysis on simple genetic algorithms".* s.l. : IEEE Transactions on Systems, Man, and Cybernetics, 1995.

Tarragó, Sánchez, Dánel. 2014. *Algoritmos para la Clasificación Multinstancia.* Granada : Universidad de Granada, 2014.

The Lemur Project. 2017. Lemur Project Home. [En línea] 20 de 1 de 2017. https://www.lemurproject.org/.

Universidad de las Ciencias Informáticas. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Vallejos, Sofia. 2006. *Minería de Datos.* Argentina : s.n., 2006.

Van Rijsbergen, C.J. 1979. *"Information Retrieval".* London : University of Glasgow, 1979.

Visual Paradigm. 2015. [En línea] 16 de 4 de 2015. [Citado el: 2016 de 11 de 18.] http://www.visual-paradigm.com.com/support/faq.jsp.

Weka. 2016. weka. [En línea] 2016. [Citado el: 2016 de 11 de 18.] http://se-weka.blogspot.com/.

Wilson, Leslie B. 1993. *Comparative Programming Languages, Second Edition.* 1993. ISBN 0-201-56885-3.

Yates, Baeza, Ricardo. 2010. *Coleccionables de Aplicaciones Informáticas. Edición 5 Motores de búsquedas.* Buenos Aires : s.n., 2010.

Zamir, Oren y Etzioni, Oren. 1999. *"Grouper: A Dynamic Clustering Interface to Web Search Results".* Computer Networks and ISDN Systems. : Proceedings of the Eighth International World Wide Web Conference, 1999.