Universidad de las Ciencias Informáticas

Facultad 10

Identificador Automatizado de Idiomas para Textos

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

Autor(es): Yurisleidy Hernández Moya

Dionny Cardoso Carmona

Tutor(es): Lic. Isachi Abreu Gil

Lic. Diosmides García Valladares

Ciudad de la Habana, junio 2008, Cuba

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste, firmamos la presente a los ___ días del mes de _____ del año _____ .

Yurisleidy Hernández Moya

_____

Firma del Autor

Dionny Cardoso Carmona

_____

Firma del Autor

Isachi Abreu Gil

_____

Firma del Autor

Diomides García Valladares

_____

Firma del Autor

Agradecimientos

A mis padres, Olga y Emilio, por todo su apoyo y confianza; sin ellos hubiese sido imposible cumplir este sueño.

A mi hermana, sarita, la niña más linda y esbelta, por hacerme sentir el compromiso de intentar cada día ser un buen ejemplo.

A mis abuelos, a toda la familia, por su cariño.

A las maravillosas "pri", en especial a yenlyta, leyany, irita, por ser conmigo las personitas más lindas de la UCI; por su amistad y ayuda incondicional.

A mi compañero de la vida, Dovier A. Ripoll, por su paciencia y amor, por ser para mí: el ejemplo de profesional a seguir.

Al neno, Dionny Cardoso, mi amigo y confidente, mi hermano, por soportarme y estar siempre para mí.

A todos los que, aunque no menciono, han constribuido a mi formación personal y profesional en el transcurso de estos cinco largos años y han hecho de la UCI un lugar inolvidable.

A los amigos que saben que los quiero, Gracias.

Yurisleidy Hernández Moya

A mis padres Iliana y Jose sin ellos no hubiera sido posible.

A mis entrañables amigos de la UCI, los que no se irán nunca, el Micho, el Yoni, Arielazo, el Yosho, Edwin, Orliandi, Los Cesar, David, Islandi, Hugo, Juanca, Mayte, el Coco, Rigo, Luisito el camarada, a Katia mi flaca y todos los que no menciono y saben que están. A Ismael por su colaboración y ayuda.

Por sobre todo a mi amiga y compañera de tesis, por confiar en mi, por darme la oportunidad de conocerla, por ser incondicional y aguantarme tanto, a Yuri.

Y a todos los que de una forma u otra han contribuido con la realización de este trabajo.

Dionny Cardoso Carmona

Dedicatoria

A mi mamá por ser mi estrella.

A mi papá y mi hermana por creer en mí, apoyarme y quererme tanto.

Yurisleidy Hernández Moya

A mi mamá por todo lo que soy, por lo que supo crear, por todo su cariño y amor, por ser la mejor madre del mundo. A mi papá José Manuel por llegar a mi vida para no irse nunca, gracias por enseñarme a construir y crear, por ser mi ejemplo a seguir. A mis hermanitos Andy y Sandy, esos locos bajitos, gracias por rescatar en mí la inocencia. A mis abuelitos Juanita y Amado, por ser más para mi que para ellos mismos, por ser los viejitos más lindos del mundo. A mis abuelas Cita y Placida. A Tony y Delia por siempre preocuparse. A mis amigos de siempre, a Manuel, Yoander, Edwin, el Chino, Katia. A Yuri que me vuelve loco y regaña como nadie, por mirarme por dentro y creer en mí, por ser mi hermana.

Dionny Cardoso Carmona

RESUMEN

En este trabajo se presenta la implementación de un Identificador Automatizado de Idiomas para Textos (IDENAIT). Se empleó la metodología ágil Programación Extrema, para guiar el proceso de desarrollo del sistema; además se usó Debian GNU/Linux como sistema operativo y lenguaje de programación Perl en su confección.

IDENAIT fue diseñado con el objetivo de suplir las necesidades de identificar el idioma del texto de un documento HTML, como uno de los pasos necesarios para el funcionamiento de módulos internos del Motor de Categorización Automatizado de Documentos HTML (MCADHTML); el cual podría ser, una vez desarrollado, una herramienta útil para incorporar al producto Filtrado de Paquetes por contenido (Filpacon). Permite la modificación de varios de sus parámetros como el número de idiomas reconocidos, entre otros, mediante un Fichero de Configuración. El sistema transcurre por dos fases: entrenar e identificar. Durante el entrenamiento crea los modelos de idiomas; en la identificación crea el modelo para cada documento de entrada y los compara con los modelos de los idiomas, asignando los documentos al idioma más parecido.

Para conformar los modelos de los idiomas y los documentos a identificar se hizo uso además: del Modelo de Espacio Vectorial, la técnica de n-gramas de palabras y la función Frecuencia de Aparición para el cálculo del peso de los términos del vector. Para la comparación entre representaciones de dos documentos se utiliza el algoritmo out-of-place.

Concluida la implementación del IDENAIT, se cuenta con una herramienta capaz de automatizar el proceso de identificar el idioma del texto para el MCADHTML y que posibilita la identificación de 39 idiomas.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

ACR Reconocimiento Artificial de Contenido, (*Artificial Content Recognition*)

API Análisis y Procesamiento de Información

CENTERNET CENTro de Estudios de InteRNET

CPAN Comprehensive Perl Archive Network

Filpacon Filtrado de paquetes por contenidos

GPL Licencia Pública General, (*General Public License*)

HTML Lenguaje de Marcado de Hipertexto, (HyperText Markup Language)

IDE Entorno Integrado de Desarrollo, (*Integrated Development Environment*)

IDENAIT IDENtificador Automatizado de Idiomas para Textos

MCADHTML Motor de Categorización Automatizada de Documentos HTML

MEV Modelo de Espacio Vectorial, (*Vector Space Model*)

Perl Lenguaje Práctico para la Extracción e Informe, (*Practical Extraction and Report Language*)

PLN Procesamiento de Lenguaje Natural, (*Natural Language Processing*)

SO Sistemas Operativo, (*Operating system*)

TF Frecuencia del Término, (*Term Frequency*)

TF-IDF Frecuencia del Término-Frecuencia Inversa del Documento, (*Text Frequency-Inverse Document Frequency*)

UCI Universidad de las Ciencias Informáticas

URL Localizador Uniforme de Recurso, (*Uniform Resource Locator*)

XP Programación Extrema, (*Extreme Programming*)

YAML YAML no es otro lenguaje de marcado, (*YAML Ain't Another Markup Language*)

INTRODUCCIÓN

La Era Digital brinda a los usuarios ilimitadas posibilidades de generar y publicar información en Internet, provocando un crecimiento exponencial (BERROCAL, 2004) de los contenidos en un plazo sorprendentemente corto. Un gran número de documentos educativos, científicos y didácticos son generados diariamente, aportando beneficios notables al conocimiento humano. Por ser Internet una inmensa red mundial, compuesta por varias redes, generalmente no es controlada por compañía o gobierno alguno; la posibilidad de publicar información en este medio trae consigo, además, la aparición de contenidos inadecuados (nocivos[1] e ilícitos[2]) (GARCÍA, 2006), tales como la pornografía infantil y entre adultos, la violencia, el terrorismo, entre otros. Por tanto, el problema ya no radica en la falta de información sino en cómo es posible controlar el acceso a la misma. Esta situación constituyó el punto de partida para el desarrollo, en la *Universidad de las Ciencias Informáticas[3] (UCI)*, de un producto denominado *Filtrado de paquetes por contenidos (Filpacon)*: solución informática que permite regular (permitir o denegar) el acceso de usuarios a contenidos de Internet. Para ello, tal filtro utiliza una base de datos con direcciones (URL) previamente categorizadas. Solución que presenta inconvenientes si se considera el carácter altamente dinámico de la Web. Dada esta situación el grupo *de Análisis y Procesamiento de Información (API)*, perteneciente al *Polo Productivo Centro de Estudios de Internet (CENTERNET)*, se propone desarrollar un *Motor de Categorización Automatizada de Documentos HTML (MCADHTML)* que, con técnicas de Inteligencia Artificial (IA), permita categorizar el contenido de páginas Web teniendo en cuenta, principalmente, los elementos primarios: textos, imágenes e idiomas; de esta manera sería una herramienta útil para crear y actualizar, automatizadamente, la información de la Base de Datos de URLs Categorizadas de *Filpacon*.

MCADHTML estará compuesto, entre otros módulos, de un *Categorizador Automatizado de Texto,* que asignará a cada documento HTML una o varias categorías definidas, analizando únicamente el contenido textual de tales documentos. Tal módulo será desarrollado para categorizar documentos en determinados idiomas. Por tanto, se hace imprescindible conocer el idioma del texto antes de ser procesado por el *Categorizador Automatizado de Texto*. Teniendo en cuenta lo explicado

1 contenidos nocivos: a pesar de ser legales son perjudiciales para los menores, afectando su desarrollo físico y mental; su acceso está autorizado para los mayores.

2 contenidos ilíticos: prohibidos para todas las personas independientemente de sus edades, ejemplos son la incitación a la violencia, actividades terroristas y la producción y tráfico de drogas.

3 http://www.uci.cu/

anteriormente, existe el siguiente **problema a resolver:** ¿Cómo automatizar el proceso de identificar el idioma de textos para *MCADHTML*?

En la investigación realizada se determinó como **objeto de estudio:** la Identificación Automatizada de Idiomas en Textos. Se decidió además, que el trabajo estará enfocado a la Identificación Automatizada de Idiomas para el MCADHTML como **campo de acción**.

Como **idea a defender** se establece la siguiente: el proceso de identificar el idioma de textos puede ser automatizado para el *MCADHTML*.

El **objetivo general** que se plantea, como base del trabajo a efectuar, es: desarrollar un Identificador Automatizado de Idiomas para Textos (IDENAIT), para el *MCADHTML*. Tal objetivo general se divide en los siguientes **objetivos específicos**:

- Indagar sobre Estado del Arte de la identificación automatizada de idiomas para conocer sobre las técnicas y tendencias más aplicadas en este tema; seleccionar la metodología de desarrollo de software a usar.
- Describir la propuesta del IDENAIT.
- Analizar las características del sistema y diseñar la estructura a seguir para el proceso de desarrollo.
- Implementar el IDENAIT y probar las funcionalidades adquiridas.

Con la idea de trazar una guía para darle cumplimiento a los objetivos antes mencionados, se proponen las siguientes **tareas de Investigación**:

- Investigar sobre los siguientes temas (*Fundamentación Teórica*):
  - Aplicaciones del Procesamiento del Lenguaje Natural.
  - Categorización Automatizada de Textos.
  - Uso del Modelo de Espacio Vectorial para la representación de los documentos.
  - Uso de palabras y n-gramas.
  - Funciones para el cálculo de pesado de términos.

- Cálculo de cercanía entre vectores.
- La identificación automatizada de idiomas para textos.
- Herramientas útiles en del desarrollo de identificadores automatizados de idiomas.
- Trabajos similares, realizados en el área de la identificación automatizada de idioma.
- Seleccionar la metodología para guiar el desarrollo del sistema.

- Para proponer las características del sistema (*Características del Sistema*), abordar sobre:
  - El objetivo Estratégico de la Organización.
  - El objeto de automatización.
  - La información que se maneja.
  - Las características de la propuesta del sistema.
- Para desarrollar las fases de Exploración, Planificación de la Entrega e Iteraciones del sistema (*Exploración, Planificación de la Entrega e Iteraciones*) especificar:
  - Las características de la fase de Exploración.
  - Las Historias de Usuarios.
  - Las características de la Fase de Planificación
  - La estimación de esfuerzos por Historia de Usuarios.
  - Las iteraciones.
  - El plan de duración de las iteraciones.
  - El Plan de entrega.
- Para realizar implementación y pruebas del sistema (*Implementación y Prueba*) generar las:
  - Tareas de ingeniería por iteraciones.
  - Pruebas de aceptación.

Además, con el objetivo de alcanzar una mayor organización y profundidad en los temas que serán tratados, deben usarse los siguientes **métodos científicos de investigación**:

Métodos teóricos.

Analítico-Sintético: permite el procesamiento de la bibliografía utilizada; posibilitando obtener los elementos necesarios para aplicar al desarrollo de la investigación.

Histórico-Lógico: para estudiar la evolución y desarrollo del objeto de estudio y sus tendencias actuales.

Métodos empíricos.

Observación: para analizar los resultados durante el proceso de desarrollo de IDENAIT.

Al concluir el presente trabajo, se esperan obtener los siguientes resultados:

- Base teórica para construir identificadores de idiomas para textos.
- Tareas de implementación y pruebas para obtener un producto funcional y acorde con las necesidades del cliente.
- Colección de Entrenamiento, para cada idioma que se identificará, pre-categorizada manualmente.
- La automatización del proceso de identificación del Idioma en textos, contribuyendo al desarrollo del MCADHTML.

El presente documento está compuesto por cuatro capítulos, los cuáles se estructuran del siguiente modo:

En el Capítulo Uno: **Fundamentación Teórica,** se abordará del Procesamiento del Lenguaje Natural algunas aplicaciones y técnicas como el Modelo de Espacio Vectorial; el uso de funciones de ponderación, n-gramas y palabras para la representación vectorial de los documentos. Se continuará con aspectos de la identificación automatizada de Idiomas: esquema general, el uso de "stopwords", estudios realizados y programas en línea. Se presentarán las herramientas y metodología a utilizar para el desarrollo del sistema propuesto. Finalmente se realizará un estudio sobre trabajos afines.

En el Capítulo Dos: **Características del Sistema**, se expondrán los objetivos estratégicos de la organización, el objeto de automatización y la información que se manejará. Concluirá el capítulo con las características de la propuesta del sistema a desarrollar.

En el Capítulo Tres: **Exploración, Planificación de la Entrega e Iteraciones**, se detallarán los artefactos generados durante las fases de exploración y planificación del proyecto. Se generarán las Historias de Usuarios. Seguidamente se definirá el alcance de cada iteración y se estimará el tiempo necesario para desarrollar el producto. Finalmente se elaborará un plan de entrega.

En el Capítulo Cuatro: **Implementación y Prueba**, se definirán las tareas de ingeniería por cada iteración y se realizan las pruebas de aceptación sobre el sistema.

1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El avance de la tecnología ha provocado que el acceso a Internet se haya extendido a casi todas las regiones del mundo, causando gran impacto en varios aspectos de la vida: trabajo, estudio y ocio, entre otros. El número de internautas[4] aumenta notablemente cada año, los cuales usan múltiples servicios: Correo Electrónico, World Wide Web, FTP y Grupos de Noticias, entre otros. Dicho espacio virtual proporciona una cantidad significativa de información generada diariamente, dando lugar a uno de los medios de más alto crecimiento del mundo. La expansión global que está alcanzando Internet y la facilidad de publicación en ella, propicia encontrar documentos electrónicos en varios idiomas y dialectos. Investigaciones recientes[5] han demostrado que los idiomas más usados en la Web son (ordenados descendentemente): Inglés (en), Chino (sh), Español (es), Japonés (ja), Francés (fr), Alemán (de), Portugués (pt), Árabe (ar), Coreano (ko) e Italiano (it).

Realizar estudios, desde un enfoque manual, de una parte significativa de los contenidos disponibles en Internet resultaría prácticamente imposible debido, fundamentalmente, a los grandes volúmenes, variedad y constante modificación de su información. Por tanto, para tales estudios, es necesario el uso de programas informáticos (como de identificación de idiomas y categorización de textos, entre otros) que procesen la información usando técnicas de Procesamiento del Lenguaje Natural (PLN).

En este capítulo se tratan algunas aplicaciones y técnicas del PLN, tales como la Categorización Automatizada de Texto y el Modelo de Espacio Vectorial respectivamente. Se revisa brevemente la necesidad de funciones de ponderación, n-gramas y palabras para la representación vectorial de los documentos. Posteriormente se expone el uso de la Función del Coseno para el cálculo de similitud entre vectores. Se tratan algunos aspectos de la identificación automatizada de idiomas para textos: esquema general, posible uso de palabras vacías (stopwords), algunas investigaciones realizadas en la UCI y en el ámbito

4 Usuarios que navegan en Internet.

5 http://www.internetworldstats.com/stats7.htm

internacional. Se mencionan las herramientas propuestas a utilizar para el desarrollo del IDENAIT. Finalmente se concluye con la selección de la metodología de software a utilizar y trabajos afines al efectuado en esta tesis.

1.2. Procesamiento de Lenguaje Natural

El PLN es una subdisciplina de la IA y la rama ingenieril de la lingüística computacional, surgida en la década de los 60. Conjunto de técnicas algorítmicas para la creación de programas informáticos que posibiliten el tratamiento de la información en lenguaje natural[6], tanto en su manifestación escrita como oral. A pesar del buen desempeño en varias tareas, presenta dificultades en el tratamiento de propiedades del lenguaje natural como la variación lingüística (ocurrencia de diferentes formas alternativas, para expresar un mismo significado, en el dominio de una lengua) y la ambigüedad lingüística (cuando una palabra o frase permite más de una interpretación) (VALLEZ y PEDRAZA-JIMENEZ, 2007).

Además, abarca un amplio espectro de investigación y trabajo (GIL y RODRÍGUEZ, 1996); entre sus principales aplicaciones se encuentran:

- Agrupamiento de Textos: agrupar documentos con características similares.
- Categorización de Textos: asignar documentos a una o varias categorías conocidas con anterioridad.
- Construcción Automática de Resúmenes: obtener el resumen de un texto.
- Reconocimiento del Habla: identificar o procesar el lenguaje natural oral.
- Traducción Automática: traducir correctamente de un lenguaje a otro.
- Respuesta a Preguntas: obtener la respuesta a una pregunta concreta.
- Extracción de Información: encontrar la información relevante en un conjunto de textos, ignorando la no significativa.

Por su parte, la Identificación de Idiomas es un proceso que requiere conocimiento de sintaxis, semántica, ortografía y gramática; es decir, necesita de técnicas del PLN. En esta tesis se abordarán aquellas técnicas del PLN enfocadas al tratamiento de la información escrita o textual.

6 Es el lenguaje hablado y/o escrito y/o usado por humanos para propósitos generales de comunicación.

1.3. Categorización Automatizada de Textos

La Categorización Automatizada de Textos es la tarea de asignar a un documento una o varias categorías temáticas (ver la **Figura 1**), partiendo de un conjunto de textos categorizados previamente y separados en clases o categorías temáticas (SEBASTIANI, 2002); para cada una de las cuales se determinan sus características representativas y se conforma su modelo, de manera que sea posible establecer futuras comparaciones. Para la categorización de un documento es necesario, primeramente, obtener su modelo y compararlo con el modelo de cada categoría. Un documento pertenecerá a la clase con la cuál posea más características en común. Generalmente, para materializar la tarea de categorización es necesario: (1) representar los documentos y (2) comparar dos representaciones de documentos. El proceso de obtener el modelo de cada categoría se conoce como *entrenamiento o aprendizaje* y el conjunto de documentos de referencia como *Colección de Entrenamiento* (FIGUEROLA, 2005).

Figura 1: Categorización de Textos.

(GAYO, 2005) planteó: *"Identificar el idioma en que está escrito un texto constituye un caso particular dentro de la categorización de documentos*". Lo anterior se debe a que la identificación de idiomas puede materializarse como un caso típico de la Categorización Automatizada de Textos, tomando como categorías los idiomas.

1.4. Modelo de Espacio Vectorial

En varias aplicaciones que realizan procesamiento de documentos es necesario obtener una representación de estos. Una de las técnicas más empleadas, para lograr tal representación, ha sido el Modelo de Espacio Vectorial (MEV) (SALTON, WONG y YANG, 1975). Dicho modelo es algebraico y ampliamente utilizado en sistemas de Recuperación de Información (en inglés, *Information Retrieval*). Permite la representación abstracta de documentos, escritos en lenguaje natural, mediante el uso de vectores de términos en un espacio *T*-dimensional, donde *T* es la cantidad de términos o rasgos distintos de la Colección de Entrenamiento. Cada término debe tener asociado una importancia o peso, que expresa su relevancia en el documento y se calcula por medio de una Función de Ponderación. De este modo, el término *wt* podrá ser representado en el documento *Di*; *wit* es el peso del término t-ésimo para el documento *Di* (ver la **Figura 2**).

$$Di = (wi1, wi2, wi3, ..., wit)$$

Figura 2: Documento en un espacio vectorial de T dimensiones

Este modelo asume que las cadenas de un mismo texto no tienen relación entre ellas, obviando además el orden en que aparecen.

Para el desarrollo de *IDENAIT*, la representación vectorial de documentos estará conformada únicamente por los términos que aparecen en el propio documento.

1.5. Palabras y N-Gramas

En tareas donde se realice procesamiento de documentos (como en la Identificación Automatizada de Idiomas) y se requiera representarlos como vectores, será necesario obtener las características más importantes del contenido de cada documento para conformar su vector representativo. Tales características pueden ser, entre otras, palabras y n-gramas.

El uso de palabras en tales tareas puede resultar ineficaz cuando el texto de los documentos contiene errores de tipo ortográficos, entre otros. Palabras con equivocaciones en su composición, como una simple omisión de acento,

imposibilitarán ser agrupadas bajo una única forma con otras que, aunque similares, no posean la misma deficiencia. Esta situación dará lugar entonces a palabras distintas y por tanto a coordenadas diferentes del vector, provocando que se modifiquen los resultados esperados.

Por su parte, un n-grama es una secuencia de *n* elementos (palabras o caracteres) de un texto dado y pueden estar no necesariamente continuos. No obstante, durante este trabajo se hará referencia únicamente a n-grama como secuencia de *n* caracteres contiguos de una palabra; por ejemplo, de la palabra *TEXT* los n-gramas de tamaño 1 hasta 3 son mostrados en la **Figura 3**.

uni-grama: T, E, X, T
bi-grama: _T, TE, EX, XT, T_
tri-grama: _TE, TEX, EXT, XT_

Figura 3: N-grama de tamaño desde 1 hasta 3 de la palabra TEXT

donde el símbolo "_ " representa el espacio en blanco que delimita las palabras.

La consideración de n-gramas como términos, para la representación de documentos, es comúnmente más utilizada que las palabras, puesto que mejora notablemente el comportamiento ante la presencia de documentos con ruido[7]. Palabras con errores al ser descompuestas en n-gramas, darán lugar a un mayor número de términos en común con las que se encuentran en su forma correcta (GAYO, 2005); ver **Figura 4**.

construir=> _con cons onst nstr stru trui ruir uir_
*contruir => _con cont ontr ntru trui ruir uir_

referencia =>_ref refe efer fere eren renc enci ncia cia_
*referecia =>_ref refe efer fere erec reci ecia cia_

Figura 4: 4-gramas obtenidos para una serie de palabras

7 Documentos que pueden tener cualquier tipo de errores, ejemplo: ortográficos

La figura anterior (Figura 4) muestra las palabras desglosadas en 4-gramas y precedido con asterisco las que poseen errores ortográficos. Marcados en estilo de fuente **negrita** los elementos en común entre aquellas escritas de manera correcta e incorrecta. Los n-gramas comunes de ambas palabras aportarán entonces mayor peso a la representación.

El uso de n-gramas para la identificación de idioma tiene la desventaja de que un mismo n-grama puede pertenecer a varios idiomas, aunque por lo general los más comunes de un idioma no los son de otro.

En el desarrollo del IDENAIT se utilizará la técnica de n-gramas, de tamaño desde 1 hasta 5 caracteres, como términos del vector para representar documentos.

1.6. Funciones de Ponderación

Para el cálculo del peso de los términos, en el contenido de un documento, existen varias alternativas conocidas como Funciones de Ponderación. Estas pueden distinguirse en funciones locales y globales (FRESNO, 2006):

Aquellas que solo tienen en cuenta la información del propio documento para generar su representación son las de ámbito "local"; entre ellas se mencionan:

Función Binaria (Binary, Bin), es la vía más simple de calcular el peso de un término, considerando que todos los rasgos de un documento son igualmente importantes. De esta manera, los pesos de los términos son binarios (**wij Є {0,1}**), cero o uno si el término aparece o no en el documento.

wij = 0 si el término j-ésimo no ocurre en el Di.
wij = 1 si el término j-ésimo ocurre en el Di. No importa cuántas veces ocurre.

A pesar de la simplicidad computacional de esta función, en la representación vectorial de documentos considerar todos los términos igualmente relevantes no es eficiente. En ocasiones un rasgo del texto puede tener más peso que otros, es decir, tener más importancia que otro dentro del propio documento.

Frecuencia de Aparición (Term Frequency TF), es una de las representaciones más sencillas. Los términos que más se repiten dentro de un documento, constituyen elementos más importantes que los menos usados. Por tanto, en el vector que representa al documento, el peso de cada término estará dado por la cantidad de veces que éste aparece en el texto; puede verse en la siguiente ecuación:

F : T F (ti , dj) = wij , frecuencia del rasgo ti en dj

Las funciones de ponderación "global" tienen en cuenta la colección para generar la representación del documento, es decir, para el cálculo de relevancia de los términos tienen en cuenta la información de los textos restantes de la colección. A continuación se menciona una de ellas:

Frecuencia del Término x Frecuencia Inversa del Documento (Term Frecuency - Inverse Document Frecuency). En el vector que representa al documento, el peso de cada término estará dado por la siguiente ecuación:

F: TF x IDF(ti, dj) = wij x log (N/df(ti))

wij: frecuencia del término ti en dj.
N: número de documentos de la colección.
df(ti): cantidad de documentos de la colección que tiene el término ti.
El peso de un término en un documento aumenta a medida que es más frecuente dentro de este documento y disminuye a medida que aparece en una mayor cantidad de documentos de la colección.

En el desarrollo del IDENAIT se utilizará la Función Frecuencia de Aparición para determinar la relevancia del término en el contenido de un documento.

1.7. Cálculo de Cercanía entre Vectores

Como se ha visto en secciones anteriores, es posible la representación de un texto como un vector de términos que tienen asociado un peso que establece su relevancia dentro del contenido del documento. Por tanto, es posible calcular alguna medida de asociación entre dos documentos, la cual podría estar relacionada con su similitud. La semejanza entre dos documentos se resume en calcular la proximidad entre vectores representativos de estos documentos. Una de las funciones utilizada con este fin, es conocida como Función del Coseno (GAYO, 2005). Partiendo de la representación vectorial de dos documentos: Q y D; (ver **Figura 5**).

$$Q = (q1, q2, q3, ..., qn) \text{ y } D = (d1, d2, d3, ..., dn)$$

Figura 5: Dos documento en un espacio vectorial de N dimensiones

Donde qi y di representan los pesos del término i-ésimo de los vectores Q y D respectivamente y además N es el número de términos distintos de la colección, entonces la Función del Coseno quedaría representada como muestra la **Figura 6.**

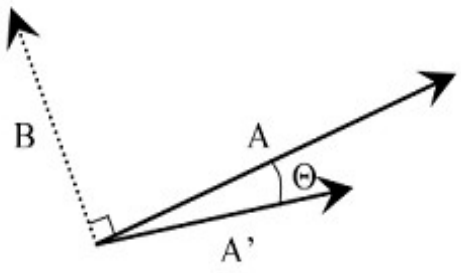
$$\frac{\sum_{i=1}^{n} q_i \cdot d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \cdot \sqrt{\sum_{i=1}^{n} d_i^2}}$$

Figura 6: Función del Coseno entre los Documentos Q y D.

Tal función, es equivalente a calcular el producto escalar de dos vectores de documentos (Q y D) y dividirlo por la raíz cuadrada de la sumatoria de los componentes del vector Q multiplicada por la raíz cuadrada de la sumatoria de los componentes del vector D.

El "producto escalar de dos vectores", es obtenido multiplicando componente a componente, de los vectores, y sumando los productos.

El valor obtenido de esta función estará comprendido entre cero y uno. Una vista geométrica de este método es mostrado en la **Figura 7,** donde los vectores **A** y **A`** son más similares pues el ángulo entre ellos es próximo a 0° ($\omega \approx 0°$) y por tanto su valor tiende a 1. Los vectores **A** y **B**, el ángulo que los conforma es 90° ($\omega \approx 90°$) y por tanto su similitud es cero. Lo anterior indica que dos vectores son más parecidos cuando su similitud, calculada por la Función del Coseno, es más próxima a uno.



$d(A, A') \cong 1 \rightarrow \Theta \cong 0^o$
$d(A, B) = 0 \rightarrow \Theta = 90^o$

Figura 7: Representación Geométrica de la Función del Coseno.

Por otra parte y suponiendo que los vectores que representan a cada documento tendrán únicamente los términos que aparecen en el contenido del mismo, ordenados descendentemente por su frecuencia de aparición. Cavnar y Trenkle, proponen una medida conocida como “fuera-de-lugar” (out-of-place) (CAVNAR y TRENKLE, 1994) para calcular la distancia entre vectores de documentos. Se basa principalmente en calcular cuán lejos están dos documentos. (ver la **Figura 8**)

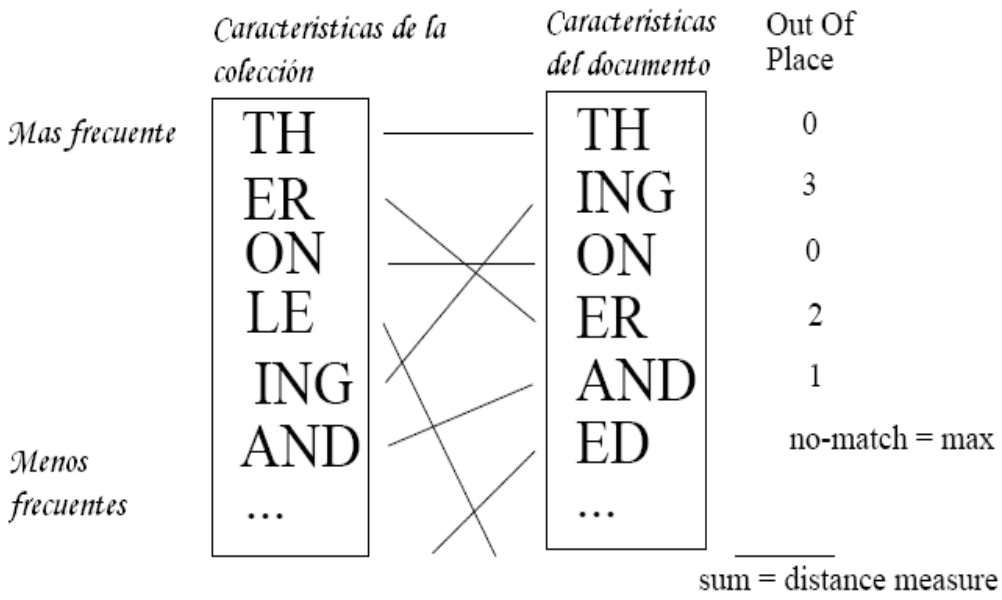


Figura 8: Representación Gráfica para el cálculo de OUT-OF-PLACE.

En la **Figura 8** se muestran dos vectores que representan la categoría y el documento que se desea clasificar. El cálculo se realiza de manera sencilla, sumando el valor modular de la resta de las posiciones de los términos comunes en ambos vectores; en caso que un término no se encuentre en el vector de la categoría se suma un número muy grande. Por ejemplo, en la figura el término TH en ambos vectores está en la posición 1 por tanto la resta de sus índices es cero, ING tiene índice 5 y 2 por tanto su resta es 3, el término ER ocupa la posición 2 en un vector y 4 en el otro dando resta 2, en el caso del término ED no se encuentra en el vector de la categoría y por tanto es sumando un número muy grande.

De esta manera, el documento será más próximo a la categoría con la cuál su medida "out-of-place" sea menor, es decir, posea mayores términos en común.

1.8. Identificación Automatizada de Idiomas para Textos

La identificación automatizada del idioma en textos puede describirse como: proceso de asignación de un documento a la categoría (idioma) con la cual tiene un mayor número de características en común. La asignación o no, del documento a la categoría final, puede estar regida por valores empíricos establecidos con anterioridad.

El esquema general de la identificación de idioma, representado en la **Figura 9,** muestra como se requiere iniciar con un conjunto de documentos clasificados a priori (Colección de Entrenamiento) usándolos para "entrenar" el sistema. Estos documentos deberán estar organizados en categorías, las cuales serán los idiomas posibles a identificar.

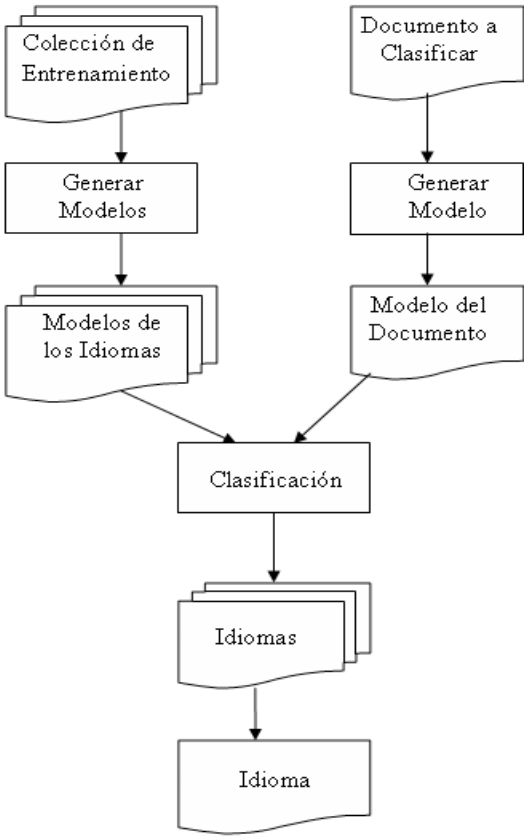


Figura 9: Esquema General de la Identificación Automatizada de Idiomas.

Por cada categoría o idioma se genera un modelo (*Generar Modelos)* donde son representadas las características significativas del documento que la compone, obteniéndose los *Modelos de los Idiomas.* Para un texto de entrada, también será necesario que transcurra por los pasos anteriores (*Generar Modelo*) para obtener el *Modelo del Documento.*

Por cada categoría o idioma se genera un modelo (*Generar Modelos)* donde son representadas las características significativas del documento que la compone, obteniéndose los *Modelos de los Idiomas.* Para un texto de entrada, también será necesario que transcurra por los pasos anteriores (*Generar Modelo*) para obtener el *Modelo del Documento.*
En la parte inferior de la figura se muestra la Clasificación; el modelo del documento es comparado con el modelo de cada idioma. De esta comparación se obtiene un valor numérico que expresa la relación del documento con el idioma. Finalmente es seleccionado el idioma más parecido al documento en dependencia del algoritmo de clasificación utilizado.

1.8.1. Uso de Palabras Vacías (StopWords)

Se denominan "stopwords" a palabras sin contenido semántico, es decir, que desde un punto de vista no-lingüístico contienen poca información. Estas características las hacen ser descartadas en tareas del PLN como recuperación de información y categorización automatizada de texto, entre otras. Sin embargo pueden ser útiles para la identificación de idiomas (GREFENSTETTE, 2005), pues son comúnmente más usadas que otras dentro de un texto. Las "stopwords" son: artículos, preposiciones, conjunciones, entre otras.

a	cual	es	fue
al	cuando	esa	han
algo	de	esas	has
algunas	del	ese	hasta
algunos	desde	eso	la
ante	donde	esos	mi
antes	durante	esta	otra
como	e	este	por
con	el	esto	sus
contra	ella	estos	tanto

Tabla 1: Algunas Palabras Vacías del Idioma Español.

La tabla anterior (**Tabla 1**) muestra una pequeña parte de la lista[8] de palabras vacías para el idioma español. Es posible, además, encontrarlas en varios idiomas.

A pesar de su uso, en varios sistemas de identificación automatizada de idiomas para textos, podría considerarse que presenta algunas desventajas similares, al uso de palabras, presentados en la sección *Palabras y n-gramas.* Esta técnica ha demostrado funcionar relativamente bien especialmente para textos de entrada grande y poca cantidad de idiomas disponibles.

1.8.2. Algunos Trabajos en la UCI

La UCI ha estado dando sus pasos en el área de la identificación automatizada de idioma con resultados favorables, (VAZQUEZ, LABRADA y GORDALES, 2008), con la implementación en Python[9] de un sistema. Obtienen del texto uni-gramas y bi-gramas de palabras, además utilizan el modelo vectorial para la representación de los documentos; el peso de los términos del vector se obtiene de la frecuencia del término en cuestión entre la cantidad de uni-gramas, si dicho término es un uni-grama, o la cantidad de bi-grama, en caso de que el término sea un bi-grama. La similitud entre los modelos de los documentos se calcula a partir de la Función del Coseno. Hasta el momento identifica dos idiomas: español e ingles.

1.8.3. Algunos Trabajos en el Ámbito Internacional

Varios son los estudios relevantes en esta área como (CAVNAR y TRENKLE, 1994), (GREFENSTETTE, 1995), (POUTSMA, 2002) y (GAYO, 2005), entre otros.

1.8.3.1. La Clasificación del Texto

William B. Cavnar y John M. Trenkle propusieron en 1994 un algoritmo para la tarea de clasificación de texto, aplicable a la identificación de idioma. Los documentos son representados como vectores de n-gramas de cantidades limitadas. El pesado de los términos del vector, se obtiene de su frecuencia de aparición en el documento.

8 La lista de las stopwords se obtuvo a partir del módulo de Perl Lingua::StopWords - Stop words for several languages, público en http://cpan.org

9 Lenguaje de programación creado por Guido van Rossum en el año 1990.

La clasificación es realizada comparando cada modelo de categoría con el modelo del documento a clasificar, usando *out-of-place* ("fuera-de-lugar"). El sistema creado demostró en las pruebas un cierto nivel de tolerancia a errores textuales, alcanzando un acierto del 99.8%.

1.8.3.2. Aplicación de Técnicas de Monte Carlo

Arjen Poutsma SmartHaven en el 2002 propuso una nueva técnica de identificación de idiomas basada en el muestreo de Monte Carlo. Demostró que crear un modelo del documento completo para conocer el idioma en el que ha sido escrito, es innecesario. Con una pequeña porción del contenido del documento es posible obtener las características que aporten la información requerida. Se extraen las características de un subconjunto lo suficientemente abarcador del documento; creando un modelo dinámico al cual se le aumenta su tamaño hasta tener las características suficientes, o sea, se toma una pequeña porción del documento y se le extraen sus características; en caso de no ser suficiente para identificar el idioma, se toman más características y se aumenta el modelo del documento.

En la Técnica de Monte Carlo se distingue el uso de dos etapas fundamentales: el modelado donde se crean las características distintivas de los idiomas que será capaz de identificar el sistema y una segunda etapa, clasificación, donde se compara el texto de entrada con cada modelo de idioma creado anteriormente, asignándole al documento entrado uno de los idiomas de los documentos de la colección, que sería el más parecido al documento a identificar.

A. Poutsma mejoró el tiempo de ejecución hasta en 85 veces del tiempo empleado por la mejor de las técnicas hasta el momento, aunque no logra una precisión tan exacta.

1.8.3.3. Técnica BlindLight

Daniel Gayo en el 2005 propone una nueva técnica que permite, entre otras cosas, la identificación del idioma. No se propone lograr una técnica que supere la exactitud alcanzada por la más precisa existente. Propuso probar su implementación con documentos cortos o con ruido (errores ortográficos, cabeceras de correo electrónico, etiquetas HTML).

Utiliza la descomposición del texto mediante n-gramas; no considera a los documentos, vectores en un espacio T-dimensional sino que dos vectores pueden tener distintas dimensiones. El valor real asignado a cada n-grama de caracteres

se denomina "significatividad"; es obtenido a partir de la frecuencia relativa y calculando la relación entre los caracteres constituyentes de cada n-grama, para ello emplea la Información Mútua (SI_f). Para calcular la similitud entre dos vectores, se obtiene un nuevo vector mediante la intersección de los dos anteriores y se compara la significatividad total del vector resultante con la de los vectores originales.

1.8.3.4. Programas En Línea (on-line)

Es posible encontrar varios software de identificación de idioma, que permiten probar su funcionamiento y en algunos casos hasta descargar el código, ejemplos de ellos son:

Textcat[10], software basado en el método propuesto por Cavnar y Trenkle, liberado bajo licencia General Public License[11] (GPL), permite identificar 69 idiomas y está desarrollado en Perl[12].

BlindLight[13] es la materialización de la propuesta de Gayo Avello que permite realizar varias tareas PLN:

- Resumen automático.
- Extracción de palabras clave.
- Clasificación o categorización.
- Recuperación de información.
- Identificación del lenguaje: Identifica 49 idiomas.

1.9. Herramientas

Para el desarrollo de sistemas es de gran utilidad el uso de herramientas que faciliten el trabajo a realizar, como:

10 http://www.let.rug.nl/~vannoord/TextCat/

11 http://www.gnu.org/licenses/licenses.html

12 Practical Extraction and Report Language. Lenguaje Práctico para la Extracción e Informe, es un lenguaje de programación diseñado por Larry Wall creado en 1987.

13 http://blindlight.uniovi.es/bLdemo/languageid.index.php

Los Sistemas Operativo (SO): conjunto de elementos de software que posibilitan y simplifican notablemente el manejo, en este caso, de una computadora pues suministra: una interfaz al usuario y administración de recursos, archivos, tareas, entre otras funciones.

El lenguaje de programación: conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Algunos ejemplos son: C++, C#, Java, Python y Perl, entre otros.

Con el objetivo de cumplir acuerdos establecidos para la creación del MCADHTML y apoyando el avance del país en el esfuerzo de la migración al software libre se estableció usar para el desarrollo del IDENAIT las siguientes herramientas:

Lenguaje de programación: Perl (Lenguaje Práctico para la Extracción e Informe), diseñado por Larry Wall en 1987. Es de propósito general; posee un robusto sistema de procesamiento de texto y ficheros; permite, entre otras tareas, la administración de sistemas y desarrollo web. Es software libre y está licenciado bajo la Licencia Artística y la GPL. Existen distribuciones disponibles para la mayoría de los sistemas operativos. Cuenta con una gran cantidad de documentación actualizada (PERLDOC, 2008), (PERL EN ESPAÑOL, 2008) y módulos[14] públicos en Comprehensive Perl Archive Network (CPAN), (CPAN, 2008); CPAN es un enorme archivo de software escrito en Perl, así como de documentación sobre el mismo. Posee 13666 módulos para una amplia variedad de tareas. Esencialmente, todo lo que hay en CPAN está disponible de forma libre. La mayor parte del software está licenciado bajo la Licencia Artística, la GPL o ambas.

Módulos de Perl: File::Spec::Fuctions[15](permite realizar operaciones con nombres de ficheros y directorios); File::Remove[16](para eliminar ficheros y directorios) y YAML[17].

Sistema Operativo: Debian GNU/Linux[18] v4.0 (Etch). El Proyecto Debian es una asociación de personas con el objetivo de crear un SO libre; tal SO es llamado Debian GNU/Linux, el cual viene con más de 18733 paquetes (software precompilado y empaquetado en un formato amigable para una instalación sencilla en su máquina), usa el núcleo de Linux (pieza de software

14 librerías de código externas, permitiendo que un simple fichero contenga rutinas comunes a varios programas.

15 http://search.cpan.org/~kwilliams/PathTools-3.2701/lib/File/Spec/Functions.pm

16 http://search.cpan.org/~adamk/File-Remove-1.41/lib/File/Remove.pm

17 http://search.cpan.org/~ingy/YAML-0.66/lib/YAML.pm

18 http://www.debian.org/

creada en un principio por Linus Torvalds y soportada por miles de programadores a lo largo del mundo) y gran parte de las herramientas básicas que completan el SO, vienen del *proyecto GNU*[19].

Entorno Integrado de Desarrollo (IDE): Anjuta[20] v2.2. Anjuta permite la programación en SO GNU/Linux en varios lenguajes como C, C++ y Perl, por mencionar algunos; es software libre, liberado bajo la licencia GPL. Incluye varias herramientas que facilitan el trabajo del ingeniero, como un editor que verifica y resalta la sintaxis escrita, entre otros.

Otras herramientas que podrían resultar interesantes para la tarea de indentificación de idiomas, aunque no son utilizadas en el desarrollo de IDENAIT, son:

"Lingua::Identify - Language identification"[21], módulo de Perl que se encuentra público en la CPAN. Utiliza las funciones *langof y langof_file* para identificar el idioma a partir de un texto o un fichero respectivamente, recibe además un conjunto de parámetros de configuración. Su funcionamiento se basa en el uso de los métodos: *smallwords, prefixes1, prefixes2, prefixes3, prefixes4, suffixes1, suffixes2, suffixes3, suffixes4, ngrams1, ngrams2, ngrams3 and ngrams4*. Permite identificar 36 idiomas y calcular el nivel de confianza para los resultados obtenidos. Otras de sus características:

- Es libre y es código abierto.
- Es portable.
- Es flexible pues permite elegir: el tamaño máximo de la entrada a analizar y los métodos a utilizar y su importancia.
- Es fácil tratar con los idiomas.
- Es mantenible.

"Text::Language::Guess - Trained module to guess a document's language"[22], módulo de la CPAN, permite la identificación del idioma a partir de la búsqueda de stopwords en el documento a identificar. Tiene soporte para 10 idiomas: Inglés(en), Francés(fr), Español(es), Portugués(pt), Italiano(it), Alemán(de), Holandés (nl), Sueco (sv), Noruego (no) y Danés (da).

19 El proyecto GNU fueiniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU

20 http://www.anjuta.org/

21 http://search.cpan.org/~cog/Lingua-Identify-0.19/lib/Lingua/Identify.pm

22 http://search.cpan.org/~mschilli/Text-Language-Guess-0.02/lib/Text/Language/Guess.pm

1.10. Metodología para el Desarrollo de Software

La evolución de la Industria del Software dio lugar al desarrollo de metodologías que le permiten a los ingenieros guiar con mayor precisión su trabajo para lograr productos de software bien acabado y que cumplan las métricas de calidad establecidas.

Para el desarrollo de cualquier producto de software es importante la búsqueda de la metodología que más se adapte a las condiciones con que se cuenta para desarrollar determinados proyectos (requerimientos, límite de tiempo, calidad, entre otros).

Existen numerosas propuestas metodológicas entre las que se encuentran las metodologías pesadas o tradicionales y las ágiles (ISSI, 2003).

Las metodologías tradicionales ponen gran énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos que se deben producir, incluyendo modelado y documentación detallada. Este enfoque tradicional es conveniente para proyectos que requieran mucho tiempo y recurso, pero no resulta muy adecuado para proyectos donde el entorno del sistema es muy cambiante y es necesario reducir los tiempos de desarrollo (CANÓS, 2008).

Las metodologías ágiles están orientadas especialmente a proyectos pequeños y permiten que los programadores se concentren solamente en aquellas funciones que se necesitan inmediatamente. Se realizan entregas al cliente frecuentemente, de las cuales se obtiene retroalimentación constante, posibilitando respuestas rápidas a los cambios en el negocio.

La **Tabla 2** muestra algunas de las diferencias entre las metodologías ágiles y tradicionales.

De acuerdo a las características del IDENAIT se optó por elegir una metodología ágil para desarrollar el sistema; la más destacada entre tales metodologías es Extreme Programming (XP), por tanto, se le dedicará especial atención.

Metodologías Ágiles	Metodologías Tradicionales
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2: Diferencias entre las Metodologías Ágiles y Tradicionales.

1.10.1. Extreme Programming

XP es una metodología ágil centrada en promover el trabajo en equipo como clave para el éxito. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Las características fundamentales de esta metodología son (LETELIER y PENADÉS, 2008):

- **Desarrollo iterativo e incremental:** Se le va incorporando mejoras, una tras otras.
- **Pruebas unitarias continuas:** Frecuentemente repetidas para probar el funcionamiento del módulo que se está desarrollando, incluyendo pruebas de regresión.
- **Programación en parejas:** Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto, esto posibilita que el código sea revisado y debatido mientas se escribe.

- **Frecuente integración del equipo de programación con el cliente o usuario:** Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo para una mayor comunicación.
- **Corrección de todos los errores antes de añadir nueva funcionalidad**. Es importante hacer pruebas de integración antes de continuar con el desarrollo y hacer entregas frecuentes.
- **Refactorización del código**, es decir, 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 compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve la idea de 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 en el código**: Realizar las funciones necesarias de la manera más sencilla posible.

Teniendo en cuenta las características del IDENAIT se optó por la metodología XP para el proceso de desarrollo del sistema. A continuación se exponen algunas razones que propiciaron su elección:

Poca disponibilidad de personal: El sistema a desarrollar es realizado por pocas personas, uno de los principios de XP es la programación en equipos pequeños (de 2 a 12 personas).

Pocos Roles: Por la poca disponibilidad de personal, no es posible la existencia de muchos roles, por lo tanto los integrantes del equipo pueden cambiar responsabilidades en un momento determinado, esta metodología presenta pocos roles siendo ideal para el desarrollo del sistema.

Proyecto pequeño: XP está concebido para ser utilizado en proyectos pequeños.

El cliente forma parte del equipo de desarrollo: En el desarrollo del sistema se está constantemente trabajando en función de las necesidades del cliente, logrando una mayor comunicación y retroalimentación, para asegurar un producto final satisfactorio, la metodología utilizada promueve esta idea.

1.11. Trabajos Similares

En el presente trabajo, se consideran afines aquellos identificadores automatizados de idiomas para textos desarrollados específicamente para el filtrado de contenido[23].

Hasta la fecha, a nivel nacional incluyendo la UCI, no se tiene noción de trabajos similares pues el único filtro de contenido creado en Cuba ha sido Filpacon; para el cuál se desarrollará la herramienta MCADHTML.

Sin embargo, existen varios sistemas de filtrado de contenidos entre los cuales se pueden mencionar:

PureSight (PURESIGHT, 2008): herramienta de software desarrollada en Israel, por iCognito Technologies Ltd, con una efectividad de 99,2%. Utiliza la tecnología *Reconocimiento Artificial de Contenido*, del inglés Artificial Content Recognition (ACR), dotada de algoritmos de Inteligencia artificial permitiéndole al software entender y clasificar el material según su contenido y decidir si la página es adecuada para mostrarse. ACR está capacitado para Inglés (en) y Alemán (de) con un nivel de efectividad de 99,2% y logra, además, el 94% de precisión en otros idiomas, incluyendo Francés (fr), Español (en), Holandés (nl) y Portugués (pt).

Optenet (OPTENET, 2008): herramienta de software desarrollada en España. Tiene un 97% de efectividad con 0,1% tasa de error. Para la categorización de documentos, entre otros métodos, utiliza el *Analizador Semántico de Contenidos* mediante técnicas de inteligencia artificial, para lograr el análisis semántico completo de las páginas solicitadas. Realiza el análisis en 7 idiomas: Español (es), Inglés (en), Francés (fr), Alemán (de), Holandés (nl), Italiano (it) y Portugués (pt).

Es de suponer que estos sistemas, PureSight y Optenet, requieren de alguna herramienta para la identificación del idioma de los textos, para llevar a cabo la categorización de contenidos; sin embargo, y a pesar de usar algoritmos de inteligencia artificial, han sido descartados por ser comerciales y no tener ninguna referencia sobre su código. No obstante existen otros

23 Los filtros de contenidos son programas informáticos que permiten o deniegan el acceso, intencionado o accidental, a contenidos no apropiados de Internet.

software, principalmente los mencionados en la sección *Identificación Automatizada de Idiomas para Textos,* que fueron desechados inicialmente por ser desarrollados con otro fin y de los cuales se podrían adoptar algunas características.

1.12. Conclusiones

Al concluir este capítulo se evidencia la necesidad de usar técnicas del Procedimiento del Lenguaje Natural para desarrollar un identificador de idiomas para textos. El uso de n-gramas para la representación de los documentos en la identificación de idiomas es, en algunos aspectos, más favorable que el uso de palabras para representar dichos documentos. Dada la cantidad de funcionalidades y características que posee el lenguaje de Programación Perl, demuestra ser una herramienta a tener en cuenta para tareas de procesamiento de textos. Se ha concluido además, utilizar la metodología XP para guiar el proceso de desarrollo de IDENAIT.

2. CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

Después de abordar, en el capítulo anterior, técnicas y tendencias del proceso de identificación automatizada de idiomas para textos, se está en condición de confeccionar una propuesta del IDENAIT. Por tanto, durante este capítulo se expresa el objetivo estratégico de la organización, se especifica el objeto de automatización, la información que se maneja y la propuesta del sistema.

2.2. Objetivos Estratégicos de la Organización

El grupo API surge en el 2008 como parte del polo CENTERNET; promueve, como se había mencionado anteriormente, el desarrollo de un MCADHTML que permita, entre otras funciones, la creación y actualización, de manera automatizada, de la información de la base de datos utilizada por el producto Filpacon. Para ello, MCADHTML deberá ser capaz de categorizar páginas web en base a su contenido.

Para el funcionamiento de módulos internos del MCADHTML, entre ellos el Categorizador Automatizado de Texto y el Extractor Automatizado de Resúmenes, se requiere, entre otros aspectos, conocer el idioma del texto de la página web entrada.

El grupo API no cuenta con la herramienta que realice el proceso de identificar el idioma de los textos y permita cumplir con los requerimientos necesarios para el funcionamiento del MCADHTML.

2.3. Objeto de Automatización

Internet es una fuente inagotable de información; cuenta con un gran número de páginas y muchas otras son creadas diariamente[24]. Para lograr la categorización de cantidades significativas de esta información, será necesario automatizar todos los procesos del MCADHTML; entre ellos la identificación de idiomas de los textos. Por tal motivo, se ha decidido la implementación de IDENAIT, sistema dedicado a la Identificación de idiomas en textos.

24 http://www.laflecha.net/canales/blackhats/en-2007-el-numero-de-paginas-web-crecio-un-48

2.4. Información que se maneja

MCADHTML en su labor de categorización deberá permitir el procesamiento de varios tipo de información que aporte un documento HTML, como imágenes, contenido textual, entre otros. No obstante, para el funcionamiento del IDENAIT se necesitará únicamente el texto de las páginas, guardado en ficheros dentro de un directorio específico, del cual leerá el sistema. Los ficheros deberán tener las siguientes características: ser texto plano[25], la codificación de sus caracteres en UTF-8, no deberá tener ningún tipo de marcado como Marca de Orden de Bytes (BOM, por el inglés *Byte Order Mark*)[26] que emplean algunos editores de Windows (UNICODE, 2008) y su contenido debe tener no menos 50 palabras.

Estos términos son establecidos con el objetivo de ofrecer mayor confiabilidad a los resultados que se obtendrán del IDENAIT.

2.5. Características de la propuesta del sistema

Para el desarrollo de software es necesario establecer una estructura de los componentes que lo integran de manera que soporte las funcionalidades requeridas. La arquitectura candidata del MCADHTML se puede observar en la **Figura 10**, donde se muestran sus componentes principales, la forma en que éstos deben interactuar y coordinar para llevar a cabo el objetivo del sistema.

Por su parte, el IDENAIT tiene una tarea fundamental dentro del MCADHTML como se ha mencionado anteriormente. Tendrá un funcionamiento basado en dos modos principales, *conectado* y *no conectado*. El modo conectado se refiere a su desempeño dentro del MCADHTML. IDENAIT mantendrá una comunicación bidireccional, a través del sistema *Comunicador*, con el módulo *Administrador Automatizado de Categorización,* recibiendo necesidades de identificación y ofreciendo el resultado una vez concluido.

25 Los archivos de texto plano están compuestos únicamente por texto sin formato, sólo caracteres; carecen de información destinada a generar formatos (negritas, subrayado, cursivas, tamaño, etc.) y tipos de letra.

26 Caracter empleado para marcar la orientación de escritura del flujo de bytes de una cadena de caracteres Unicode con código en UTF-16 o UTF-32 y/o como marca para indicar que el texto está codificado en UTF-8, UTF-16 o UTF-32

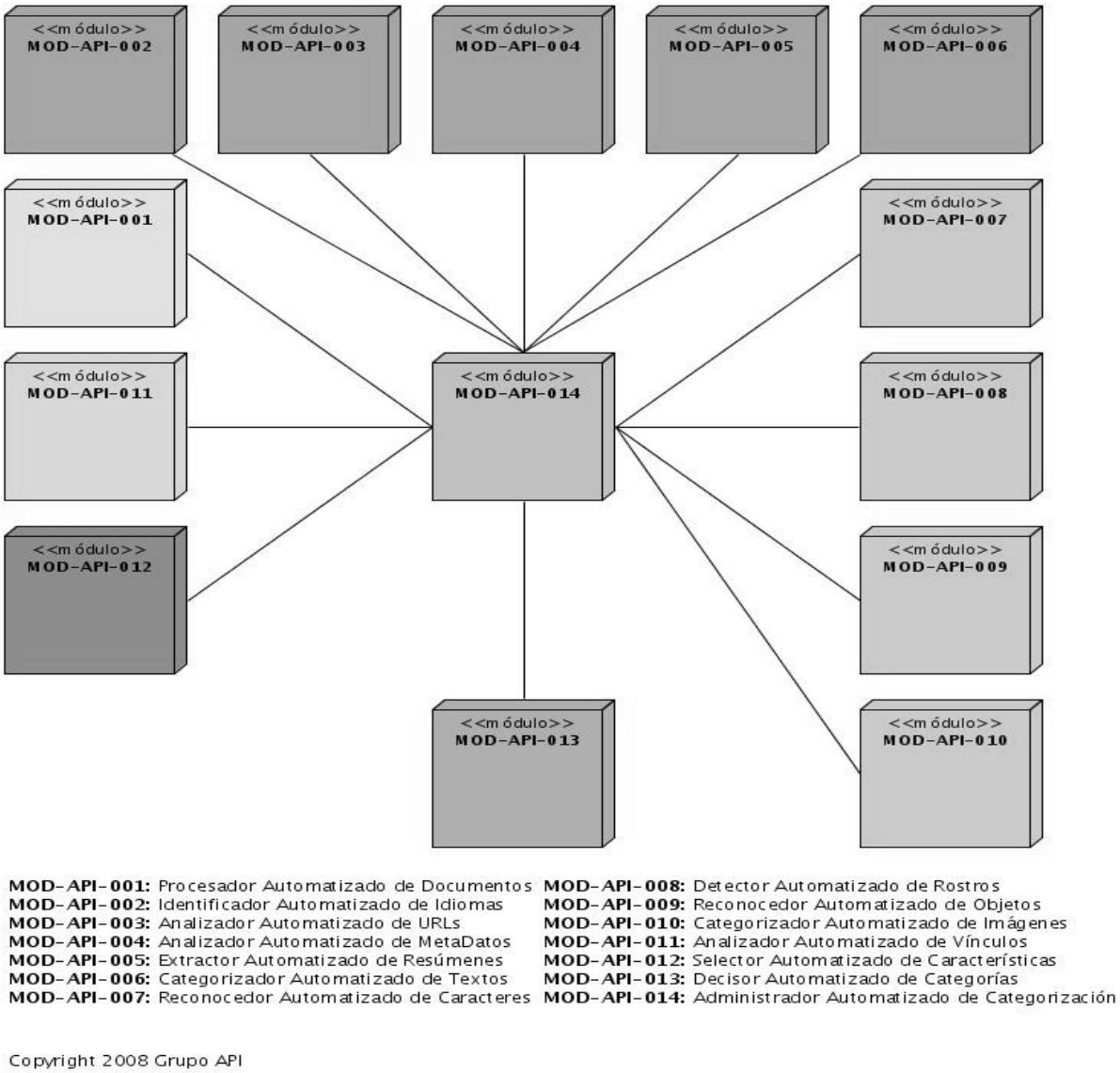


Figura 10: Arquitectura candidata MCADHTML

Por otra parte, el modo no conectado expresa su comportamiento fuera del MCADHTML donde un usuario, *ajustador*, pueda cambiar aspectos de su configuración y realizar pruebas (entrenar e identificar).

Se propone, además, que el sistema sea una aplicación de consola, pues de esta manera funcionará en modo conectado. Las acciones a realizar cuando el sistema esté no conectado son básicamente sencillas y no necesitará de una interfaz gráfica.

El sistema tiene dos fases fundamentales entrenar e identificar; por tal motivo se decidió separar el código en dos ficheros independientes: entrenar.pl e identificar.pl. Tal división ofrecerá la posibilidad de no ser necesario entrenar el software cada vez que se requiera identificar nuevos documentos, sino cuando: la Colección de Entrenamiento haya sido cambiada, se desee reducir el número de idiomas disponibles u otros parámetros de configuración. Una vista general de la propuesta del sistema puede observarse en la **Figura 11.**

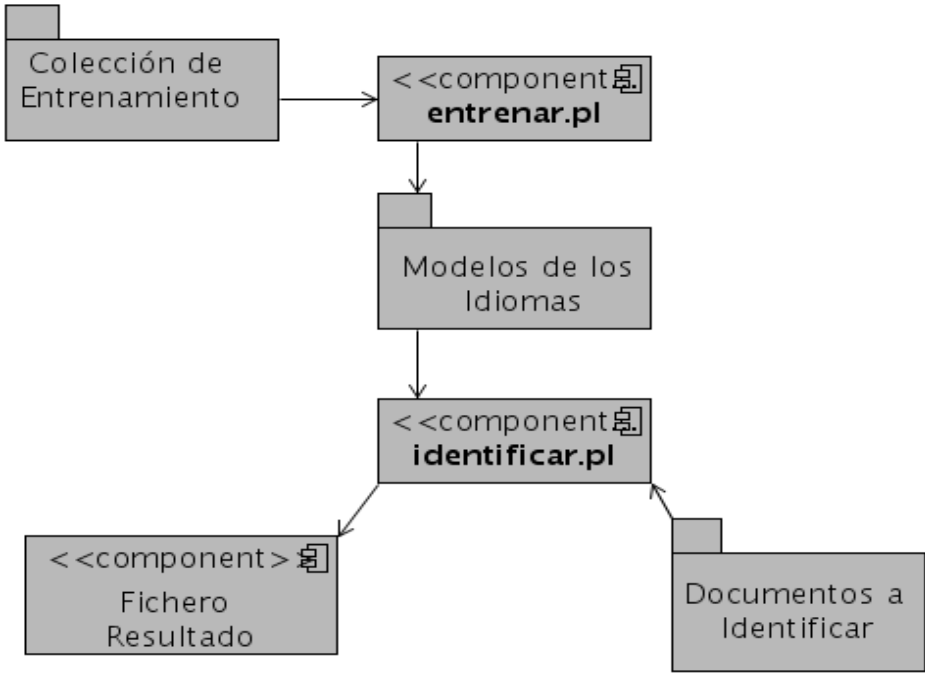


Figura 11: Propuesta del sistema IDENAIT

La Colección de Entrenamiento poseerá un documento por cada idioma que se desea identificar con IDENAIT; tales documentos, tendrán el mismo texto traducido al idioma al que representará.

Dada la dificultad de encontrar texto traducidos a un número significativo de idiomas y además de manera correcta, se propone usar La Declaración de los Derechos Humanos[27]. Estos textos serán archivados en ficheros de texto plano y con la codificación de los caracteres en UTF-8.

Por su parte, durante el entrenamiento del sistema se realizarán una serie de pasos en orden lógico, los cuales serán:

- Leer el Fichero de Configuración del sistema, en YAML[28]. En esta tarea se utilizará el módulo YAML de Perl.
- Leer el contenido de todos los documentos de la Colección de Entrenamiento (ubicados en un directorio específico al cual accederá el sistema). El contenido es procesado de la siguiente manera:
  - se eliminan algunos elementos como: números, signos de puntuación, caracteres no alfabéticos, entre otros. Tales caracteres no se desea que formen parte de la representación vectorial del documento.
  - se convierten todos los caracteres a minúsculas
  - se divide el texto en palabras. A cada una se le obtienen sus n-gramas y se conforma la lista de n-gramas del documento.
  - se cuenta la cantidad de n-gramas de cada tipo y se ordenan descendentemente. Esta lista ordenada es guardada en un fichero; de la lista total de n-gramas se seleccionan los más frecuentes para formar parte de la representación del documento; conformando así, el modelo del documento (idioma).

El modelo del documento será un vector cuyos términos son los n-gramas del texto y el peso de estos. Este modelo será almacenado igualmente en un fichero.

En la fase de clasificación también se accederá primeramente al Fichero de Configuración. Posteriormente se leen todos los modelos de los idiomas obtenidos durante el entrenamiento. Para cada documento a identificar (uno o varios) se realizará un proceso similar al descrito durante la fase anterior, obteniendo así: el modelo del documento. Tal modelo será comparado con todos los modelos de los idiomas usando el algoritmo descrito anteriormente como “out-of-place”. De esta comparación se obtendrá para cada documento una lista de relaciones: idioma - medida; la menor de estas medidas indicará entonces el idioma más probable del documento. El resultado de esta clasificación será registrado en un fichero (Fichero Resultado).

27 Declaración de los Derechos Humanos (http://www.unhchr.ch/udhr/navigate/alpha.htm).

28 Formato de serialización de datos.

El Fichero de Configuración leído por cada archivo (entrenar.pl e identificar.pl) al iniciarse, permitirá modificar algunos aspectos del sistema como: Idiomas que estarán disponibles, número de idiomas más probables que se mostrarán como resultado de la identificación de los textos, cantidad de términos (n-gramas) del documento a considerar para conformar el modelo de idiomas y documentos de entrada, el tamaño mínimo de los n-gramas obtenidos de cada palabra y la longitud máxima de los n-gramas. Este fichero le permitirá al usuario (*ajustador*) facilidades en la configuración del sistema e independencia del código para modificar tales parámetros.

Durante los procesos de entrenamiento y clasificación, el sistema guarda registros de su actividad: una lista de palabras y n-gramas por cada documento procesado, entre otras. Esta característica, ofrece facilidades en el seguimiento del funcionamiento del sistema como: encontrar con mayor facilidad posibles errores, comprobar los resultados obtenidos, por mencionar algunas.

IDENAIT será desarrollado para permitir la identificación de 39 idiomas los cuales son mostrados en la T**abla 3.**

639-1	Idioma
af	Afrikaans
br	Breton
da	Danish
nl	Dutch
en	English
fi	Finnish
fr	French
de	German
ko	Korean
id	Indonesian
lv	Latvian
it	Italian
pl	Polish

639-1	Idioma
ru	Russian
es	Spanish
sv	Swedish
sw	Swahili
pt	Portuguese
sq	Albanian
eo	Esperanto
co	Corsican
eu	Basque
bg	Bulgarian
el	Greek
mg	Malagasy
sk	Slovak

639-1	Idioma
ca	Catalan
fy	Western Frisian
hu	Hungarian
is	Icelandic
ga	Irish
lt	Lithuanian
cs	Czech
qu	Quechua
ro	Romanian
tl	Tagalog
tr	Turkish
uk	Ukrainian
cy	Welsh

Tabla 3: Lista de Idiomas identificados por IDENAIT

A pesar de la cantidad de idiomas que serán reconocidos por este software, no será objetivo de este trabajo validar la entrada para idiomas no sustentados por el sistema. Por cada documento entrado, se dará una lista de idiomas probables para su

texto. En caso de no tener al menos un elemento (n-grama) en común con ningún idioma, mostrará entonces idioma *no identificado*.

2.6. Conclusiones

Una vez expuestas las características que poseerá IDENAIT al ser desarrollado, se está en condiciones de realizar la Exploración, Planificación de la Entrega e Iteraciones del mismo y se considera que sea una herramienta que pueda contribuir al proceso de Identificación del Idioma dentro del MCADHTML.

3. CAPÍTULO 3: EXPLORACIÓN, PLANIFICACIÓN DE LA ENTREGA E ITERACIONES.

3.1. Introducción

La metodología XP se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promueve el trabajo en equipo y se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo. El ciclo de vida de XP consta con 6 fases: *Exploración*, *Planificación de la Entrega* (Release), *Iteraciones*, *Producción, Mantenimien*to y *Muerte del Proyecto* (BECK, 1999). En el presente capítulo se aborda lo concerniente al trabajo realizado en la fase de *Exploración* y *Planificación de la Entrega*, en el desarrollo del sistema propuesto.

3.2. Fase Exploración

En esta, la primera fase de XP, se elaboran las *Historias de Usuario* necesarias para comenzar el proceso de desarrollo y lograr la primera entrega del producto. Al mismo tiempo el equipo de trabajo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán para el desarrollo del IDENAIT.

3.3. Historias de Usuarios

Historias de Usuario es la técnica utilizada en XP para especificar los requisitos del software, tarjetas donde el cliente describe las características que el sistema debe poseer. Su contenido debe ser concreto y sencillo (CASTILLO, CEPERO y SANTANA, 2008), con el objetivo de poder programarse cada una en un tiempo aproximado de una a tres semanas, para no superar el tamaño de una iteración. Una explicación, con más detalles, de los campos que componen las tablas de Historias de Usuarios es mostrada en el Anexo I.

Durante esta fase, para el desarrollo del IDENAIT se elaboraron 3 Historias de Usuarios: *Entrenar* (**Tabla 4**), *Obtención del modelo para el documento a identificar* (**Tabla 5**) e *Identificar* (**Tabla 6**).

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Entrenar
Usuario: Comunicador o Ajustador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 3	**Iteración Asignada:** 1
Descripción: La fase de entrenamiento permite obtener un modelo para cada idioma, utilizando los textos de la Colección de Entrenamiento. Cada modelo tendrá las características más significativas del documento, es decir, del documento que representa al idioma en cuestión. Para lograr esto será necesario: determinar, para cada documento de la Colección de Entrenamiento, los n-gramas y ordenarlos descendentemente por su frecuencia de aparición	
Observaciones:	

Tabla 4: Historia de Usuario Entrenar.

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Obtención del modelo para el documento a identificar
Usuario: Comunicador o Ajustador	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: Para cada documento de entrada, será necesario obtener una representación (modelo) con sus características más significativas. Para ello se determinan todos sus n-gramas, los cuales son ordenados de forma descendente por sus frecuencias de aparición.	
Observaciones:	

Tabla 5: Historia de Usuario Obtención del modelo para el documento a identificar.

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Identificar
Usuario: Comunicador o Ajustador	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 3	**Iteración Asignada:** 3
Descripción: Se compara el modelo del documento a identificar con el modelo de cada idioma obtenidos en la fase de entrenamiento. El resultado de la comparación da la relación del documento en cuestión con cada idioma permitiendo obtener el idioma más probable al documento.	
Observaciones:	

Tabla 6: Historia de Usuario Identificar

3.4. Planificación

En la fase de planificación se establece por parte del cliente la prioridad de cada Historia de Usuario y los programadores obtienen la estimación del esfuerzo necesario. La medida para calcular el esfuerzo necesario es el *punto.* Un punto, equivale a una semana ideal de programación, generalmente una historia no excede los 3 puntos (BECK y FOWLER, 2000).
En esta fase se obtiene el número de iteraciones necesarias para la implementación del producto.

3.5. Estimación de esfuerzo por Historia de Usuario

Se hizo un estudio con el objetivo de estimar el esfuerzo necesario para realizar cada una de las Historias de Usuario, los cuales se reflejan en la tabla que se muestra a continuación (**Tabla 7**):

Historia de Usuario	Puntos estimados
Entrenar	3
Obtención del modelo para el documento a identificar.	2
Identificar	3

Tabla 7: Estimación de esfuerzos por Historia de Usuario

3.6. Iteraciones

En esta fase se incluyen las iteraciones por las que pasará el sistema antes de ser entregado. Además se definen los objetivos de cada iteración y el cliente decide cuales Historias de Usuarios se implementarán en cada una de las iteraciones definidas. Al final de la última iteración se entrega la primera versión del producto final. El trabajo de la iteración es expresado en tareas de programación.
Se decidió dividir el trabajo en tres iteraciones, las cuales se detallan a continuación:

Primera Iteración: Se implementa la Historia de Usuario número uno, o sea la de alta prioridad, al concluir se contará con una primera versión (0.1) que será mostrada al cliente, como versión de prueba, para comprobar si satisface sus expectativas.

Segunda Iteración: En esta iteración se implementa la Historia de Usuario número dos, de prioridad media, al terminar dicha iteración se contará con una nueva versión del producto (0.2), a la cual se le realizarán pruebas para verificar si cumple con las funcionalidades requeridas. El cliente observará dichas pruebas con el objetivo de realizar cambios en el sistema en proceso de desarrollo, si lo opina conveniente.

Tercera Iteración: En esta iteración se implementa la Historia de Usuario número tres, de prioridad baja, al finalizarla se contará con la primera versión del producto final (1.0) y se le harán pruebas al sistema completo para definir si cumple con todos los requerimientos necesarios.

3.7. Plan de duración de las Iteraciones

Como parte del ciclo de vida de cualquier proyecto usando XP se crea el plan de duración de las iteraciones, mostrando: las iteraciones involucradas en el proceso de desarrollo, las Historias de Usuario a desarrollar en cada iteración y el tiempo de duración estimado para cada una. El orden que poseen las Historias de Usuarios en la **Tabla 8**, será el orden a seguir para la implementación de las mismas.

Iteración	Orden de las Historias de Usuarios a implementar	Duración de la iteración
Iteración 1	Entrenar	3 semanas
Iteración 2	Obtención del modelo para el documento a Identificar.	2 semanas
Iteración 3	Identificar	3 semanas

Tabla 8: Plan de duración de las iteraciones

3.8. Plan de Entregas

A continuación se presenta el plan de entregas ideado para la fase de implementación. Al final de cada iteración se le realizarán pruebas al producto obtenido, en la fecha indicada en la **Tabla 9**. Al finalizar la tercera iteración se obtendrá la primera versión del producto final.

Módulo	Final 1ra Iteración 2da semana de febrero	Final 2da Iteración 4ta semana de febrero	Final 3ra Iteración 3ra semana de marzo
Identificador automatizado de idioma.	0.1	0.2	1.0

Tabla 9: Plan de entregas.

3.9. Conclusiones

En este capítulo se desarrollaron las fases de Exploración, Planificación de la Entrega e Iteraciones, se generaron las Historias de Usuarios necesarias para el correcto desarrollo del sistema, se organizó el trabajo por iteraciones, se estimó el tiempo necesario para desarrollar el producto y se elaboró un plan de entrega. Al finalizar la última iteración descrita en este capítulo, el sistema estará listo para entrar en producción.

4. CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS.

4.1. Introducción

Después de definidas las iteraciones, el equipo de desarrollo precisa el orden de las tareas de ingeniería por iteración para cada Historia de Usuario, o sea, se descomponen las Historias de Usuario en tareas de desarrollo; las cuales se asignan a un equipo o una persona responsable de su implementación. Dichas tareas son para el uso de los programadores, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente (BECK, 1999).

Al principio de cada iteración se revisa el plan de iteraciones y se modifica de ser necesario (BECK y FOWLER, 2000). Al finalizar cada iteración, cumpliendo con todas las tareas elaboradas, se debe obtener un producto funcional, el cual será probado y mostrado al cliente.

Para el desarrollo del IDENAIT se definieron tres iteraciones, en este capítulo se detalla cada iteración así como las tareas de desarrollo y las pruebas de aceptación.

4.2. Primera Iteración

En esta iteración se aborda la Historia de Usuario de mayor prioridad (ver **Tabla 10**), el equipo define las tareas de desarrollo con el fin de obtener un producto con las funcionalidades necesarias para mostrar al cliente y obtener criterios de este.

Historia de Usuario	Estimación	Real
Entrenar	3	3
Total	3	3

Tabla 10: Historia de Usuario abordada en la primera iteración

4.2.1. Tareas de las Historias de Usuarios Abordadas en la Iteración

Las Tareas de Ingienería referente a la primera iteración, han sido resumidas en las tablas a continuación (ver **Tabla 11, 12, 13 y 14**).

Para un mejor comprensión de cómo completar los campos de la tabla de Tareas de Ingeniería, remitirse al Anexo II.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Leer ficheros de la Colección de Entrenamiento y preprocesamiento del texto.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 20 enero 2008	**Fecha Fin:** 23 enero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Por cada elemento de la colección, se carga su información y se preprocesa el contenido, eliminando caracteres no significativos como signos de puntuación, números, etc.	

Tabla 11: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Entrenar

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Guardar las palabras de los documentos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 24 enero 2008	**Fecha Fin:**27 enero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Para cada documento de la Colección de Entrenamiento, se divide el texto preprocesado en palabras, las cuales son guardadas en fichero.	

Tabla 12: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Entrenar

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 1
Nombre Tarea: Obtener de los documentos de la Colección de Entrenamiento listas de n-gramas.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 28 enero 2008	**Fecha Fin:** 3 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Por cada palabra se obtienen sus n-gramas y se cuentan la cantidad de n-gramas por tipo. La lista de n-gramas es ordenada descendentemente por su frecuencia de aparición; sus elementos (n-grama-frecuencia) son almacenados en fichero.	

Tabla 13: Tarea de Ingeniería Nro. 3 para la Historia de Usuario Entrenar.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 1
Nombre Tarea: Obtención de los modelos de n-gramas de los documentos de la Colección de Entrenamiento.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 4 febrero 2008	**Fecha Fin:** 10 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: De cada lista de n-gramas, obtenida de los documentos de la Colección de Entrenamiento, se toman los 399 primeros n-gramas y se guardan en un fichero junto a su frecuencia.	

Tabla 14: Tarea de Ingeniería Nro. 4 para la Historia de Usuario Entrenar.

4.3. Segunda Iteración

En esta iteración se aborda la Historia de Usuario de prioridad media (ver **Tabla 15**) y se definen las tareas de desarrollo para la misma.

Historia de Usuario	Estimación	Real
Obtención del modelo para el documento a identificar	2	2
Total	2	2

Tabla 15: Historia de Usuario abordada en la segunda iteración

4.3.1. Tareas de las Historias de Usuario abordadas en cada Iteración

Las siguientes tablas: **Tablas 16, 17, 18 y 19**; han sigo organizados los datos referentes a las Tareas de Ingienería de la segunda iteración.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Preprocesamiento del documento a identificar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 11 febrero 2008	**Fecha Fin:** 14 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Se toma el documento a identificar y se preprocesa su contenido, eliminando los caracteres acordados como no significativos.	

Tabla 16: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Obtención del modelo para el documento a identificar.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Guardar en fichero las palabras de los documentos a identificar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 15 febrero 2008	**Fecha Fin:** 18 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: El texto del documento a identificar es separado en palabras. La lista de palabras es almacenada en ficheros.	

Tabla 17: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Obtención del modelo para el documento a identificar.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Obtener del documento a identificar, la lista de n-gramas.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 19 febrero 2008	**Fecha Fin:** 21 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Por cada palabra del documento a identificar, después del preprocesamiento del mismo, se obtienen sus n-gramas, se cuentan la cantidad de n-gramas por tipo. La lista de n-gramas es ordenada descendentemente y guardada en un fichero de la siguiente manera: n-grama-frecuencia.	

Tabla 18: Tarea de Ingeniería Nro. 3 para la Historia de Usuario Obtención del modelo para el documento a identificar.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Obtención del modelo de n-gramas del documento a identificar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 22 febrero 2008	**Fecha Fin:** 25 febrero 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: De la lista de n-gramas obtenida a partir del documento a identificar se toman los 399 primeros n-gramas y se guardan en un fichero junto a su frecuencia.	

Tabla 19: Tarea de Ingeniería Nro. 4 para la Historia de Usuario Obtención del modelo para el documento a identificar.

4.4. Tercera Iteración

En el transcurso de esta iteración se implementan las tareas de desarrollo concernientes a la última Historia de Usuario (ver **Tabla 20**). Concluida esta iteración, se consta de un producto listo para su puesta en funcionamiento.

Historia de Usuario	Estimación	Real
Identificar	3	3
Total	3	3

Tabla 20: Historia de Usuario abordada en la tercera iteración

4.4.1. Tareas de las Historias de Usuario abordadas en la iteración

Datos de las tareas de Ingeniería relacionadas con la tercera Historia de Usuario son mostradas en: **Tabla 21 y 22.**

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 3
Nombre Tarea: Comparar modelos de n-gramas	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.5
Fecha Inicio: 26 febrero 2008	**Fecha Fin:** 7 marzo 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: Compara el modelo del documento a identificar con cada uno de los modelos de n-gramas extraídos de la Colección de Entrenamiento. Por cada documento comparado se obtiene una lista de relaciones: (idioma, medida). Tal medida representará la posibilidad que tiene el documento de pertenecer a un idioma determinado	

Tabla 21: Tarea de Ingeniería Nro. 1 para la Historia de Usuario Identificar

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 3
Nombre Tarea: Obtener idiomas probables	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.5
Fecha Inicio: 8 marzo 2008	**Fecha Fin:** 19 marzo 2008
Programador Responsable: Yurisleidy Hernández Moya – Dionny Cardoso Carmona	
Descripción: La lista de relaciones idioma-medida de un documento es ordenada ascendentemente de acuerdo a la medida. De este ordenamiento se obtendrá una lista organizada por idiomas más probables, o sea, el primer idioma más probable será el de menor medida.	

Tabla 22: Tarea de Ingeniería Nro. 2 para la Historia de Usuario Identificar

4.5. Pruebas

Entre las prácticas de XP está la de seguir un desarrollo guiado por pruebas, posibilitando darle al cliente una idea de las verdaderas funcionalidades que tiene el producto. Mediante esta filosofía se reduce el número de errores no detectados, así como el tiempo entre la introducción de este en el sistema y su detección (Crispin y House, 2002). Realizándole al producto pruebas constantemente al final de cada iteración se eleva la calidad del mismo.
XP divide las pruebas en dos grupos:

Pruebas unitarias: desarrolladas por los programadores y encargadas de verificar el código de forma automática
Pruebas de aceptación: destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

Por cada iteración se traducen las Historias de Usuarios en pruebas de aceptación, con el objetivo de demostrar al cliente que el producto cumple con los requerimientos necesarios.

Por cada Historia de Usuario se pueden realizar todas las pruebas que se estimen convenientes. A continuación se mostrarán las pruebas de aceptación realizadas al sistema.

Detalles de como completar los campos de la tabla de Caso de Prueba de Aceptación son mostrados en el Anexo III.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU1-P1	**Número Historia de Usuario:** 1
Nombre: Leer y preprocesar el texto de los documentos de la Colección de Entrenamiento.	
Descripción de la Prueba: Se le realizará el preprocesamiento al texto de la Colección de Entrenamiento.	
Condiciones de Ejecución: El texto a leer y preprocesar debe estar en codificación UTF8 y ubicado en el directorio especificado por los programadores. Deben estar instalados los módulos de Perl File::Spec::Functions y Yaml. Y ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intentan eliminar los caracteres tomados como no significativos y guardar el texto preprocesado.	
Resultado Esperado: El documento es preprocesado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 23: Caso de Prueba de Aceptación HU1-P1

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU1-P2	**Número Historia de Usuario:** 1
Nombre: Obtener listas de n-gramas a partir de los documentos preprocesados de la Colección de Entrenamiento.	
Descripción de la Prueba: Se obtendrá una lista de n-gramas por cada documento de la Colección de Entrenamiento.	
Condiciones de Ejecución: El texto de la Colección de Entrenamiento a descomponer en n-gramas debe haber sido preprocesado satisfactoriamente. Debe estar instalado el módulo Yaml de Perl y debe ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intenta descomponer el texto preprocesado en listas de n-gramas y organizarlos en orden decreciente en relación a la frecuencia de aparición.	
Resultado Esperado: Se guarda en un fichero la lista de n-gramas del documento a identificar y su frecuencia de aparición.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 24: Caso de Prueba de Aceptación HU1-P2

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU1-P3	**Número Historia de Usuario:** 1
Nombre: Obtener modelos de n-gramas para la Colección de Entrenamiento.	
Descripción de la Prueba: Se obtendrán los modelos de n-gramas, para cada documento de la Colección de Entrenamiento.	
Condiciones de Ejecución: Debe haber sido sacada satisfactoriamente la lista de n-gramas de la Colección de Entrenamiento. Debe estar instalado el módulo Yaml de Perl y debe ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intenta dividir la lista de n-gramas, tomando los primeros 399 n-gramas organizados según su frecuencia de aparición.	
Resultado Esperado: Se guardan, en un fichero, los primeros 399 n-gramas de cada documento de la Colección de Entrenamiento y su frecuencia de aparición**.**	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 25: Caso de Prueba de Aceptación HU1-P3

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU2-P1	**Número Historia de Usuario:** 2
Nombre: Leer y preprocesar el texto del documento a identificar	
Descripción de la Prueba: Se le realizará el preprocesamiento al texto del documento a identificar.	
Condiciones de Ejecución: El texto a identificar debe estar en codificación UTF8 y ubicado en el directorio especificado por los programadores. Deben estar instalados los módulos de Perl File::Spec::Functions y Yaml. Y ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intentan eliminar los caracteres tomados como no significativos y guardar la nueva versión del texto preprocesado.	
Resultado Esperado: El documento es preprocesado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 26: Caso de Prueba de Aceptación HU2-P1

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU2-P2	**Número Historia de Usuario:** 2
Nombre: Obtener listas de n-gramas a partir del documento a identificar.	
Descripción de la prueba: Se obtendrá una lista de n-gramas del documento a identificar, a partir del texto preprocesado.	
Condiciones de Ejecución: El texto a identificar debe haber sido preprocesado satisfactoriamente. Debe estar instalado el módulo Yaml de Perl y debe ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intenta descomponer el texto preprocesado en listas de n-gramas y organizar dichos n-gramas en orden decreciente en relación a la frecuencia de aparición	
Resultado Esperado: Se guarda en un fichero la lista de n-gramas del documento a identificar y su frecuencia de aparición.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 27: Caso de Prueba de Aceptación HU2-P2

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU2-P3	**Número Historia de Usuario:** 2
Nombre: Obtener modelo de n-gramas del documento a identificar.	
Descripción de la prueba: Se obtendrá el modelo de n-gramas, para el documento a identificar.	
Condiciones de Ejecución: Debe haber sido obtenida satisfactoriamente la lista de n-gramas del documento a identificar. Debe estar instalado el módulo Yaml de Perl y debe ser capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intenta dividir la lista de n-gramas obtenida a partir del documento a identificar, tomando los primeros 399 n-gramas organizados según su frecuencia de aparición.	
Resultado Esperado: Se guarda en un fichero los primeros 399 n-gramas del documento a identificar y su frecuencia de aparición.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 28: Caso de Prueba de Aceptación HU2-P3

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU3-P1	**Número Historia de Usuario:** 3
Nombre: Cargar los modelos de n-gramas	
Descripción de la Prueba: Se cargarán los modelos de n-gramas generados a partir de la Colección de Entrenamiento y de los documentos a identificar.	
Condiciones de Ejecución: Que se hallan obtenidos los modelos de n-gramas de la colección y de los elementos a identificar satisfactoriamente, capaz de leer el Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intentan cargar los modelos de n-gramas generados a partir de la Colección de Entrenamiento y de los documentos a identificar	
Resultado Esperado: Documentos cargados.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 29: Caso de Prueba de Aceptación HU3-P1

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU3-P2	**Número Historia de Usuario:** 3
Nombre: Identificar el idioma de los documentos a identificar.	
Descripción de la Prueba: Se comparan los modelos de n-gramas extraídos de la Colección de Entrenamiento con los modelos de n-gramas de los documentos a identificar y se obtienen como resultado los idiomas más probables para los documentos a identificar.	
Condiciones de Ejecución: El sistema es capaz de leer del Fichero de Configuración.	
Entrada / Pasos de ejecución: Se intenta identificar el idioma más probable de los documentos a identificar.	
Resultado Esperado: Idiomas identificados.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 30: Caso de Prueba de Aceptación HU3-P2

4.6. Conclusiones

En este capítulo se abordaron las etapas de implementación y prueba del sistema a desarrollar. Se generaron todas las tareas de implementación para cada Historia de Usuario por cada una de las iteraciones así como las pruebas de aceptación realizadas sobre cada Historia de Usuario.

CONCLUSIONES

- La implementación del sistema (IDENAIT) dota al MCADHTML de una herramienta capaz de automatizar el proceso de identificar el idioma del texto de un documento HTML.
- El sistema implementado permite la identificación de 39 idiomas.
- El sistema resultante está provisto de un ambiente configurable, posibilitando modificar determinados parámetros.
- El uso de n-gramas es una técnica adecuada para la identificación de idioma pues dota al sistema desarrollado de robustez ante la presencia de errores.
- El modelo de espacio vectorial es apropiado para la identificación de idioma, pues permite de forma satisfactoria representar los documentos.
- El algoritmo utilizado, out-of-place, para el cálculo de similitud entre documentos, permite obtener resultados aceptables en la identificación de idioma y ser sencillo de implementar.
- Dada las características de XP, resultó adecuada para guiar el proceso de desarrollo del IDENAIT, ofreciéndole a los programadores claridad en las tareas a cumplir.

Al finalizar la implementación del IDENAIT se concluye que los objetivos propuestos han sido cumplidos satisfactoriamente. Se incluyen además una serie de recomendaciones a tenerse en cuenta para futuras versiones del mismo.

Recomendaciones

Otras funcionalidades pudieran incluirse en aras de lograr mejoras en la actividad del Identificador (IDENAIT). A continuación se exponen algunas recomendaciones:

- Enriquecer la Colección de Entrenamiento, con el objetivo de aumentar la gama de idiomas identificados y/o robustecer el contenido de los documentos representativos de los idiomas existentes.
- Incorporar, al identificador, la funcionalidad para el reconocimiento de idiomas no identificados.
- Incrementar el número de codificaciones de caracteres aceptados por el Identificador.

Otras recomendaciones:

Profundizar en el estudio de la metodología ágil empleada en este trabajo: Programación Extrema; la cual ha demostrado ser adecuada para el desarrollo de software con características en común con IDENAIT.

Referencias Bibliográficas

BECK, Kent. *Extreme programming explained: embrace chang*. [en línea]. United States of America. September 29, 1999. Addison Wesley. [fecha de consulta: 6 junio 2008]. Disponible en: http://www.google.com.cu/books?id=G8EL4H4vf7UC&printsec=frontcover&dq=extreme+inauthor:%22kent+beck%22&lr=&as_brr=0&sig=axkIphVYXSPwHCS7WVUvWx9_qog#PPT1,M1. ISBN: 0201616416. 224 p.

BECK, Kent y FOWLER, Martin. *Planning Extreme Programming*. [en líena]. Addison Wesley. 2000. [fecha consulta: 6 juno 2008]. Disponible en: http://www.google.com.cu/books?id=u13hVoYVZa8C&printsec=frontcover&dq=planning+extreme+programming&lr=&sig=v-67ooHZy8EmCUdREs7eABUBaQY#PPT1,M1. ISBN: 0201710919

BERROCAL, J.L. [et al] (2004). *LA RECUPERACIÓN DE INFORMACIÓN EN EL WEB: RETOS Y ¿SOLUCIONES?*. [en línea]. [Fecha de Consulta: 2 junio 2008]. Disponible en: http://reina.usal.es/pub/alonso2004recuperacion.pdf

CANÓS, José H. y LETELIER, Patricio y PENADÉS, Mª Carmen (2008). *Métodologías Ágiles en el Desarrollo de Software.* [fecha de consulta: 10 junio 2008]. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

CASTILLO Arzola, Niria, CEPERO Valdes, Alexis y SANTANS Cruz, Julio V. Sistema automatizado para el control de los expedientes de tutoría en la Sede Universitaria de Florencia. Revista de Informática Educativa y Medios Audiovisuales. [en línea]. Vol. 5, no. 10, págs. 34-55. 2008. [fecha consultada: 2 junio 2008]. Disponible en: http://www.fi.uba.ar/laboratorios/lie/Revista/Articulos/050510/A5mar2008.pdf ISSN 1667-8338

CAVNAR, William B. y TRENKLE, John M. (1994). *N-Gram-Based Text Categorization*. [fecha de consulta: 4 junio 2008]. Disponible en: http://lesfourmisrouges.com/bs/documentation/@%20work/sdair-94-bc.pdf

CPAN. *Comprehensive Perl Archive Network*. Randy Kobes. [en línea]. 26 octubre de 1995. [fecha consultada: 5 junio 2008]. Diponible en: http://cpan.org/

FIGUEROLA, Carlos G. [et al] (2005). *Algunas Técnicas de Clasificación Automática de Documentos*. [en línea]. [fecha consulta: 2 junio 2008]. Disponible en: http://reina.rec.usal.es/pub/figuerola2005algunas.pdf

FRESNO, V. D. *Representación Autocontenida de Documentos HTML: una propuesta basada en Combinaciones Heurísticas de Criterios*. Tesis (Doctoral). Móstoles, España: Universidad Rey Juan Carlos, Escuela Superior de Ciencias Experimentales y Tecnología, Departamento de Ingeniería Telemática y Tecnología Electrónica, 2006. 271 p. Disponible en: http://www.escet.urjc.es/~vfresno/phd_sp.html

GARCÍA M., María Jesús (2006). *Regulación y autorregulación en Internet: el control de los contenidos y los datos en la LSSI* . [en línea]. Disponible en: http://www.apdcat.net/media/303.pdf

GAYO Avello, Daniel. *BlindLight Una nueva técnica para procesamiento de texto no estructurado mediante vectores de n-gramas de longitud variable con aplicación a diversas tareas de tratamiento de lenguaje natural*. Tesis (Doctor). Oviedo, España: Universidad de Oviedo, Departamento de Informática, 2005. 213 p.

GIL L., Isidoro y RODRÍGUEZ M., José Vicente. *EL PROCESAMIENTO DEL LENGUAJE NATURAL APLICADO AL ANÁLISIS DEL CONTENIDO DE LOS DOCUMENTS*. [en línea]. *Revista General de Información y Documentación* [en línea]. Vol. 6, no. 2. 1996. [Fecha Consulta: 31 mayo 2008]. Disponible en: http://www.ucm.es/BUCM/revistas/byd/11321873/articulos/RGID9696220205A.PDF

GREFENSTETTE, Gregory. *Comparing two language identification schemes*. En: 3rd International conference on Statistical Analysis of Textual Data (1995, Roma, Fracia). 6 p. Disponible en: http://www.xrce.xerox.com/Publications/Attachments/1995-012/Gref---Comparing-two-language-identification-schemes.pdf

ISSI . *Metodologías Ágiles en el Desarrollo de Software*. En: Taller Metodologías Ágiles en el Desarrollo de Software. [en línea]. 12 Noviembre 2003. Taller realizado en el marco de las VIII Jornadas de Ingeniería del Software y Base de Datos. Alicante, España. 59 p. Disponible en: http://issi.dsic.upv.es/tallerma/actas.pdf

LETELIER, Patricio y PENADÉS, Ma. Carmen. (2008). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP) .* [en líena]. [fecha consultada: 5 junio 2008]. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/masyxp.pdf

OPTENET (2008). *Optenet Web Filter*. [fecha de consulta: 5 junio 2008]. Disponible en: http://www.optenet.com/es/index.asp

PERLDOC (2008). Jon Allen. [en línea]. [fecha consultada: 5 junio 2008]. Disponible en: http://perldoc.perl.org/perlintro.html

PERL EN ESPAÑOL, (2008). Uriel Lizama. [en línea]. [fecha consultada: 5 junio 2008]. Dsiponible en: http://perlenespanol.baboonsoftware.com/

POUTSMA, Arjen. (2002). *Applying Monte Carlo Techniques to Language Identification*. [fecha consultada: 4 junio 2008]. Disponible en: http://www.xs4all.nl/~ajwp/langident.pdf

PURESIGHT (2008). *Puresight Filter*. [en línea]. [fecha consultada: 5 junio 2008]. Disponible en: http://www.puresight.com/

SALTON, G., WONG, A. y YANG, C. S. *A Vector Space Model for Automatic Indexing.* Communications of the ACM [en línea]. Noviembre 1975, vol. 18, no. 11. [fecha de consulta: 5 junio 2008]. Disponible en: http://aidb.cs.iitm.ernet.in/cs625/salton-VectorSpaceModel.pdf ISSN: 0001-0782

SEBASTIANI, F. *Machine Learning in Automated Text Categorization. ACM Computing Surveys (CSUR)* [en línea]. Marzo 2002, vol. 34, no. 1. [fecha de consulta: 1-marzo-2008]. Disponible en: http://www.isti.cnr.it/People/F.Sebastiani/Publications/ACMCS02.pdf. ISSN: 0360-0300

UNICODE (2008). *Unicode home page*. [fecha de consulta: 1-febrero-2008]. Disponible en: http://unicode.org/faq/utf_bom.html

VALLEZ, Mari y PEDRAZA-JIMENEZ, Rafael. *El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines.* [en línea]. "Hipertext.net", núm. 5, 2007. [Fecha Consultada: 31 mayo 2008]. Disponible en: http://eprints.rclis.org/archive/00010700/01/El_Procesamiento_del_Lenguaje_Natural_en_la_Recuperaci%C3%B3n_de_Informaci%C3%B3n_Textual_y_%C3%A1reas_afines.pdf. ISSN 1695-5498

VAZQUEZ Acosta, Manuel y LABRADA Sedeño, Roberto O. y GORDALES Cruz, Miguel Y. *Detección automática del idioma de un texto mediante el uso de unigramas y 2-gramas*. *Serie científicos.* [en línea]. No. 2. enero 2008. [fecha de consulta: 4 junio 2008]. Disponible en: http://seriecientifica.uci.cu/repositorio/sc_200802/SC_200802_07.pdf

Bibliografía Consultada

AGUILAR Sierra, Alejandro. Programación Extrema y Software Libre. [en línea]. Octubre 2002. [fecha consultada: 4 de junio 2008]. Disponible en: http://www.seguridad.unam.mx/eventos/datos/ev11/semi18/mat.7.pon19.semi18.pdf

AGUILAR Sierra, Alejandro. Introducción a la Programación Extrema. [en línea]. Octubre 2002. [fecha consultada: 4 de junio 2008]. Disponible en: http://www.willydev.net/Descargas/Articulos/General/IntroXP.PDF

ARSENIO, Lprena. Menores y tecnologías de la información. PUBLICACIÓN DIGITAL DE LA AGENCIA DE CALIDAD DE INTERNET. [en línea]. Número 51. Barcelona. 2003. [fecha consultada: 4 de junio 2008]. Disponible en: http://magno.uab.es/epsi/mjgarcia.pdf

FIGUEROLA, C. G., ZAZO, A. F. y ALONSO, J. L. (2000). Categorización automática de documentos en español: algunos resultados experimentales. [fecha de consulta: 4 junio 2008]. Disponible en: http://reina.usal.es/pub/figuerola2000categorizacion.pdf

GÓMEZ Hidalgo José María [*et al*]. Categorización de texto sensible al coste para el filtrado de contenidos inapropiados en Internet. [en línea]. 2002. [fecha de consulta: 4 jinuo 2008]. Disponible en: http://www.esp.uem.es/jmgomez/papers/sepln03.pdf

MIRANDA, R. (2005). *Los menores en la Red: comportamiento y navegación segura.* [fecha consultada: 4 de junio 2008]. Disponible en: http://www.financialtech-mag.com/_docum/50_Tendencias_05.pdf

SOUTER, Clive [*et al*]. *Natural Language Identification using Corpus-Based Models.* [en línea]. 1994. [fecha consultada: 4 junio 2008]. Disponible en: http://hermes2.asb.dk/archive/FreeH/H13_15.pdf

VILLATE, Javier. Libertad de expresión en Internet. *OBSERVATORIO PARA LA CIBERSOCIEDAD.* [en línea]. Número 64. 20 de marzo 2002. [fecha de consulta: 4 junio 2008]. Disponible en: http://www.cibersociedad.net/archivo/articulo.php?art=37

XP Extreme Programming: A gentle introduction. 17 de Febrero 2006. Disponible en: http://www.extremeprogramming.org/

Anexos

Anexo I. Plantilla para especificación de Historias de Usuario.

Historia de Usuario	
Número:	**Nombre Historia de Usuario:**
Usuario:	
Prioridad en Negocio:	**Riesgo en Desarrollo:**
Puntos Estimados:	**Iteración Asignada:**
Descripción:	
Observaciones:	

Número: Número de la historia.

Usuario: El tipo de usuario que la ejecuta.

Nombre Historia de Usuario: Nombre que identifica la historia.

Prioridad en negocio: Prioridad que tiene la historia para la entidad. (Según cliente)

Riesgo en desarrollo: Grado de dificultad de implementación estimado. (Según programadores)

Puntos estimados: Tiempo en semanas que se le asignará. (Estimado)

Iteración asignada: En qué iteración se desarrollará. (Según su importancia)

Descripción: Breve descripción del proceso que define la historia.

Observaciones: Alguna acotación importante de señalar acerca de la historia.

Anexo II. Plantilla para especificación de Tareas de ingeniería.

Tarea de Ingeniería	
Número Tarea:	**Número Historia de Usuario:**
Nombre Tarea:	
Tipo de Tarea:	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable:	
Descripción:	

Número Tarea: Número de la tarea de ingeniería.

Número Historia de Usuario: Número que identifica la historia a la que pertenece la tarea.

Nombre Tarea: Nombre que identifica a la tarea.

Tipo de Tarea: Desarrollo / Corrección / Mejora / Otra (especificar).

Puntos estimados: Tiempo en semanas que se le asignará. (Estimado).

Fecha Inicio: Fecha de inicio de elaboración de la tarea.

Fecha Fin: Fecha donde concluye la elaboración de la tarea.

Programador responsable: Programador responsable de su implementación.

Descripción: Breve descripción del proceso que define la tarea.

Anexo III. Plantilla para especificación para los Casos de Prueba de Aceptación.

Caso de Prueba de Aceptación	
Código Caso de Prueba:	**Número Historia de Usuario:**
Nombre:	
Descripción de la Prueba:	
Condiciones de Ejecución	
Entrada / Pasos de ejecución:	
Resultado Esperado:	
Evaluación de la Prueba:	

Código Caso de Prueba: Código del Caso de Prueba de Aceptación.

Número Historia de Usuario: Número que identifica la historia a la que se le realizará la prueba.

Nombre: Nombre que define el Caso de Prueba de Aceptación.

Descripción de la Prueba: Breve descripción de los pasos a seguir para realizar la prueba.

Condiciones de Ejecución: Condiciones iniciales con las que debe cumplir el sistema o parte del sistema a desarrollar, para poder realizar la prueba.

Entrada / Pasos de ejecución: Pasos a seguir para lograr la correcta ejecución de la prueba.

Resultados Esperados: Descripción de los resultados que se esperan obtener.

Evaluación de la Prueba: Definir si fueron logrado o no los resultados esperados.

Glosario de Términos

Base de Datos: Conjunto organizado e integrado de datos almacenados en computadora, con el fin de facilitar su uso para aplicaciones con múltiples finalidades, de manera que sea posible acceder a la información de forma fácil y rápida.

Correo Electrónico: Servicio de red que permite a los usuarios enviar y recibir mensajes rápidamente mediante sistemas de comunicación electrónicos.

Dialecto: Variante particular de un idioma hablada en ciertas zonas geográficas.

Documento HTML: documento electrónico en el cual se ha empleado algunos tabs HTML.

FTP: Protocolo de transferencia de archivos entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente sea posible conectarse a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.

Grupos de Noticias (Newsgroups): Grupos de discusión formados por personas que tienen un interés común, dedicados a un asunto y organizado en una jerarquía. Los usuarios pueden recibir y enviar mensajes textuales sobre una gran variedad de temas.

HTML: Lenguaje de Marcas de Hypertexto. Lenguaje para elaborar paginas Web, usado para conformar la apariencia y contenido de un documento, así como para complementar el texto con objetos tales como imágenes.

Inteligencia Artificial: rama de la informática que desarrolla procesos que imitan a la inteligencia de los seres vivos**.**

La Web: sistema de documentos (o webs) interconectados por enlaces de hypertexto, que se ejecutan en Internet.

Recuperación de Información: Ciencia de la búsqueda de información en documentos**.**

Requisito: condición o capacidad que el sistema (software) debe cumplir.

URL: cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información, como documentos e imágenes, disponibles en la *Internet.* El URL de un recurso de información es su dirección en Internet, la cual permite que el navegador la encuentre y la muestre de forma adecuada. Por ello el URL combina el nombre del ordenador que proporciona la información, el directorio donde se encuentra, el nombre del archivo y el protocolo a usar para recuperar los datos (http, https, entre otros).

Utf-8: norma de transmisión de longitud variable para caracteres codificados utilizando Unicode. Usa grupos de bytes para representar el estándar de Unicode para los alfabetos de muchos de los lenguajes del mundo.

World Wide Web: sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet.