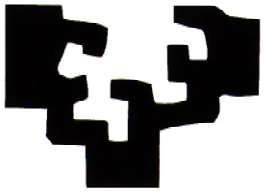
eman ta zabal zazu



Universidad del País Vasco

Euskal Herriko Unibertsitatea

Facultad de Informática

Informatika Fakultatea

TITULACION: Ingeniería Técnica en Informática de Sistemas

News Monitor: Desarrollo de un sistema de seguimiento de noticias

Alumno: Dersu García Sanz

Director: German Rigau Claramunt

Proyecto Fin de Carrera, noviembre de 2012

Índice

1 – Introducción

1.1 – Presentación

Este documento es la memoria de un Proyecto de Fin de Carrera del año 2012 realizado por Dersu Garcia Sanz, alumno de la Ingeniería Técnica en Informática de Sistemas y dirigido por German Rigau Claramunt, profesor de Ingeniería del Software en la Facultad de Informática (UPV) de San Sebastián.

El proyecto se enmarca en el área de la Ingeniería del Software con especial hincapié en la minería Web y el seguimiento de noticias. Para ser más exactos la finalidad del proyecto es desarrollar una aplicación que monitorice un portal llamado News Explorer[1], cuyo funcionamiento viene descrito en un apartado específico de la presente memoria. El proceso y las herramientas de monitorización serán descritos a lo largo de esta memoria.

1.2 – Un primer contacto

La finalidad de este apartado consiste en proporcionar una idea general del ámbito de trabajo en el que se desarrolla el proyecto.

En la Inteligencia Artificial[2] existen dos procesos que en la mayoría de las ocasiones van de la mano: la extracción de información o *data mining* y la utilización de dicha información para aprender nuevas reglas aplicables al mundo real o al modelo de mundo real de nuestro sistema inteligente.

En el caso del Procesamiento del Lenguaje Natural[3], dichas reglas se obtienen a medida que se analizan textos por medio de diferentes métodos. Uno por uno, en común, o realizando análisis léxicos, sintácticos y semánticos. Pero para ello, antes alguien tiene que obtener dichos textos. Y qué mejor manera de obtenerlos que automatizando su extracción.

Ahora que hemos entrado en el contexto podemos pasar a describir el funcionamiento y la estructura del portal EMM News Explorer, del que tenemos que extraer una serie de datos.

---

[1] http://emm.newsexplorer.eu/NewsExplorer/home/es/latest.html

[2] Según John McCarthy, quien acuñó el término, es la ciencia e ingeniería de hacer máquinas inteligentes, especialmente programas inteligentes de cómputo.

[3] El Procesamiento del Lenguaje Natural es una sub-disciplina de la Inteligencia Artificial que se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales.

1.3 – News Explorer

News Explorer es un sistema informático de obtención, análisis, clasificación y exploración de noticias multilingüe desarrollado por el Joint Research Centre (JRC) de la Comisión Europea. Forma parte de la familia de aplicaciones Europe Media Monitor (EMM), entre las cuales también se encuentran NewsBrief (agrupación y clasificación de noticias similares a lo largo del tiempo), MedISys (Medical Information System, especializada en detección y alerta de enfermedades en todo el mundo) y EMMLabs (conjunto de herramientas responsables entre otras cosas de monitorizar eventos violentos y desastres naturales). En la Figura 1 se muestra la página principal del EMM[4], que nos da la opción de acceder a la aplicación que más nos interese.

Figura 1: Página principal del EMM

A continuación exponemos brevemente los aspectos clave de la aplicación News Explorer.

1.3.1 – Funcionamiento de News Explorer

Todas las aplicaciones mencionadas se alimentan de los datos obtenidos por el motor central EMM, que recopila una media de 50.000 artículos al día de más de 1.500 páginas web en 42 idiomas. La aplicación News Explorer se centra en 19 de ellos para agrupar las noticias relacionadas en clústers o agrupaciones. Asimismo, también realiza un seguimiento de clústers similares a lo largo del tiempo y entre idiomas, y los muestra como una misma historia.

[4] http://emm.newsbrief.eu/overview.html

En la actualidad los idiomas que maneja News Explorer son: Árabe, Búlgaro, Danés, Holandés, Inglés, Estonio, Farsi, Francés, Alemán, Italiano, Noruego, Polaco, Portugués, Rumano, Ruso, Esloveno, Español, Sueco y Turco.

He aquí algunos de los componentes de análisis de textos utilizados en News Explorer:

- Agrupación (*clustering*) monolingüe de documentos.
- Reconocimiento y desambiguación de nombres de personas, organizaciones y lugares.
- Categorización de documentos mediante el diccionario de sinónimos multilingüe Eurovoc[5].
- Cálculo de semejanza entre clústers, tanto monolingüe como entre idiomas.

1.3.2 – Estructura de News Explorer

Una vez visto su funcionamiento, es necesario tener una idea de cómo está organizada la información para poder decidir qué nos interesa extraer y cómo podemos hacerlo. En la Figura 2 se muestra la página principal del News Explorer[6], en la que se pueden apreciar varias formas de acceder a su contenido.

Por una parte, a la izquierda podemos seleccionar la fecha y el idioma en el que queremos realizar la consulta de noticias.

A la derecha se muestran agrupados en bloques los países (Countries), las personas (People) y las entidades (Other Names) que se han nombrado directa o indirectamente en las noticias del día.

En el centro de la página podemos ver un mapa del mundo que nos permite saber, de un vistazo, dónde se ha producido cada noticia.

Debajo del mapa se muestran las agrupaciones o clústers de las noticias que más han dado de qué hablar a lo largo del día. Por ejemplo, vemos que el día 30 de octubre de 2012 News Explorer recopiló 32 noticias de diferentes medios en castellano sobre el huracán Sandy.

Además del título del clúster, también se muestra una descripción de la noticia y una lista de enlaces para ver la misma noticia en otros idiomas. En el caso del clúster sobre el huracán estos son los idiomas disponibles:

[5] http://eurovoc.europa.eu/
[6] http://emm.newsexplorer.eu/NewsExplorer/home/es/latest.html

Árabe (ar), búlgaro (bg), danés (da), alemán (de), inglés (en), persa (fa), francés (fr), italiano (it), holandés (nl), noruego (no), polaco (pl), portugués (pt), rumano (ro), ruso (ru), esloveno (sl), sueco (sv) y turco (tr).

Figura 2: Página principal de News Explorer

Siguiendo con el ejemplo anterior, si pulsamos en el enlace de un clúster determinado (El devastador rastro de 'Sandy'), llegamos a la página del clúster[7].

Como vemos en la siguiente figura, en la página del clúster de noticias tenemos varios grupos de elementos que podemos consultar.

[7] http://emm.newsexplorer.eu/NewsExplorer/clusteredition/es/20121030,abc-958a86ccebb2ed5234e9eea4498953a5.html

El primero y más importante es el grupo de noticias que compone el clúster. Aparece centrado y en primer plano y contiene un enlace a la dirección de cada noticia. Además para cada una de ellas se muestra la fuente de la noticia (el periódico) y la hora de publicación.

Debajo de las noticias se encuentra el apartado relacionado con la historia a la que pertenece el clúster. Además del enlace a la página de la historia se muestran las palabras clave que se han repetido en todas las noticias y que han provocado que se agrupen en una misma historia.

En este bloque también se muestra la historia a lo largo del tiempo representada por los clústers de noticias en el mismo idioma que se han recopilado en días anteriores.

Al igual que en la página principal de News Explorer, aquí también se muestran los países, personas y entidades mencionadas en las noticias. En este caso sin embargo, también aparece una lista de lugares (Places). Además cada uno de estos elementos viene acompañado por el número de veces que aparece en el clúster.

Por último, al final de la figura podemos ver un encabezado que dice: “Related Clusters – across languages”. En este apartado aparecen los enlaces a páginas de clústers relacionados en otros idiomas junto con la semejanza que guardan con el clúster actual. En la sección Captura de Requisitos de la presente memoria volveremos a hablar de ellos y especificaremos qué queremos recopilar de cada elemento de News Explorer.

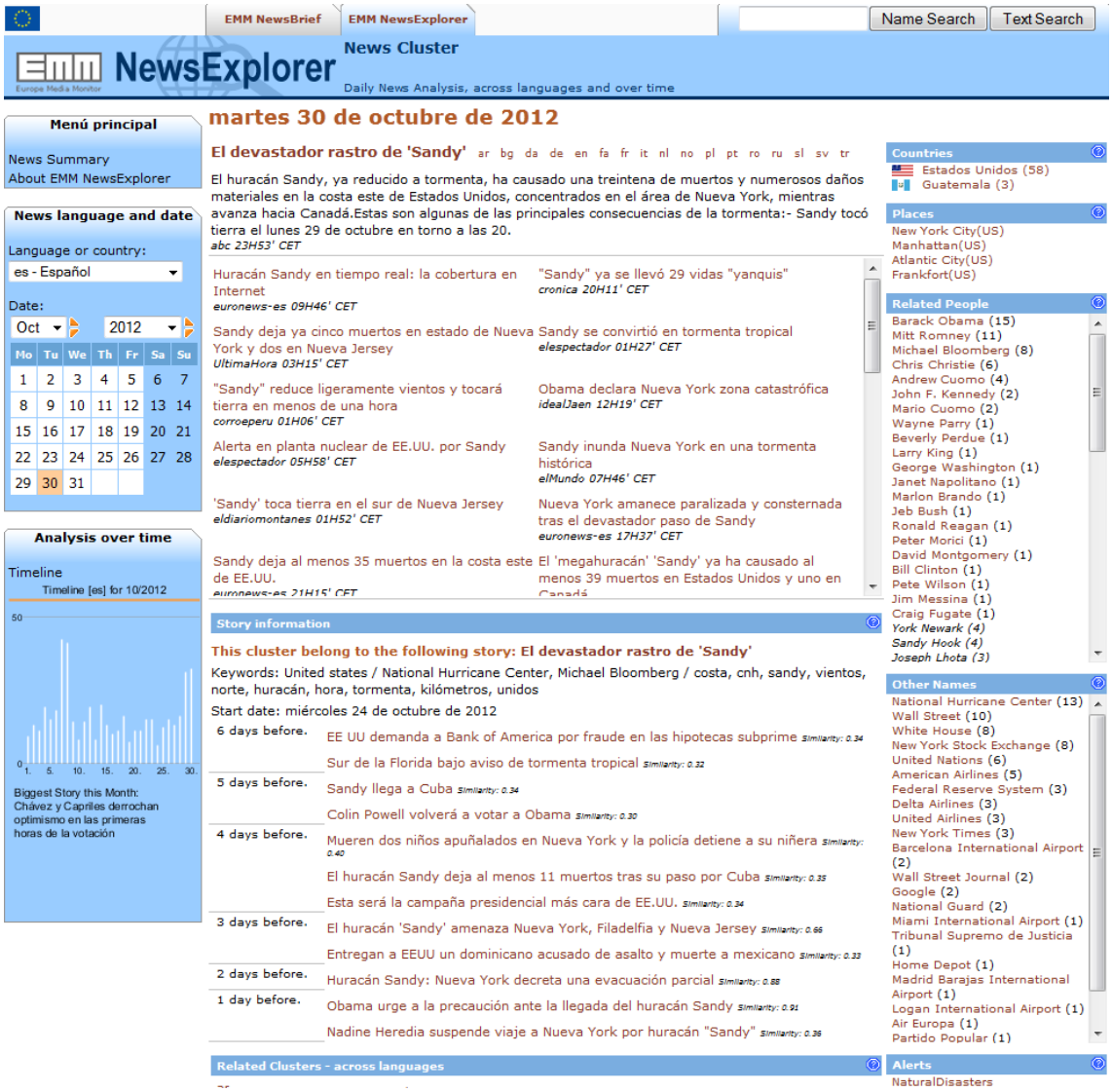


Figura 3: Página de clúster de News Explorer

2 – Europe Media Monitor

En este capítulo desgranamos en detalle el sistema de agrupación de noticias EMM exponiendo su funcionamiento interno y explicando las herramientas relacionadas con nuestro proyecto.

Como ya hemos dicho en el capítulo 1, EMM News Explorer recopila noticias de periódicos de diferentes países en varios idiomas y las agrupa en clústers, mostrando los países, las personas y las entidades referenciadas en cada noticia. Esta recopilación es posible gracias a una herramienta llamada Web Crawler, que automatiza la búsqueda de determinados contenidos a través de la Web.

2.1 –Web Crawler

Un Web Crawler es un programa informático que automáticamente analiza la Web en busca de contenido diverso, desde texto plano a archivos multimedia. También recibe los nombres de Screen Scraper, Bot y Spider. Generalmente funciona partiendo de una o varias direcciones Web, a las que accede y recopila los hiperenlaces que encuentra para continuar la búsqueda a través de ellos.

Sus usos más frecuentes son:

- Almacenar el contenido de las páginas visitadas para que un motor de búsqueda las analice e indexe de cara a proporcionar una búsqueda más rápida.
- Automatizar el mantenimiento de un sitio web analizando periódicamente enlaces rotos o validando el código HTML.
- Recopilar direcciones de correo electrónico para enviar SPAM.

Debido a la sobrecarga que producen en la red, es aconsejable tener una política respetuosa con los recursos de los servidores. Una solución sencilla es intercalar temporizadores de unos pocos segundos entre accesos a un mismo servidor para permitirle que sirva peticiones de otros usuarios.

Los Web Crawlers se identifican ante el servidor mediante la cabecera User-Agent del protocolo HTTP. Este campo suele contener una breve descripción del origen y propósito del robot. Adicionalmente también puede incluir una dirección Web donde los administradores del servidor pueden encontrar información de contacto del desarrollador. Es habitual que los administradores utilicen esta información para alertar al desarrollador de un mal uso de sus recursos. Algunos de los Crawlers más conocidos son Googlebot[8] y Yahoo! Slurp[9].

[8] http://es.wikipedia.org/wiki/Googlebot

2.2 – El multilingüismo en News Explorer

En el primer capítulo hemos visto que News Explorer recopila y organiza noticias en muchos idiomas y las relaciona entre sí. Esta relación de noticias no es casual, y responde a la necesidad de las grandes organizaciones de realizar un seguimiento de las noticias en varios idiomas.

Por una parte, cada área del mundo está periodísticamente mejor cubierta en su lengua principal. Por ejemplo las noticias sobre sucesos acontecidos en Brasil suelen ser más detalladas en portugués que en otros idiomas. Por otro lado, la información obtenida en diferentes idiomas es a menudo complementaria

Para resumir las técnicas que utilizan para conseguir relacionar noticias en tantos idiomas, diremos que en general utilizan reglas independientes del idioma, lo que hace posible ampliar el sistema con nuevas lenguas.

2.3 – Reconocimiento de referencias geográficas (geo-tagging)

Antes también hemos dicho que News Explorer tiene constancia de los países y lugares mencionados en cada noticia. Esto es posible gracias a la técnica de geo-tagging, que consiste en reconocer en un texto referencias a ubicaciones geográficas e identificar unívocamente sus rangos de latitud y longitud.

Para ello News Explorer utiliza varios diccionarios geográficos o *gazetteers* que contienen diferentes listas de lugares existentes asociados a su latitud y longitud.

Existen varios problemas asociados a esta técnica. Es el caso de varios lugares que tienen el mismo nombre (London, Canadá y London, Gran Bretaña). También se puede dar que una persona se llame igual que un país (Israel). Además un mismo lugar puede tener diferentes nombres.

Para superar estos obstáculos, utilizan una serie de normas para asegurarse de que los lugares se reconocen correctamente:

- Para nombres conocidos de personas u organizaciones, preferir interpretaciones de nombre sobre interpretaciones de lugar.
- Hacer uso de la información sobre la importancia de un lugar y su tamaño.
- Utilizar la información de contexto del país del que se habla en el artículo.
- Preferir ubicaciones que estén cerca de otros lugares no ambiguos mencionados en el texto.
- Ignorar lugares que son muy difíciles de desambiguar.

---

[9] http://en.wikipedia.org/wiki/Yahoo!_Slurp

2.4 – Identificación de nombres conocidos y sus variantes

Del mismo modo que sucede con los países, también existen diferentes variantes de una misma entidad sea ésta una persona o una organización. Ello requiere prestar especial atención a las siguientes situaciones:

- Se utilizan indistintamente el guión y el espacio en nombres compuestos.
- Variaciones diacríticas.
- Abreviaciones.
- A veces el nombre viene precedido por el apellido en lugar de al revés.
- Errores tipográficos.
- Transliteración: Al traducir un nombre originario de un país a otras lenguas, se producen pequeñas diferencias en la escritura del mismo relacionadas con su pronunciación en cada idioma.
- Variaciones de vocales: En especial al traducir desde y hacia el árabe. Puesto que sus símbolos sólo contienen consonantes, existe una gran variedad de formas de escribirlos.

Una vez más los procedimientos empleados son independientes del lenguaje. Dos de los procedimientos utilizados para atajar estas variantes son la transliteración y la normalización. El primero se utiliza en lenguas con símbolos diferentes en sus alfabetos, como el cirílico o el árabe y consiste en sustituir símbolos de otros idiomas en letras o conjuntos de letras de nuestro alfabeto.

En el caso de la normalización lo más común sería emplear una representación de su pronunciación, pero como se debe conocer el origen un nombre para determinar su pronunciación, esta práctica resulta inviable. En su lugar se emplean reglas obtenidas a partir de observar las diferencias entre pronunciaciones de un mismo nombre en diferentes idiomas, desarrollando así reglas de normalización aplicables a cualquier nombre independientemente de su origen.

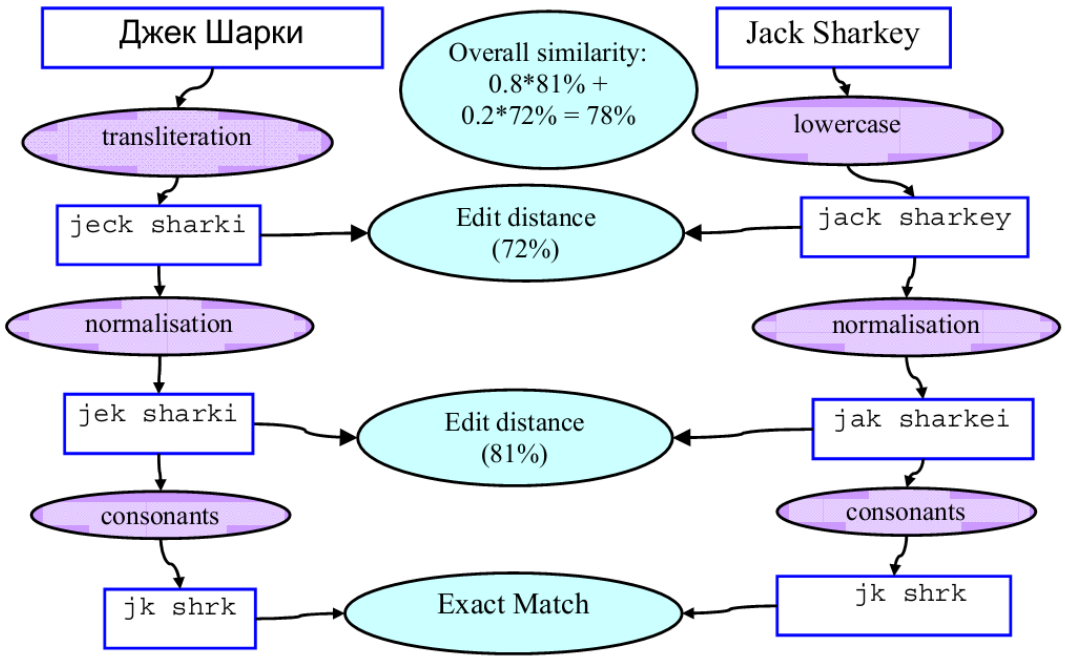


Figura 4: Proceso de normalización de nombres

Como último paso se eliminan todas las vocales del nombre normalizado para obtener una expresión idéntica de todas las variantes de un mismo nombre. El resultado se puede ver en la Figura 4.

Podemos ver un ejemplo visitando la página de persona de News Explorer. A la izquierda de la Figura 5 vemos los diferentes nombres que se le atribuyen a Barack Obama en diferentes idiomas.

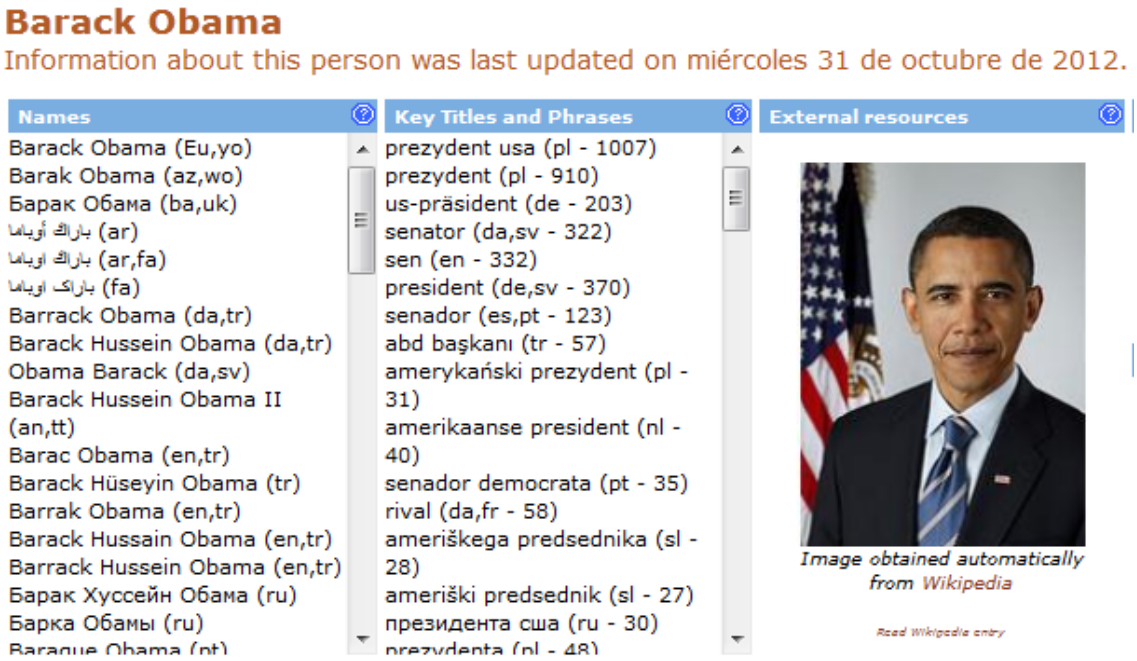


Figura 5: Página de Persona en News Explorer

2.5 – Catalogación de documentos mediante Eurovoc

Otra característica del EMM mencionada anteriormente es la categorización de documentos mediante el diccionario de sinónimos multilingüe Eurovoc[10].

Eurovoc es un vocabulario organizado jerárquicamente creado por el Parlamento Europeo y la Oficina de Publicación de la Comisión Europea junto con varios parlamentos nacionales de dentro y fuera de la Unión Europea. Su objetivo principal es el de catalogación, búsqueda y obtención de su amplia colección de documentos multilingüe.

A diferencia de los clasificadores habituales, el EMM no clasifica cada documento en una única sección. En cambio, genera una larga lista de temas más o menos relevantes por cada documento. Esta representación resulta muy beneficiosa a la hora de calcular la semejanza entre documentos.

2.6 – Enlazando clústers en el tiempo y entre idiomas

Como ya hemos dicho, News Explorer relaciona los clústers de noticias similares en varios idiomas. Para ello, cada día calcula la semejanza de los clústers del día con todos

[10] http://eurovoc.europa.eu/

los clústers (en el mismo idioma) de los últimos 7 días. Si la semejanza entre dos clústers es mayor al 50 %, se enlazan los clústers en una historia.

Además las historias se van actualizando a lo largo del tiempo con: El título del primer clúster de la historia, el título del clúster más grande de la misma, los nombres de personas, países y organizaciones más frecuentes de la historia y una lista de keywords o palabras clave que la representan.

Del mismo modo, cada día se calcula la semejanza de los clústers de nueve de los idiomas (holandés, inglés, francés, alemán, italiano, portugués, esloveno, español y sueco) con los otros 18.

Por último se calcula una nueva semejanza realizando la siguiente comprobación: Si un clúster A está enlazado con los clústers B y C, entonces B debería estar enlazado con C. Y si no lo está, entonces es menos probable que B esté relacionado con A.

Este algoritmo comprueba esos enlaces internos y calcula la nueva semejanza combinando la calculada anteriormente con el número de inter-enlaces. Esta fórmula penaliza los enlaces que conducen a clústers aislados y bonifica la puntuación de los enlaces que apuntan a clústers que a su vez enlazan a otros clústers.

2.7 – RSS

Como vemos en la Figura 6, la página principal de News Explorer nos permite acceder a los últimos clústers de noticias mediante un canal RSS. En dicho canal encontramos una lista de enlaces a las páginas de los clústers con más noticias del día junto con su título y descripción.

Figura 6: Enlace al canal RSS de News Explorer

Las siglas RSS han ido cambiando de significado a lo largo de sus diferentes versiones. Lo mismo ha sucedido con su estructura, que intermitentemente ha respondido a los estándares XML y RDF. Así, en la versión 0.9 y 1.0 significaba RDF Site Summary, en la 0.91 significaba Rich Site Summary y en la actual versión 2.0 significa Really Simple Syndication.

No obstante todas las versiones tienen el mismo objetivo: ofrecer actualizaciones periódicas de un sitio web sin necesidad de que los usuarios accedan al contenido html completo de la página. Los usuarios se suscriben al RSS mediante una aplicación denominada *agregador de noticias* y reciben los titulares de las noticias.

3 – Documento de Objetivos del Proyecto

3.1 – Objetivo

El objetivo de este proyecto es el de desarrollar un programa que recopile periódicamente el contenido de las noticias del portal News Explorer (noticias, clústers, lugares, países, personas, etc.) y las almacene en disco para su posterior análisis.

Este programa además deberá guardar la relación entre clústers y deberá mantener un registro de los datos añadidos y de la fecha de ejecución del mismo.

3.2 – Método de trabajo

Se utilizará el Proceso Unificado de Desarrollo (PUD). Este método de trabajo destaca por estar dirigido por casos de uso y orientado a la arquitectura y por su desarrollo iterativo e incremental.

Se realizarán tantas iteraciones como sean necesarias. En cada reunión se establecerán los objetivos a cumplir en dicha iteración y se verificará el trabajo realizado anteriormente.

En principio se prevén seis iteraciones, cada una determinada por un prototipo:

1. Realizar un programa que lea el contenido de un RSS.
2. Actualizar el programa para que extraiga de dicho RSS los datos necesarios para acceder a cada clúster (título, enlace).
3. Extraer el contenido de cada clúster (título, enlace, noticias, países, personas, lugares, clústers relacionados, etc.) de forma que se pueda almacenar en disco.
4. Diseñar el modelo de datos (XML o BD Relacional) siguiendo la estructura que guardan los elementos en el portal News Explorer.
5. Modificar el programa para que almacene los datos extraídos en la 3ª iteración en la base de datos.
6. Mostrar un registro log de los datos extraídos y automatizar el programa para que se ejecute periódicamente.

3.2.1 – Procesos Formativos

Principalmente se realizarán al inicio del proyecto para que el alumno tenga una buena base de los lenguajes de programación y las herramientas y técnicas utilizadas. No obstante a lo largo del proyecto se consultarán las dudas que surjan tanto en libros de referencia y en internet, como con el profesor.

3.2.2 – Procesos Tácticos

Durante el transcurso del proyecto se realizarán reuniones entre el alumno y el director para analizar y evaluar el progreso del mismo y, en caso de que sea necesario, cambiar de rumbo. También forma parte de los procesos tácticos la planificación del proyecto, reflejada en el Documento de Objetivos del Proyecto.

3.2.3 – Procesos Operativos

Unos de los procesos operativos más importantes son las reuniones. En las que, además de analizar la situación, también se establecerán unos objetivos claros y específicos cada uno con su plazo de entrega establecido. La realización de cada objetivo conlleva su propia captura de requisitos, análisis, diseño, implementación y pruebas.

3.2.4 – Herramientas que se van a utilizar

El alumno utilizará su propio ordenador para desarrollar el proyecto y aunque se haya trabajado casi todo el tiempo sobre Windows 7, para las pruebas hemos utilizado Ubuntu 11.10. La razón para utilizar Windows es que el alumno está más habituado a utilizar ciertas aplicaciones.

Para programar en Perl e implementar la base de datos en SQL se utilizará el programa Notepad ++. Del mismo modo, se empleará WampServer para instalar la base de datos y realizar las pruebas de manera visual. Se ha elegido la distribución Strawberry Perl en lugar de ActivePerl porque la primera utiliza CPAN[11] para descargar librerías, igual que en Linux.

Para las copias de seguridad el alumno utilizará el software Cobian Backup junto con Dropbox, que aunque sea una solución algo arriesgada es suficiente para este proyecto.

Con respecto a la creación de la memoria, se utilizará el editor de textos Microsoft Word 2007, el programa Dia para la creación de diagramas UML, Gantt Project para editar el diagrama de planificación de Gantt y la aplicación de Dibujo de Google Docs para crear el resto de diagramas.

Notepad++

Notepad++[12] es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. Solo funciona en Microsoft Windows.

[11] http://search.cpan.org/
[12] http://notepad-plus-plus.org/

Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores. Algunas de ellas son:

- Coloreado y envoltura de sintaxis.
- Pestañas.
- Resaltado de paréntesis e indentación.
- Grabación y reproducción de macros.
- Soporte de extensiones.

Elegimos este programa porque el alumno lo había utilizado anteriormente para desarrollar programas en otros lenguajes de programación y es de los editores más ligeros para Windows.

Cobian Backup

Cobian Backup[13] es un programa multitarea capaz de crear copias de seguridad en un equipo, en una red local o incluso en/desde un servidor FTP. También soporta SSL. Se ejecuta sobre Windows y uno de sus grandes fuertes es que consume muy pocos recursos y puede estar funcionando en segundo plano.
Cada tarea de respaldo que le asignemos puede ejecutarse en el momento, diaria, semanal, mensual o anualmente, o en un tiempo especificado. Hace copias completas, incrementales y diferenciales.

Dropbox

Dropbox[14] es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre computadoras y compartir archivos y carpetas con otros. La sincronización de Dropbox usa transferencias SSL y almacena los datos mediante el protocolo de cifrado AES-256. Si bien Dropbox funciona como un servicio de almacenamiento, se enfoca en sincronizar y compartir archivos.

Dia

Dia[15] es una aplicación informática de propósito general para la creación de diagramas, desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades.

[13] http://www.cobiansoft.com/index.htm
[14] https://www.dropbox.com/home
[15] https://live.gnome.org/Dia

Dia está diseñado como un sustituto de la aplicación comercial *Visio* deMicrosoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML.

El formato para leer y almacenar gráficos es XML (comprimido con gzip, para ahorrar espacio). Puede producir salida en los formatos EPS, SVG y PNG.

GanttProject

GanttProject[16] es una herramienta de escritorio multiplataforma para la programación y gestión de proyectos. Se ejecuta en Windows, Linux y MacOSX, es libre y su código es opensource. Permite crear el diagrama de Gantt y guardarlo como imagen PNG.

3.3 – Alcance

3.3.1 – Recursos humanos

Los responsables de este proyecto son el alumno y el profesor asignado como director del proyecto. El alumno tiene la responsabilidad de realizar las tareas concretadas en el tiempo acordado mientras que el director evalúa el progreso del proyecto y orienta al alumno hacia la consecución de los objetivos marcados.

3.3.2 – Recursos materiales

El alumno dispone de un ordenador portátil, en el que desarrollará la totalidad del proyecto, así como el software necesario para ello. Además dispondrá del material de oficina necesario para realizar borradores. El alumno tiene acceso a la biblioteca de la UPV/EHU donde encontrará la mayor parte de la bibliografía del proyecto. También dispone de acceso a Internet en todo momento para comunicarse con el director mediante correo electrónico y para buscar información que no haya encontrado en la biblioteca.

Del mismo modo, el director dispone de un despacho en el que se realizarán las reuniones, un ordenador con conexión a Internet para comunicarse con el alumno en horario de la facultad de Informática y tiene acceso a un servidor en el que se instalará la aplicación desarrollada por el alumno y donde se ejecutará periódicamente una vez terminada.

[16] http://www.ganttproject.biz/

3.3.3 – Estructura de Descomposición de Trabajo (EDT)

En la Figura 7 se puede ver el diagrama EDT con las tareas que se realizarán a lo largo del proyecto. Aunque es algo genérico, en esta fase aún no conocemos todas las herramientas y las técnicas de programación específicas que utilizaremos para desarrollar la aplicación.

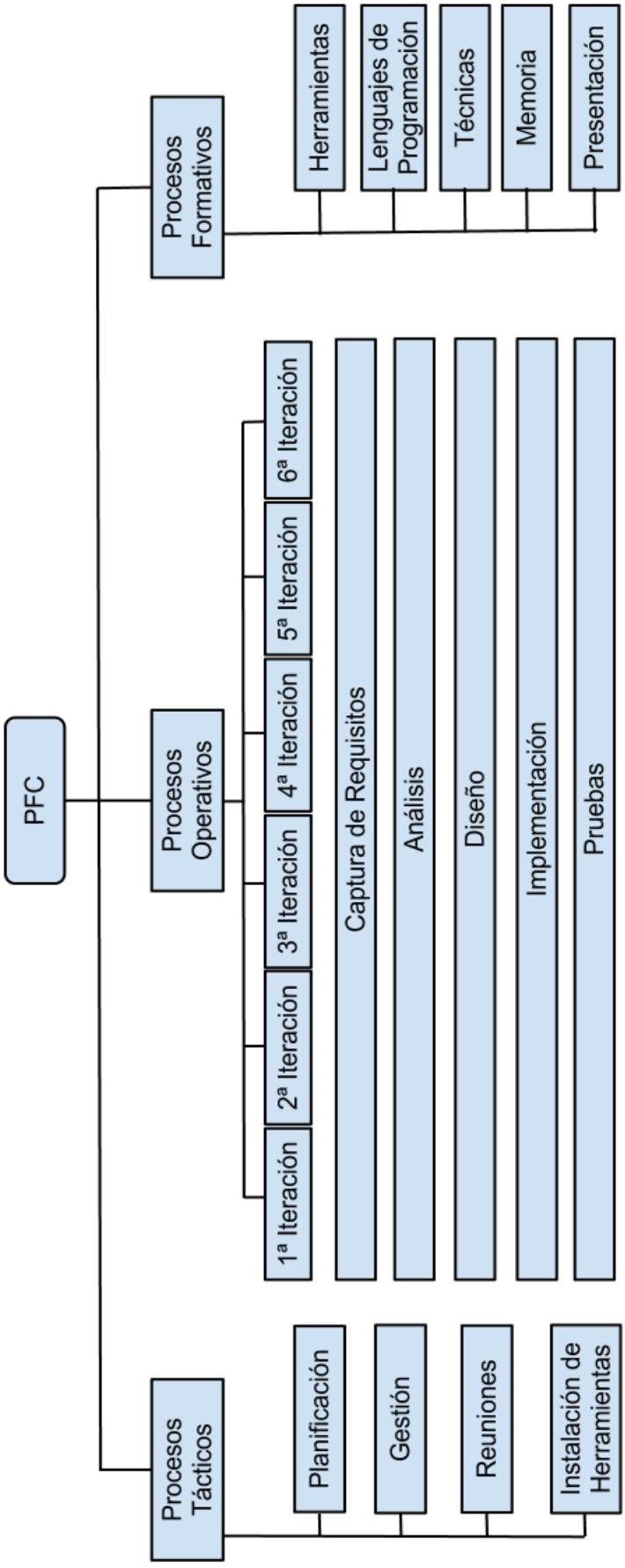


Figura 7: Diagrama de EDT

3.3.4 – Proyecto de Fin de Carrera (PFC)

Procesos Tácticos

Planificación
Descripción: La planificación del proyecto y la creación del DOP.
Tiempo estimado: 20h

Gestión
Descripción: Realización de las tareas de gestión del proyecto.
Tiempo estimado: 10h

Reuniones
Descripción: Reuniones en las que se realiza un análisis de situación y se concretan nuevos objetivos entre alumno y profesor con fecha de entrega determinada.
Tiempo estimado: 15h

Instalación de Herramientas
Descripción: Puesta a punto del ordenador de trabajo del alumno para empezar a desarrollar (Perl, Módulos Perl, Unix,…).
Tiempo estimado: 5h

Procesos Formativos

Herramientas
Descripción: Aprender a utilizar los entornos de desarrollo, sistemas operativos, y sobre todo entender la organización del portal de noticias News Explorer.
Tiempo estimado: 15h

Lenguajes de Programación
Descripción: Adquirir una base en los lenguajes de programación utilizados (Perl).
Tiempo estimado: 10h

Técnicas
Descripción: Aprender y comprender las técnicas de programación independientes del lenguaje a implementar (Expresiones Regulares, ...).
Tiempo estimado: 5h

Memoria
Descripción: Realización de la memoria del proyecto.
Tiempo estimado: 80h

Presentación
Descripción: La creación de las transparencias de la presentación y la preparación de la defensa.
Tiempo estimado: 5h

Procesos Operativos

Primera Iteración:

Descripción: Realizar un programa que lea el contenido de un RSS.

Implementación
Tiempo estimado: 1h

Pruebas
Tiempo estimado: 1h

Segunda Iteración:

Descripción: Actualizar el programa para que extraiga de dicho RSS los datos necesarios para acceder a cada clúster (título, enlace).

Implementación
Tiempo estimado: 1h

Pruebas
Tiempo estimado: 1h

Tercera Iteración:

Descripción: Extraer el contenido de cada clúster (título, enlace, noticias, países, lugares, clústers relacionados, etc.) de forma que se pueda almacenar en disco.

Captura de Requisitos
Descripción: En esta etapa es necesario comprender la estructura y el funcionamiento del News Explorer para realizar el modelo de dominio.
Tiempo estimado: 10h

Análisis
Descripción: Al haber tantos elementos diferentes a extraer, la mejor opción es obtenerlos por separado mediante funciones o procedimientos bien diferenciados.
Tiempo estimado: 5h

Diseño
Descripción: La parte de diseño de esta iteración implica realizar un pseudocódigo de las operaciones que se van a implementar dependiendo del método de extracción de información elegido.
Tiempo estimado: 5h

Implementación
Descripción: Implementar todas las operaciones lleva su tiempo. A menudo surgen errores y hay que reescribir parte del código. Por suerte al haber separado las operaciones resulta más sencillo localizar el fallo.
Tiempo estimado: 15h

Pruebas
Descripción: Este es el apartado que más programación requiere y por lo tanto en el que más fallos vamos a encontrar. Antes de continuar es preciso realizar todas las pruebas que sean necesarias.
Tiempo estimado: 5h

Cuarta Iteración:

Descripción: Diseñar el modelo de datos siguiendo la estructura que guardan los elementos en el portal News Explorer.

Captura de Requisitos
Descripción: Es necesario repasar el modelo de dominio junto con los datos extraídos.
Tiempo estimado: 1h

Diseño
Descripción: En este apartado debemos realizar el diseño de la base de datos a partir del modelo de dominio y la estructura del News Explorer.
Tiempo estimado: 2h

Implementación
Descripción: Hay que pasar el diseño de la base de datos al lenguaje de consultas SQL.
Tiempo estimado: 2h

Pruebas
Descripción: Las pruebas a realizar en este apartado guardan relación con el motor de bases de datos y con las BD en un sistema operativo Linux.
Tiempo estimado: 1h

Quinta Iteración:

Descripción: Modificar el programa para que almacene en la base de datos los datos extraídos en el tercer prototipo.

Captura de Requisitos
Descripción: Son necesarias nuevas operaciones para almacenar los datos en la BD.
Tiempo estimado: 1h

Análisis
Descripción: Realizar los contratos de las operaciones a implementar.
Tiempo estimado: 2h

Diseño
Descripción: En este apartado debemos ajustar el tamaño de los campos de la base de datos según los datos extraídos. Además realizaremos el pseudocódigo de las operaciones a implementar. En la mayoría de los casos esto supondrá modificar los datos obtenidos con el formato en el que los queremos almacenar.
Tiempo estimado: 5h

Implementación
Descripción: Desarrollar dichas operaciones por separado e incluirlas en el programa principal.
Tiempo estimado: 5h

Pruebas
Descripción: Hay que comprobar el tamaño de los campos de la base de datos, los enlaces relativos, la conexión con la base de datos y los campos nulos.
Tiempo estimado: 5h

Sexta Iteración:

Descripción: Almacenar un registro log de los datos extraídos y automatizar el programa para que se ejecute periódicamente.

Captura de Requisitos
Descripción: Identificar las operaciones necesarias para guardar el registro y programar la ejecución de la aplicación.
Tiempo estimado: 1h

Diseño
Descripción: Realizar un boceto de cómo se quiere implementar el código.
Tiempo estimado: 1h

Implementación
Descripción: Implementar las operaciones que permitan llevar a cabo el objetivo de la iteración.
Tiempo estimado: 4h

Pruebas
Descripción: Además de comprobar que todo funciona como se pretende, será necesario probar el programa durante una semana para detectar errores de última hora.
Tiempo estimado: 6h

Estimamos que el tiempo total invertido en el proyecto será de unas 245 horas. Donde dedicaremos 50 horas a los procesos tácticos, 115 a los procesos formativos y 80 a los procesos operativos.

3.4 – Planificación Temporal: Diagrama de Gantt

En la Figura 8 se muestra el diagrama de planificación temporal, en el que se puede ver la duración estimada de cada tarea. Se prevé finalizar el proyecto hacia finales de Diciembre por lo que la planificación se ha realizado con arreglo a esto.

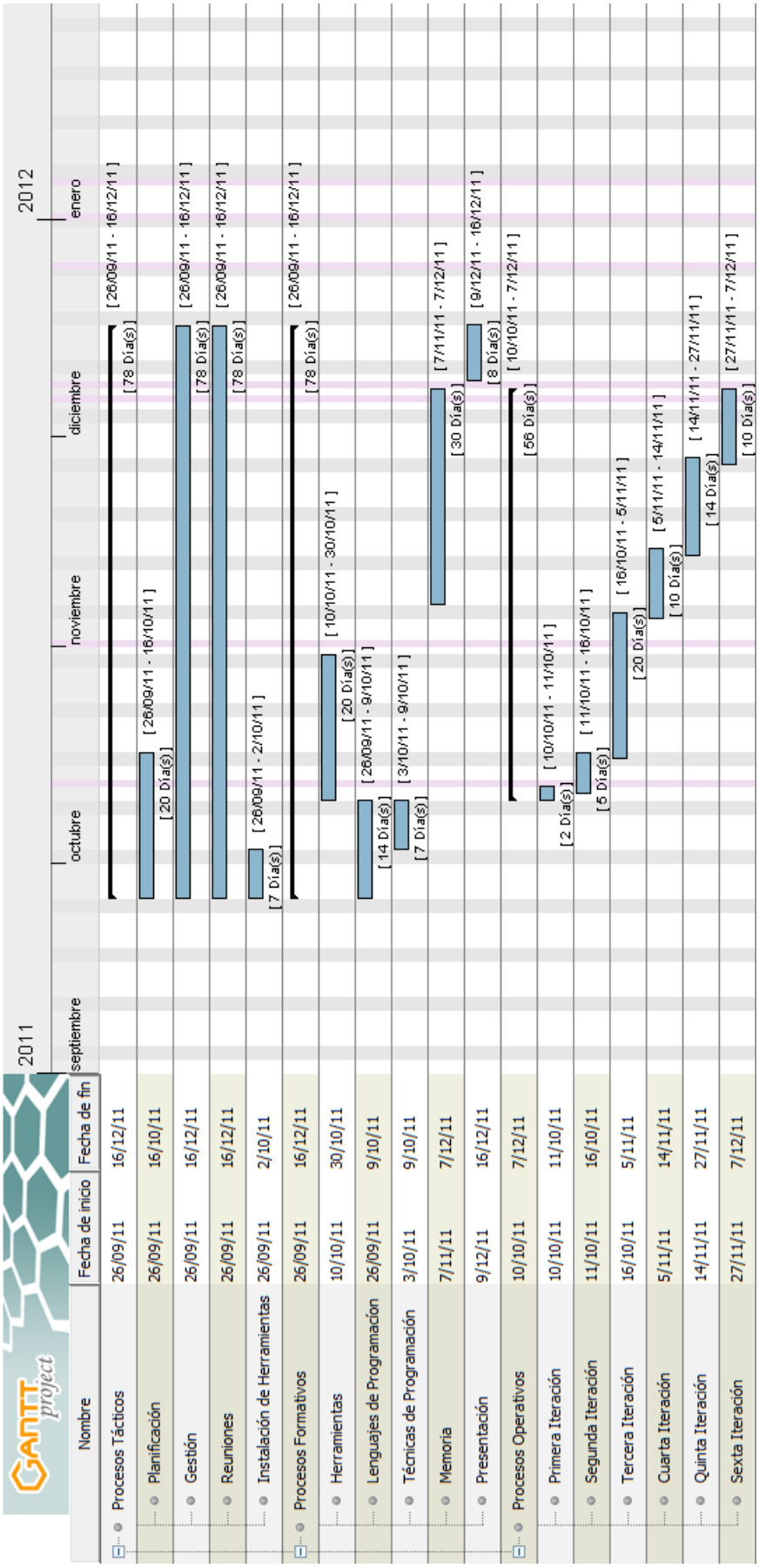


Figura 8: Diagrama de Gantt

3.5 – Plan de Contingencia

En todo proyecto es primordial conocer los riesgos y disponer de una serie de medidas para paliar las dificultades que se puedan hallar a lo largo de su desarrollo.

A continuación se muestran dichos problemas en la tabla de riesgos, seguidamente se describirán y se organizarán en tres grupos: riesgos formativos, riesgos tácticos y riesgos operativos.

3.5.1 – Tabla de riesgos

En la Tabla 1 se muestran los riesgos acompañados por la probabilidad de que sucedan y la gravedad en caso de producirse.

Riesgo	Probabilidad	Gravedad
Problemas en el aprendizaje	Media	Baja
Pérdida de documentación	Baja	Baja
Indisponibilidad para reunir	Baja	Muy baja
Retraso en la tarea	Media	Media
Ordenador estropeado	Muy baja	Media
Pérdida de datos	Baja	Media
Problemas en manejar software	Media	Baja

Tabla 1: Tabla de Riesgos

3.5.2 – Riesgos formativos

Problemas en el aprendizaje

Riesgo: El alumno tiene problemas para entender algo relacionado con el proyecto y no lo ha solucionado visitando la biblioteca ni buscando en Internet. Si no se soluciona pronto puede dar lugar a que no se realicen las tareas a tiempo.
Solución: Hablar con el director del proyecto y pedirle más fuentes de información o en su caso la explicación necesaria para seguir adelante.

Pérdida de documentación

Riesgo: El alumno pierde la documentación relacionada con la parte formativa del proyecto o el acceso a la documentación.
Solución: Buscar en Internet, si no se encuentra el material deseado el alumno solicitará al director del proyecto material bibliográfico que pueda serle de ayuda.

3.5.3 – Riesgos tácticos

Indisponibilidad para reunir

Riesgo: El alumno o el director del proyecto se encuentra indispuesto o reunido y le es imposible asistir a una reunión concertada.
Solución: La persona que no se encuentre disponible enviará un correo electrónico a la otra a fin de informarle de la situación y se acordará otra fecha que satisfaga a las dos partes. Si es posible solucionar el motivo de la reunión por correo se procederá a ello.

Retraso en la tarea

Riesgo: El alumno no ha calculado bien el tiempo que le va a llevar terminar una tarea y a mitad de ella o cerca de la fecha de entrega se da cuenta de ello.
Solución: El alumno contactará con el director del proyecto para informarle de que necesita más tiempo para realizar la tarea y de los problemas que está experimentando que le han inducido a prolongar la duración de la misma. Ambos establecerán una nueva fecha de entrega y se guardará un registro de problemas y cambios en la planificación.

Ordenador estropeado

Riesgo: El ordenador del alumno se avería y no se puede utilizar para continuar el desarrollo del proyecto.
Solución: Se recuperará una copia de seguridad del proyecto y se empezará a trabajar en otro ordenador. Si no queda más remedio y como última opción se trabajará en los ordenadores de la Facultad de Informática teniendo especial cuidado de realizar copias de seguridad cada vez y de eliminar del disco toda la información del proyecto.

Pérdida de datos

Riesgo: El alumno pierde los ficheros relacionados con la gestión del proyecto.
Solución: Se recuperará una copia de seguridad del proyecto que contenga los datos perdidos.

3.5.4 – Riesgos Operativos

Pérdida de datos

Riesgo: El alumno pierde algún fichero relacionado con el desarrollo del proyecto, como el diseño de la base de datos o la implementación de alguna función.

Solución: Se recuperará la copia de seguridad del proyecto que contenga los datos perdidos.

Problemas en manejar software

Riesgo: El alumno tiene dificultades para manejar el software utilizado en el proyecto.
Solución: El alumno realizará una búsqueda rigurosa por Internet con el fin de encontrar manuales, foros de ayuda, preguntas frecuentes, etcétera que aclaren sus dudas. Si esto falla consultará al director del proyecto.

3.6 – Análisis de factibilidad

Dado que conocemos las tareas que habremos de realizar y conocemos los riesgos de las mismas, y como hemos elaborado un plan de contingencia y un cálculo del tiempo estimado necesario para completar cada una de ellas (que en total suman 245 horas), podemos concluir que con el esfuerzo y dedicación adecuados seremos capaces de terminar el proyecto en el tiempo establecido.

4 – Arquitectura del sistema y elección tecnológica

4.1 – Arquitectura del sistema

La arquitectura de nuestro sistema se compondrá de un servidor continuamente activo en el que se ejecutará nuestro programa una vez al día. Nuestro programa tendrá acceso al servidor de Europe Media Monitor y a su RSS así como a una base de datos alojada en nuestro servidor en la que se almacenarán los datos recopilados del News Explorer.

En la Figura 9 se muestra un diagrama que representa la arquitectura del sistema.

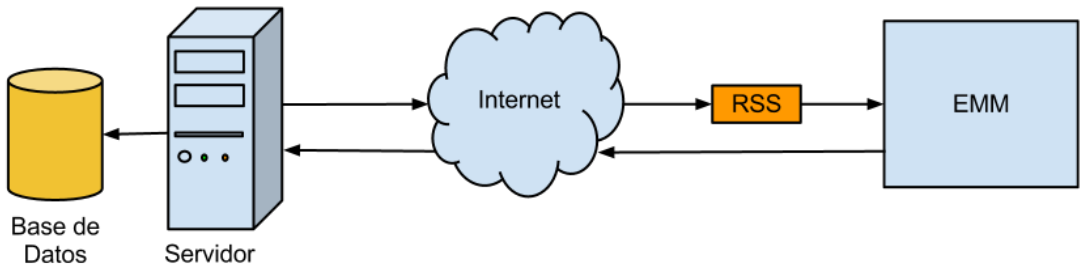


Figura 9: Diagrama explicativo de la arquitectura del sistema.

4.2 – Elección tecnológica

En este apartado detallamos el software elegido para nuestro sistema y el software utilizado para desarrollar el proyecto.

Nuestro sistema estará desarrollado en Perl[17] y se ejecutará sobre un servidor con Sistema Operativo Linux[18], más concretamente Fedora[19]. La ejecución automatizada se llevará a cabo mediante el programa cron. Utilizaremos MySQL como sistema gestor de bases de datos y MySQL Server como servidor de Bases de Datos.

[17] http://www.perl.org/
[18] http://www.linux-es.org/
[19] http://fedoraproject.org/es/

Fedora

Fedora es una distribución Linux para propósitos generales basada en RPM, que se caracteriza por ser un sistema estable, la cual es mantenida gracias a una comunidad internacional de ingenieros, diseñadores gráficos y usuarios que informan de fallos y prueban nuevas tecnologías. Cuenta con el respaldo y la promoción de Red Hat.

El proyecto no busca sólo incluir software libre y de código abierto, sino ser el líder en ese ámbito tecnológico. Algo que hay que destacar es que los desarrolladores de Fedora prefieren hacer cambios en las fuentes originales en lugar de aplicar los parches específicos en su distribución, de esta forma se asegura que las actualizaciones estén disponibles para todas las variantes de GNU/Linux.

La elección de Fedora como sistema operativo viene dada por ser el sistema presente en el servidor al que tenemos acceso.

Perl

Perl es un lenguaje de programación diseñado por Larry Wall en 1987. Perl toma características del lenguaje C, del lenguaje interpretado bourne shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación.

Estructuralmente, Perl está basado en un estilo de bloques como los del C o AWK, y fue ampliamente adoptado por su destreza en el procesado de texto y no tener ninguna de las limitaciones de los otros lenguajes de script.

Hemos elegido Perl como lenguaje de desarrollo de nuestra aplicación porque sabíamos que íbamos a tener que tratar con una gran cantidad de texto (código HTML). Otra razón es que ya se había utilizado en otros proyectos similares.

Además, Perl dispone de varios módulos o librerías preparadas para simular el comportamiento de un navegador web y avanzar a través de los enlaces de una página web.

Cron

En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab. Cron se podría definir como el "equivalente" a Tareas Programadas de Windows.

Utilizaremos este programa porque viene con el propio sistema operativo en el que vamos a instalar la aplicación.

MySQL

MySQL[20] es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Guardaremos los datos recogidos del News Explorer en una base de datos MySQL instalada en un servidor de la Facultad de Informática de San Sebastián. Una de las ventajas que tienen estas bases de datos es que se pueden consultar de una manera más legible que por consola de comandos mediante el programa phpMyAdmin[21], una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web.

WampServer

WAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- **W**indows, como sistema operativo.
- **A**pache, como servidor web.
- **M**ySQL, como gestor de bases de datos.
- **P**HP (generalmente), **P**erl, o **P**ython, como lenguajes de programación.

WampServer es una implementación de servidor WAMP que, como la mayoría de ellos, incluye PHPMyAdmin para gestionar de manera sencilla las bases de datos.

Como en otras ocasiones, el alumno ya conocía esta herramienta por lo que la eligió con el fin de centrar su atención en el aprendizaje de Perl.

CPAN

CPAN (Comprehensive Perl Archive Network) es una colección de sitios web que almacenan y distribuyen fuentes en Perl, binarios, documentación, scripts y módulos. Actualmente hay más de 100.000 módulos empaquetados en más de 23.000 paquetes,

[20] http://www.mysql.com/
[21] http://www.phpmyadmin.net/

aportados por más de 9.000 autores. Se dispone de módulos para una amplia variedad de tareas, incluyendo matemáticas avanzadas, conectividad de bases de datos y conexión de redes.
Los módulos en CPAN pueden ser descargados e instalados a mano. Sin embargo, es muy común que los módulos dependan de otros módulos y seguir las dependencias a mano puede ser tedioso. Tanto el módulo CPAN.pm (incluido en la distribución Perl) como el módulo mejorado CPANPLUS ofrecen instaladores en línea de comandos que entienden las dependencias; pueden configurarse para descargar automáticamente e instalar un módulo y, recursivamente, todos los módulos de los que dependa.

Como hemos dicho anteriormente se ha elegido Strawberry Perl porque queremos que, a pesar de trabajar con Windows, la forma de trabajo sea lo más parecida posible al sistema en el que se instalará nuestra aplicación.

5 – Captura de Requisitos

El objetivo de nuestra aplicación es acceder al canal RSS de EMM News Explorer, donde obtendrá los enlaces que conducen a cada nuevo clúster de noticias. En la Figura 10 se aprecia parte del contenido del canal RSS a 10 de Julio de 2012.


```
▼<rss xmlns:emm="http://emm.jrc.it" xmlns:iso="http://www.iso.org/3166" xmlns:georss="http://www.georss.org/georss"
 version="2.0">
  ▼<channel>
    ▼<description>
       Europe Media Monitor (EMM) reads and analyses around 40.000 new news items per day. One of the results of
       this analysis is the clusters of the main stories in several languages. The selection and placement of
       stories are determined automatically by a computer program. This site is a joint project of DG-JRC and DG-
       COMM. The information on this site is subject to a disclaimer (see
       http://europa.eu/geninfo/legal_notices_en.htm). Please acknowledge EMM when (re)using this material.
     </description>
    ▼<link>
       http://emm.newsexplorer.eu/NewsExplorer/dayedition/es/latest.html
     </link>
     <lastBuildDate>Tue, 10 Jul 2012 15:40:40 GMT</lastBuildDate>
     <title>EMM Clusters: Clustered news stories - Español</title>
    ▼<item>
      ▼<title>
         El bono español supera el 7% en espera del Eurogrupo
       </title>
      ▼<description>
         La prima de riesgo española sigue subiendo sin tregua desde el inicio de la sesión y se encuentra ya en 580
         puntos básicos, después de que el rendimiento del bono nacional a diez años se situara en el 7,1%. Así, la
         prima de riesgo española se acerca al máximo histórico diario que marcó el pasado 18....
       </description>
       <pubDate>Mon, 09 Jul 2012 13:06:00 +0200</pubDate>
       <iso:language>es</iso:language>
      ▼<link>
         http://emm.newsexplorer.eu/NewsExplorer/clusteredition/es/20120709,abc-
         829766fc8ec1bdb9b0cfe41b7d0f40ec.html
       </link>
```


Figura 10: RSS de NewsExplorer en Español

El canal RSS está accesible desde la siguiente dirección:

http://emm.newsexplorer.eu/rss?type=clusters&language=es

Donde pone language se puede sustituir por cualquiera de los otros 18 idiomas manejados por News Explorer, de los cuales nosotros sólo vamos a utilizar seis: Inglés (en), español (es), francés (fr), italiano (it), alemán (de) y holandés (nl).

La codificación utilizada para determinar los países es la ISO 639-1[22], que consiste en 136 códigos de dos letras usados para identificar los idiomas principales del mundo.

Una vez el programa ha accedido a la página de un determinado clúster, tendrá que extraer los siguientes elementos:

- Enlace al clúster.
- Idioma del clúster.

[22] http://es.wikipedia.org/wiki/ISO_639-1

- Título del clúster
- Descripción del clúster.
- Noticias del clúster junto con los enlaces a las fuentes de origen y las horas de publicación.
- Nombres de países referenciados junto con los enlaces a las páginas de país de News Explorer y la frecuencia con la que aparecen en el clúster.
- Nombres de lugares referenciados junto con el código de país al que pertenecen, la latitud y longitud de su ubicación y el enlace a su ubicación en http://www.mapquest.com.
- Nombres de personas referenciadas junto con el enlace a la página de entidades de News Explorer (si existe) y la frecuencia con la que aparece en el clúster.
- Otros nombres referenciados junto con el enlace a la página de entidades de News Explorer y la frecuencia con la que aparece en el clúster.
- Título de la historia a la que pertenece el clúster (en caso de pertenecer a alguna) junto con el enlace a la página de historias de News Explorer y la fecha de inicio de la historia.
- Palabras clave o keywords que aparecen en la historia.
- Enlaces a clústers relacionados en los idiomas adicionales establecidos por el usuario, que guarden al menos el grado de semejanza con el propio clúster especificado por el usuario.
- Fecha del clúster.

El programa tendrá que acceder a su vez a dichos enlaces que conducen a otros clústers y recopilar sobre ellos la misma información mencionada en los puntos anteriores, con la excepción de los clústers relacionados.

La relación entre clústers y la semejanza que guardan entre sí también serán almacenadas en la base de datos.

En la Figura 11 podemos ver los apartados que tenemos que extraer. Aunque visualmente está bien estructurado y si estuviera en XML sería fácil extraerlo, más adelante veremos que obtener nuestros elementos a partir del código HTML no es una tarea sencilla.

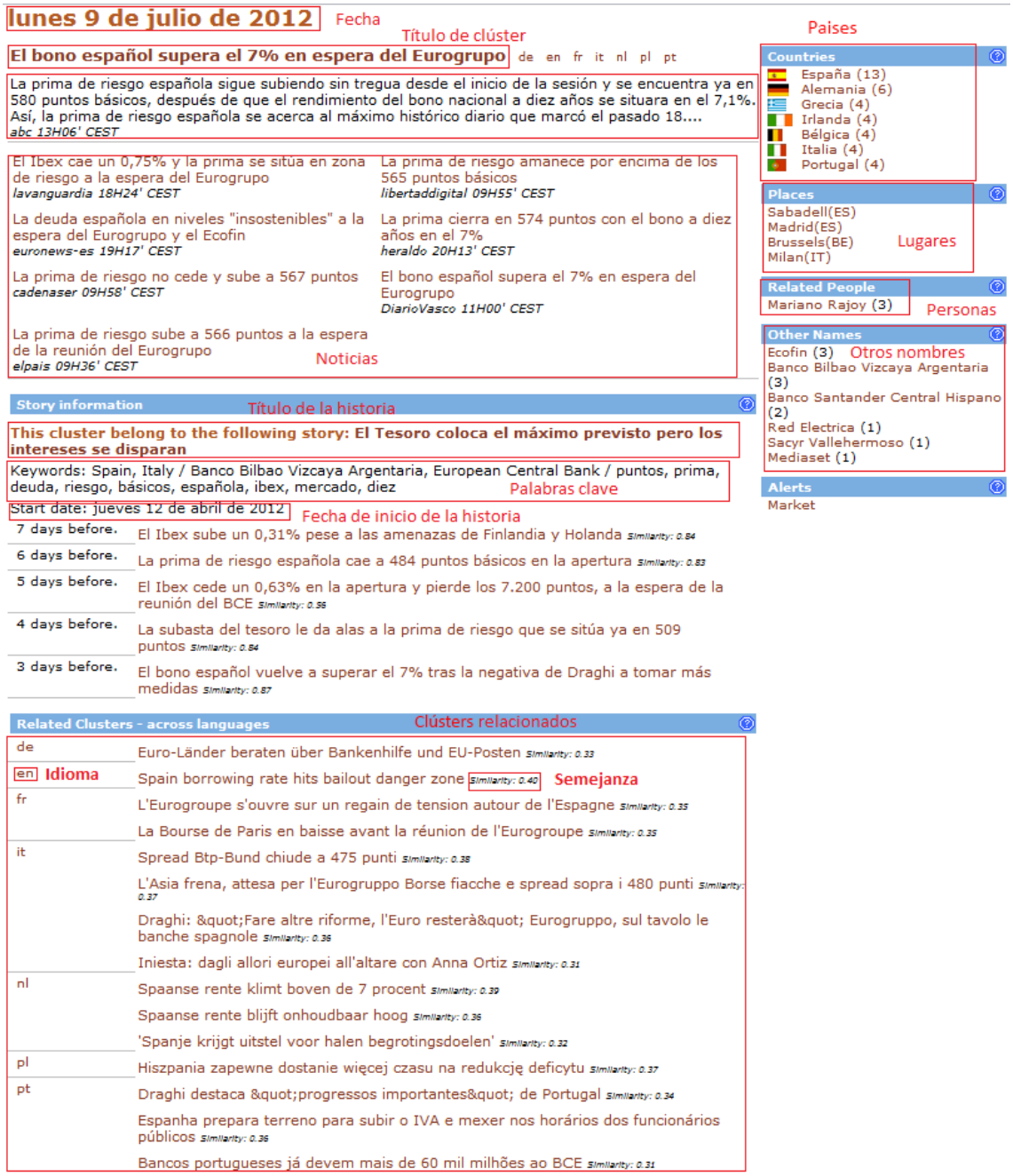


Figura 11: Página de un clúster de News Explorer

5.1 – Descripción de la interfaz de usuario

Nuestro programa utilizará una interfaz de línea de comandos, dado que no será un usuario quien lo ejecute sino el planificador de tareas cron. No obstante recibirá varios datos como argumentos de ejecución.

El primero será el lenguaje principal de la consulta que realizará el programa, que lo utilizará para acceder al canal RSS del News Explorer en ese idioma.

A continuación irán los lenguajes secundarios empleados para buscar clústers relacionados en dichos idiomas. Podrán introducirse todos los lenguajes secundarios que se deseen pero habrá de indicarse al menos uno.

Por último irá el umbral de semejanza que los clústers relacionados en los idiomas secundarios especificados deben igualar o superar para que se tengan en cuenta. Este umbral deberá tener un valor entre cero y uno y estará indicado en punto flotante.

Ejemplo: '0.5'
Ejemplo de ejecución completa: 'emm.pl es en fr it de 0.6'

5.2 – Modelo de Casos de Uso

En nuestro sistema el actor será el usuario que realice pruebas o el planificador que ejecute nuestra aplicación dependiendo de la situación. Hemos definido un solo caso de uso que, a pesar de ser complejo y cubrir múltiples operaciones, dichas operaciones se ejecutan siempre en el mismo orden y de manera secuencial. En la Figura 12 se muestra el diagrama de casos de uso con el Reloj del Sistema como actor.

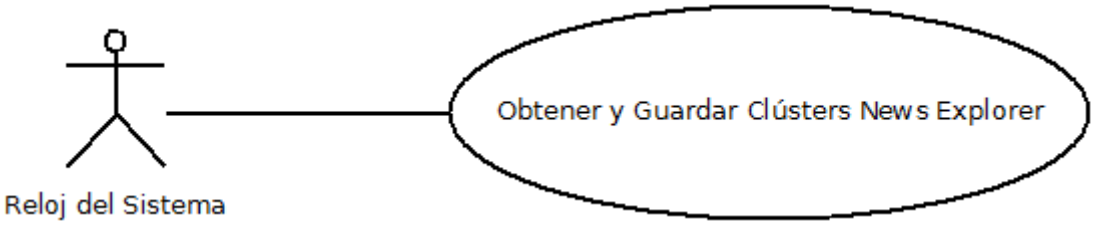


Figura 12: Diagrama de Casos de Uso

5.2.1 – Obtener y Guardar Clústers News Explorer

El programa se inicia verificando los parámetros lenguaje principal, lenguajes secundarios y umbral de semejanza. A continuación accede al canal RSS del lenguaje principal especificado y recopila los enlaces a clústers para trabajar más adelante con ellos.

Por cada enlace extraído del RSS, primero verifica que el clúster no se encuentre ya almacenado en la base de datos. Luego accede a la página del clúster y extrae la fecha, descripción, título, historia, noticias, personas, países, lugares, keywords, y otros nombres relacionados, además de una lista con los enlaces a los clústers relacionados que cumplan o superen el umbral de semejanza y que estén en alguno de los lenguajes secundarios especificados al ejecutar la aplicación.

A continuación se insertan todos esos datos en la base de datos excepto los de los clústers relacionados, que se guardarán al final.

Asimismo, por cada enlace de clúster relacionado, se accede a la página del clúster y se extraen todos los datos nombrados anteriormente para guardarlos en la base de datos acto seguido.

Una vez terminado el proceso de obtención y almacenamiento de clústers relacionados, el sistema guarda en la base de datos la relación entre dichos clústers y el clúster origen junto con la semejanza que comparten entre sí.

Por último, el programa escribe en un fichero log el número de registros insertados en cada tabla junto con la línea de ejecución del programa, la fecha y la hora de ejecución.

Escenario principal:

1. **Reloj:** Ejecuta el programa con los datos de entrada.
2. **Sistema:** Verifica los datos de entrada.
3. **Sistema:** Accede al canal RSS de News Explorer en el lenguaje principal especificado y recopila los enlaces a clústers.
4. **Sistema:** Por cada enlace:
    4.1. Accede al enlace del clúster y recopila sus datos.
    4.2. Almacena los datos en la base de datos.
    4.3.Recopila de la página los enlaces, las semejanzas y las lenguas de los clústers relacionados.
    4.4.Por cada enlace a un clúster relacionado:
        4.4.1. Accede al enlace del clúster y recopila sus datos.
        4.4.2. Almacena los datos en la base de datos.
    4.5. Almacena en la base de datos la relación de semejanza entre clústers.
5. **Sistema:** Escribe en un fichero de bitácora el número de registros insertados en la base de datos junto con la fecha y la hora de ejecución.

Extensiones:

Paso 2: Alguno de los datos de entrada falta o es incorrecto.

1. **Sistema:** Termina el programa mostrando un ejemplo de ejecución correcta junto con los lenguajes que se pueden utilizar.

Pasos 4.1 y 4.4.1: El clúster ya se encuentra en la base de datos.

1. **Sistema:** Continúa la ejecución del programa y pasa al siguiente clúster.

Paso 4.2 y 4.4.2: Alguno de los datos del clúster ya se ha guardado anteriormente.

1. **Sistema:** Se omite el dato en cuestión y se procede a introducir el siguiente dato.

5.3 - Modelo de Dominio

Para esta aplicación se ha creado un modelo de dominio basándonos en la estructura de los elementos en las páginas de clúster del News Explorer. Como se ve en la Figura 13, el elemento principal es el clúster y a su alrededor se van agregando los demás datos. La excepción son las palabras clave o keywords, que pertenecen a una historia.

De los clústers se quiere guardar el título, la descripción, el enlace al clúster, el idioma, y la fecha. Un clúster puede tener muchas noticias y, aunque una noticia pueda aparecer en varios clústers, esta relación no se va a tener en cuenta en nuestro modelo.

Además el clúster contiene varias personas, países y otros nombres, de los cuales además de guardar su nombre y enlace también se guardará la frecuencias con las que aparecen en dicho clúster. Este último dato es consecuencia de la relación y se guardará aparte. La razón es que queremos guardar sólo una vez los datos principales de cada persona, lugar y otro nombre y añadir cada vez sólo su relación con el clúster.

Un clúster puede tener varios lugares, cada uno con su nombre, código de país, enlace, latitud y longitud. Sin embargo, un clúster pertenece a una sola historia, mientras que en una historia puede haber múltiples clústers. De la historia se quieren guardar su título, su enlace y su fecha de inicio.

Una historia puede tener varias palabras clave y, aunque en News Explorer un keyword pueda aparecer en varias historias, del keyword sólo se quiere guardar su nombre por lo que esa relación no será guardada.

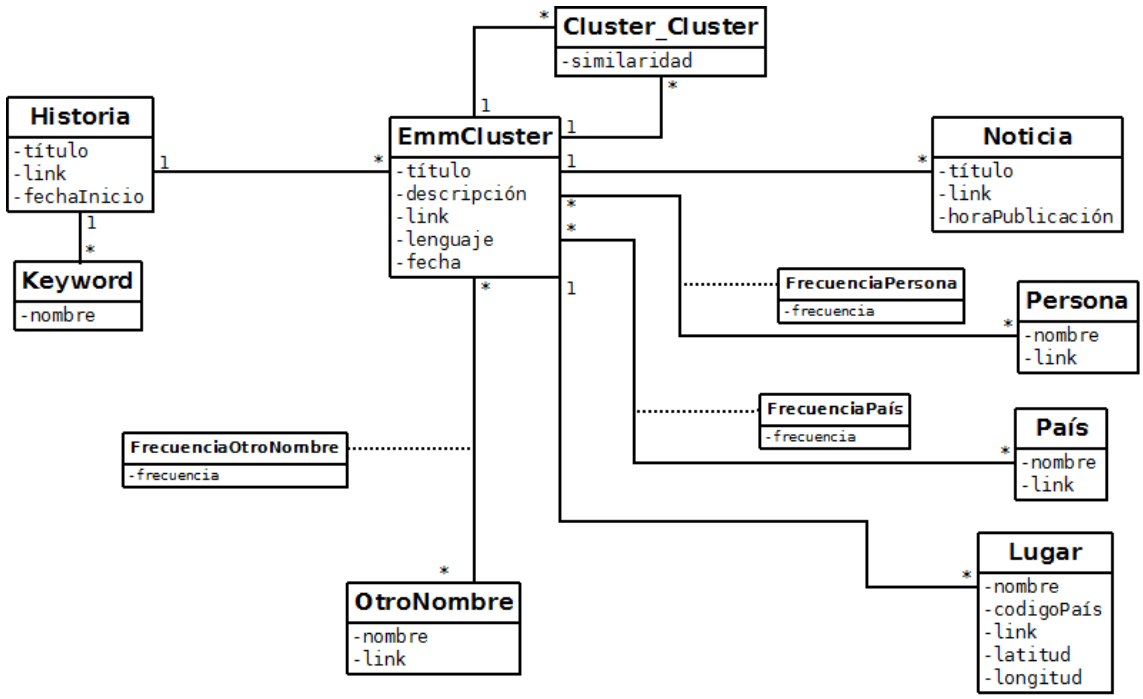


Figura 13: Modelo de Dominio

6 – Análisis

Aunque sólo disponemos de un caso de uso, lo vamos a descomponer de manera que sea más sencillo presentar los diagramas de secuencia del sistema. Además nos servirá para dividir las operaciones de cara a la especificación de sus contratos.

6.1 – Diagramas de Secuencia del Sistema y Contratos

A continuación se muestran los diagramas correspondientes a cada nivel de abstracción, junto con una breve explicación de las operaciones y sus parámetros de entrada y de salida. Seguidamente se especifican los contratos de las operaciones.

6.1.1 – Obtener y Guardar Clústers

Este caso de uso se puede dividir en las siguientes operaciones:

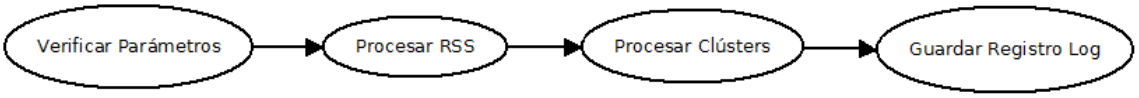


Figura 14: Descomposición de Obtener y Guardar Clústers News Explorer

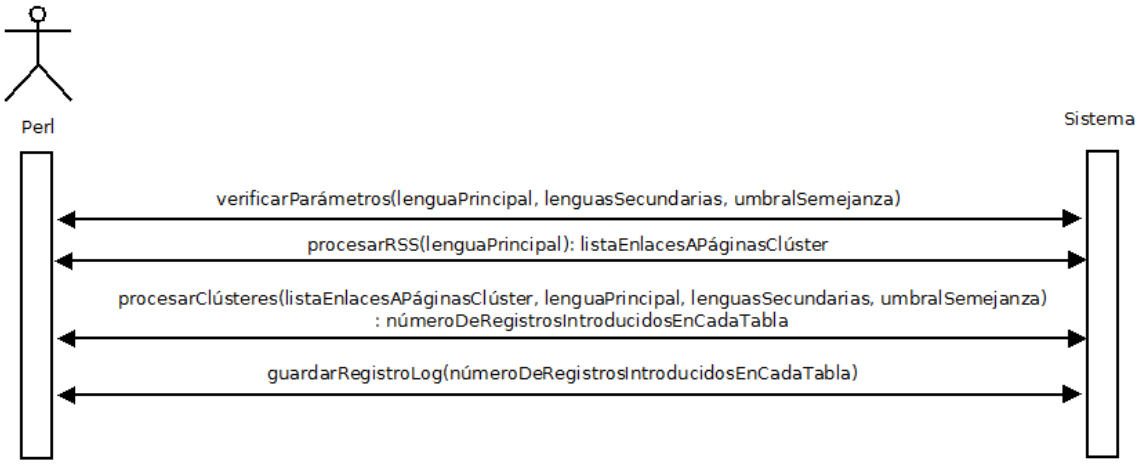


Figura 15: DSS Obtener y Guardar Clústers

Verificar Parámetros: la primera tarea del programa será comprobar los parámetros de entrada para verificar que no falta ninguno y que todos tienen el formato requerido.

Procesar RSS: A continuación el programa accede al canal RSS del News Explorer y extrae las direcciones web de todos los clústers que allí aparezcan.

Procesar Clústers: Después se procesan todos los clústers accediendo a dichas páginas, extrayendo los datos precisos y guardándolos en la base de datos. Esta operación es compleja y será desglosada en breve.

Guardar Registro Log: Para finalizar, el programa guarda en un fichero de texto la fecha y hora de ejecución, la línea de comandos ejecutada y el número de registros insertados en cada tabla.

Contrato verificarParámetros

Nombre: check_Params(INlang, OUTlangs, similarity)
Responsabilidades: Determinar si los parámetros de entrada tienen el formato correcto, donde **INlang** es la lengua principal, **OUTlangs** es una lista con las lenguas secundarias y **similarity** es el umbral de semejanza que han de cumplir los clústers relacionados para que se les tenga en cuenta.

- INlang debe ser una cadena de dos caracteres.
- OUTlang debe ser una lista de cadenas de dos caracteres que puede ser vacía.
- similarity debe ser un número real entre 0 y 1.

Precondición: Ø
Postcondición: Ø
Salida: Si alguno de los parámetros incumple las restricciones, el programa termina con un mensaje de error.

Contrato procesarRSS

Nombre: process_RSS(INlang)
Responsabilidades: A partir de la lengua principal **INlang** accede al canal RSS correspondiente del News Explorer, y extrae los enlaces a las páginas de clúster que allí aparezcan.
Precondición: INlang es una cadena de dos caracteres.
Postcondición: Ø
Salida: Devuelve una lista con los enlaces a las páginas de clúster.

Contrato guardarRegistroLog

Nombre: print_Log()
Responsabilidades: Escribe en un fichero el número de registros introducidos en cada tabla, el comando ejecutado junto con los parámetros de entrada y la fecha y hora de ejecución.
Precondición: El programa tiene permiso para escribir en la ruta especificada para el fichero. Las variables globales siguientes están declaradas e inicializadas:

- numclusters: Número de clústers introducidos en la base de datos.

- numnews: Número de noticias introducidas en la base de datos.
- numstories: Número de historias introducidas en la base de datos.
- numkeywords: Número de palabras clave introducidas en la base de datos.
- numpeople: Número de personas introducidas en la base de datos.
- numcountries: Número de países introducidos en la base de datos.
- numplaces: Número de lugares introducidos en la base de datos.
- numother: Número de otros nombres introducidos en la base de datos.

Postcondición: Los datos quedan grabados en el fichero 'emm.log'.
Salida: El número de registros introducidos en cada tabla, el comando ejecutado junto con los parámetros de entrada y la fecha y hora de ejecución.

6.1.2 – Procesar Clústers

La operación Procesar Clústers se descompone en el siguiente esquema:

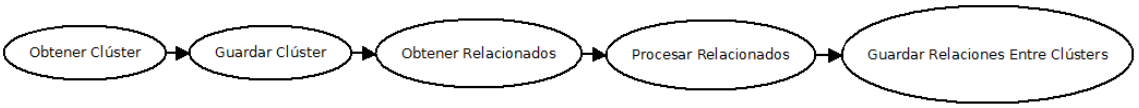


Figura 16: Descomposición de Procesar Clústers

Por cada clúster encontrado en el RSS se ejecutarán estas operaciones de la Figura 17:

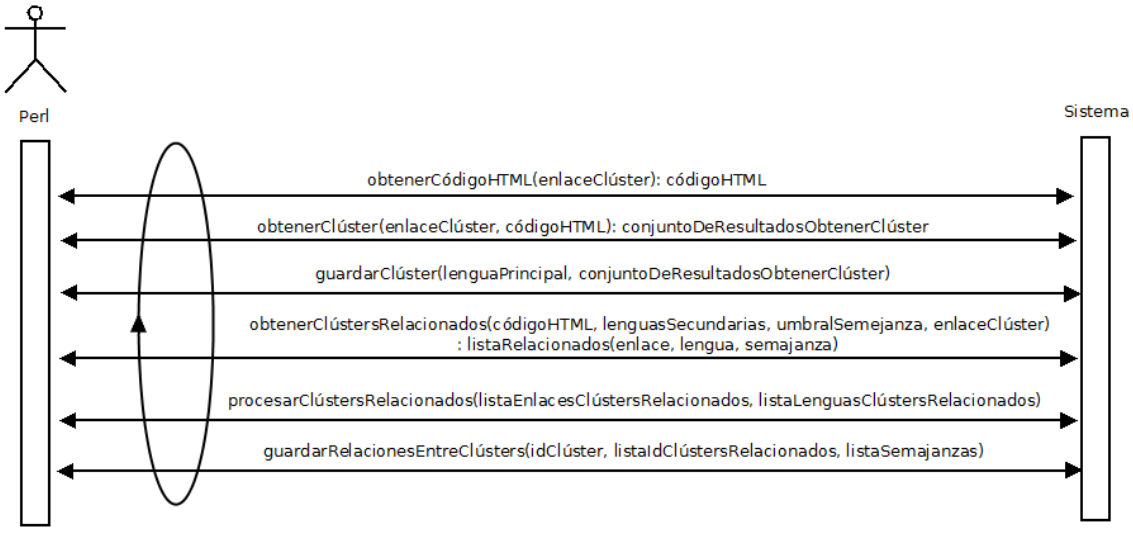


Figura 17: DSS Procesar Clústers

Obtener Código HTML: El programa accede a la página del clúster y extrae el código HTML para que sea utilizado en otras operaciones. Es importante que dicho acceso sólo se produzca una vez por clúster para no saturar el servidor.

Obtener Clúster: El programa accede a la página web del clúster y extrae los datos de las personas, países, etc. relacionados con el mismo. Agrupamos los resultados de la operación obtener Clúster porque todos se utilizan en la siguiente operación. De este modo el diagrama queda más claro.

Guardar Clúster: Seguidamente se almacenan dichos datos en la base de datos. Para ello la operación necesita también la lengua del clúster, que en este caso es la lengua principal pasada al programa.

Obtener Clústers Relacionados: A continuación se extraen los enlaces, lenguas y semejanzas de los clústers relacionados que cumplan con el umbral de semejanza especificado al iniciar el programa y estén en alguna de las lenguas secundarias.

Procesar Clústers Relacionados: En esta operación se accede a las páginas de todos los clústers relacionados (Obtener clúster) cuyos enlaces hemos recopilado antes y una vez más se extraen sus datos y se almacenan (Guardar Clúster) en la base de datos.

Guardar Relaciones Entre Clústers: Una vez analizados todos los clústers relacionados con el clúster origen, se procede a insertar la relación entre ellos en la base de datos. Esto es así porque utilizando una base de datos relacional tenemos una restricción de claves foráneas, que nos obliga a que tenga que existir el clúster antes de ser referenciado.

Contrato obtenerClústersRelacionados

Nombre: get_Related(html, langs, similarity, uri)
Responsabilidades: Obtiene el enlace, idioma y semejanza de los clústers relacionados que igualen o superen el umbral se semejanza **similarity** y que estén en alguno de los idiomas contenidos en la lista **langs**.
Precondición: Los clústers relacionados están localizados en el código HTML en enlaces con alguno de los códigos de lenguaje **langs**. Ej: ‘href=”.***/es/***.*”’. El parámetro de entrada **uri** contiene la dirección absoluta de la página, **similarity** un número real entre 0 y 1, **langs** una lista de códigos de lenguaje de dos caracteres y **html** el código HTML de la página.
Postcondición: Ø
Salida: Lista de clústers relacionados. Cada uno con enlace, idioma y semejanza que guardan con el clúster en el que aparecen. Se devuelve en tres listas.

Contrato guardarRelacionesEntreClústers

Nombre: insert_CCs(sourcecluster, targetclusters, similarities)
Responsabilidades: Almacenar en la base de datos la relación entre clústers junto con la semejanza que comparten.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. Los clústers **sourcecluster** y **targetclusters** están almacenados en la base de datos.
Postcondición: Por cada targetcluster en la lista targetclusters y cada similarity en la lista similarities se almacena una tupla en la tabla CLUSTER_CLUSTER con los valores (sourcecluster, targetcluster, similarity) en las columnas (sourceCluster, targetCluster, similarity).
Salida: Ø

6.1.3 – Procesar Clústers Relacionados

Por cada clúster relacionado se ejecutan estas dos operaciones, ya explicadas anteriormente.

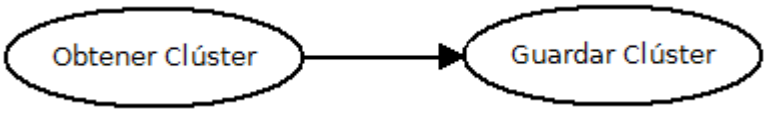


Figura 18: Descomposición de Procesar Clústers Relacionados

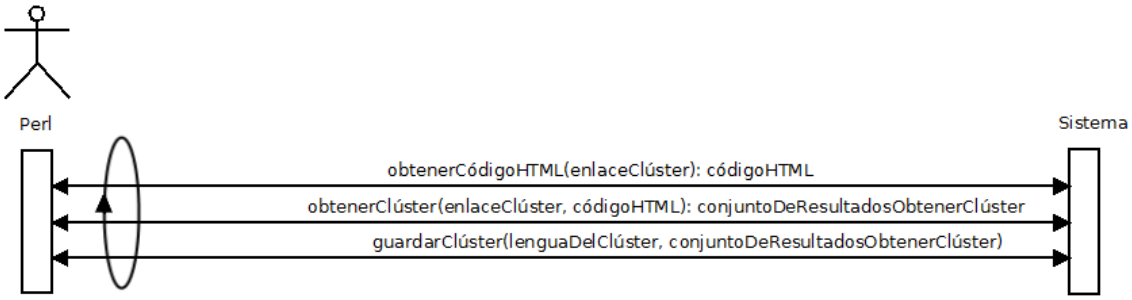


Figura 19: DSS Procesar Clústers Relacionados

La diferencia entre Procesar Clústers y Procesar Clústers Relacionados es que en el primer caso, además de obtener y guardar los datos del clúster, se obtienen los clústers relacionados para ser procesados y en el segundo caso no. Ésta es la razón principal por la que he separado la operación obtener Clústers Relacionados de obtener Clúster. De esta manera las operaciones 'obtener Clúster' y 'guardar Clúster' se pueden reutilizar.

A efectos prácticos estas dos operaciones se agruparán más adelante en una sola porque los datos de salida obtenerClúster son los datos de entrada de guardarClúster.

Por ejemplo, los datos que recopilamos sobre los países de un clúster mediante la operación obtenerPaíses (nombres, enlaces, frecuencias), no se utilizan en ningún otro punto del programa aparte de en la operación insertarPaíses.

6.1.4 – Obtener Clúster

Esta operación engloba la extracción de todos los datos de una página de clúster.

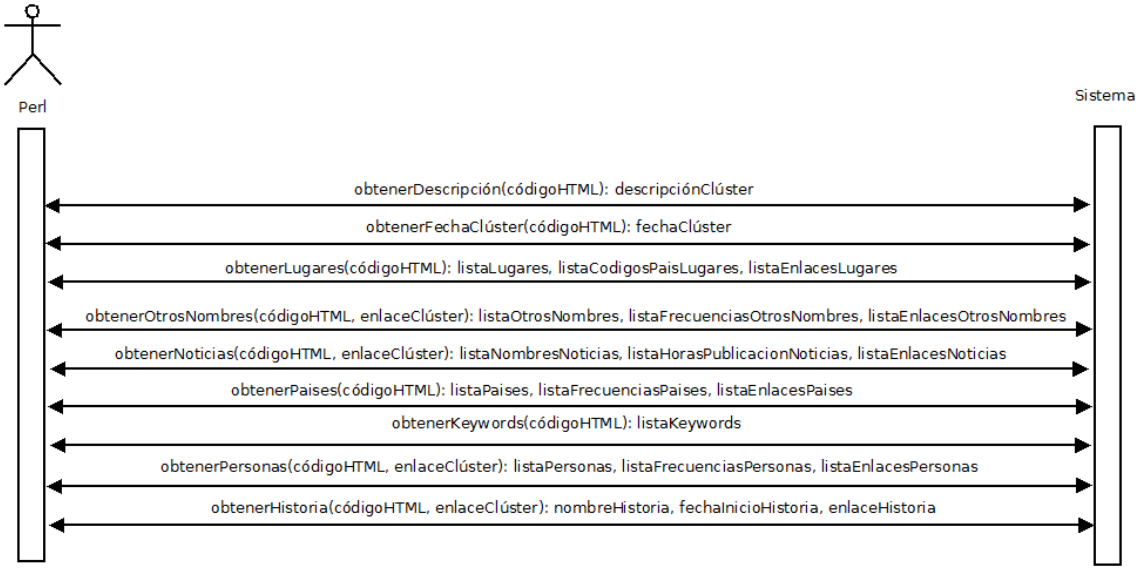


Figura 20: DSS Obtener Clúster

Como hemos dicho antes, aquí se extraen todos los datos a partir de una página de clúster. Para ello se obtiene el código HTML y se trabaja sobre él. En el apartado de contratos se describe lo que hace cada operación, sus parámetros y lo que devuelve. Del mismo modo, en el apartado de diseño se detalla minuciosamente cada subprograma mediante pseudocódigo.

Contrato obtenerDescripción

Nombre: get_Description(html)
Responsabilidades: Obtener la descripción del clúster a partir del código HTML.
Precondición: Si existe descripción, ésta se encontrará entre las etiquetas y del código HTML.
Postcondición: Ø
Salida: Descripción del clúster.

Contrato obtenerFechaClúster

Nombre: get_ClusterDate(html)
Responsabilidades: Obtener la fecha del clúster a partir del código HTML.
Precondición: La fecha está entre las etiquetas y del código HTML.
Postcondición: Ø
Salida: Fecha del clúster literal.

Contrato obtenerLugares

Nombre: get_Places(html)
Responsabilidades: Obtener la información de los lugares relacionados con el clúster.
Precondición: Los lugares están localizados en el código HTML a partir de la palabra ‘Places’ hasta la primera etiqueta . Más concretamente en etiquetas <a> con atributos target=”_blank” y class=”headline_link”. El parámetro de entrada contiene el código HTML de la página.
Postcondición: Ø
Salida: Lista de lugares con nombre, código del país al que pertenecen y enlace. Se devuelve en tres listas.

Contrato obtenerOtrosNombres

Nombre: get_Other(html, uri)
Responsabilidades: Obtener la información de otros nombres relacionados con el clúster.
Precondición: Los otros nombres están localizados en el código HTML a partir de la palabra ‘Other Names’ hasta la primera etiqueta . Más concretamente en etiquetas <a> con el atributo class=”headline_link”. El parámetro de entrada **uri** contiene la dirección absoluta de la página y **html** el código HTML de la misma.
Postcondición: Ø
Salida: Lista de otros nombres con nombre, frecuencia con la que aparecen en el cluster y enlace. Se devuelve en tres listas.

Contrato obtenerNoticias

Nombre: get_News(html, uri)
Responsabilidades: Obtener la información de las noticias pertenecientes al clúster.
Precondición: Las noticias están localizadas en el código en etiquetas <a> con el atributo target=”EMMARTICLE”. Las horas de publicación se encuentran entre las etiquetas y . El parámetro de entrada uri contiene la dirección absoluta de la página y html el código HTML de la misma.
Postcondición: Ø

Salida: Lista de noticias con nombre, hora de publicación y enlace. Se devuelve en tres listas.

Contrato obtenerPaíses

Nombre: get_Countries(html)
Responsabilidades: Obtener la información de los países relacionados con el clúster.
Precondición: Los países están localizados en el código en etiquetas con el atributo class=”country_link cluster_group_item” y sus enlaces contienen el tag countryedition. El parámetro de entrada **html** contiene el código HTML de la página.
Postcondición: Ø
Salida: Lista de países con nombre, frecuencia con la que aparecen en el clúster y enlace. Se devuelve en tres listas.

Contrato obtenerKeywords

Nombre: get_Keywords(html)
Responsabilidades: Obtener las keywords (palabras clave) relacionadas con el clúster.
Precondición: Las keywords están localizadas en el código a partir de la palabra ‘Keywords:’ hasta la primera etiqueta </p>. El parámetro de entrada **html** contiene el código HTML de la página.
Postcondición: Ø
Salida: Lista de keywords.

Contrato obtenerPersonas

Nombre: get_People(html, uri)
Responsabilidades: Obtener la información de las personas relacionadas con el clúster.
Precondición: Las personas están localizadas en el código a partir de la palabra ‘Related People’ hasta la primera etiqueta . Más concretamente las personas con enlace se encuentran en etiquetas <a> con el atributo class=”headline_link” y las que no tienen enlace en etiquetas . El parámetro de entrada **uri** contiene la dirección absoluta de la página y **html** el código HTML de la misma.
Postcondición: Ø
Salida: Lista de personas con nombre, frecuencia con la que aparecen en el clúster y enlace. Se devuelve en tres listas.

Contrato obtenerHistoria

Nombre: get_Story(html, uri)
Responsabilidades: Obtener la información de la historia de la que forma parte el clúster.

Precondición: La historia está localizada en el código a partir de la frase 'This cluster belong to the following story:'. Más concretamente en una etiqueta <a> con el atributo class="headline_link_bold". El parámetro de entrada **uri** contiene la dirección absoluta de la página y **html** el código HTML de la misma.
Postcondición: Ø
Salida: Nombre, fecha de inicio y enlace de la historia.

6.1.5 – Guardar Clúster

Mediante este conjunto de operaciones se almacena todo lo que hemos extraído de un clúster en la base de datos. Los datos se van insertando tabla por tabla al mismo tiempo que se incrementan los contadores de registros introducidos por tabla. Dichos contadores son los que se utilizan para guardar un registro log al final de la ejecución.

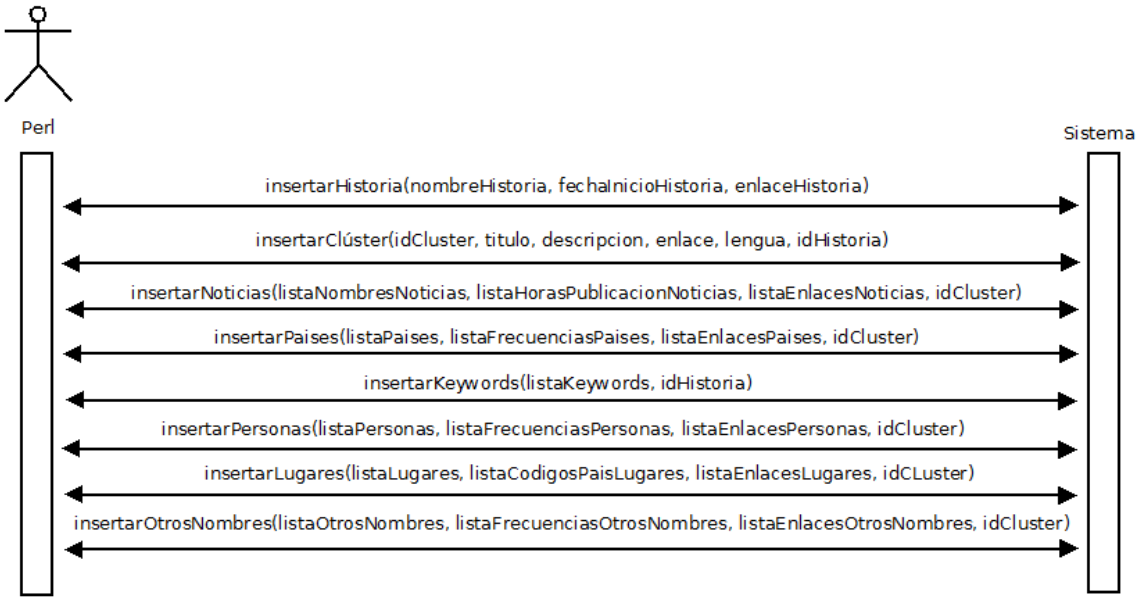


Figura 21: DSS Guardar Clúster

En el diagrama aparecen algunos identificadores de tabla que utilizamos para relacionar los elementos en la base de datos. En el apartado de diseño lo expondremos con más detalle.

Contrato insertarHistoria

Nombre: insert_Story(storyId, storytitle, storystartdate, storylink)
Responsabilidades: Almacenar los datos de una historia en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. La historia no está almacenada en la base de datos.

Postcondición: En la tabla STORY se ha guardado una tupla con los valores (storyId, storytitle, storylink, storystartdate) en las columnas (id, title, link, startDate).
Salida: Ø

Contrato insertarClúster

Nombre: insert_Cluster(clusterId, title, description, link, lang, clusterdate, storyId)
Responsabilidades: Almacenar los datos de un clúster en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El parámetro **storyId** puede estar vacío, pero si no lo está, la historia con dicho id se ha de encontrar almacenada en la base de datos. El parámetro **description** también puede estar vacío.
Postcondición: Se ha insertado una nueva tupla en la tabla EMMCLUSTER con los valores (clusterId, title, description, link, lang, clusterdate, storyId) en las columnas (id, title, description, link, lang, clusterdate, storyId).
Salida: Ø

Contrato insertarNoticias

Nombre: insert_News(newstitles, newspubtimes, newslinks, clusterId)
Responsabilidades: Almacenar los datos de las noticias relacionadas con el clúster **clusterId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El clúster **clusterId** está almacenado en la base de datos.
Postcondición: Por cada title de la lista newstitles, cada link de la lista newslinks y cada pubTime de la lista newspubtimes se ha insertado una nueva tupla en la tabla NEWS con los valores (sha1(link), clusterId, title, link, pubtime) en las columnas (id, clusterId, title, link, pubTime). Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

Contrato insertarPaíses

Nombre: insert_Countries(countrynames, countryfreqs, countrylinks, clusterId)
Responsabilidades: Almacenar los datos de los países relacionados con el clúster **clusterId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El clúster **clusterId** está almacenado en la base de datos.
Postcondición: Por cada name de la lista countrynames, cada link de la lista countrylinks y cada frequency de la lista countryfreqs se ha añadido una tupla a la tabla COUNTRY con los valores (name, link) en las columnas (name, link) y una tupla en la tabla COUNTRY_CLUSTER con los valores (clusterId, name, frequency) en las columnas (clusterId, countryName, frequency). Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

Contrato insertarKeywords

Nombre: insert_Keywords(keywords, storyId)
Responsabilidades: Almacenar keywords relacionadas con la historia **storyId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. La historia con id **storyId** se encuentra almacenada en la base de datos.
Postcondición: Por cada name en la lista keywords se ha insertado una tupla en la tabla KEYWORD con los valores (name, storyId) en las columnas (name, storyId). Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

Contrato insertarPersonas

Nombre: insert_People(peoplenames, peoplefreqs, peoplelinks, clusterId)
Responsabilidades: Almacenar los datos de las personas relacionadas con el clúster **clusterId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El clúster **clusterId** está almacenado en la base de datos.
Postcondición: Por cada name de la lista peoplenames, cada link de la lista peoplelinks y cada frequency de la lista peoplefreqs se ha añadido una tupla a la tabla PERSON con los valores (name, link) en las columnas (name, link) y una tupla en la tabla PERSON_CLUSTER con los valores (clusterId, name, frequency) en las columnas (clusterId, personName, frequency). Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

Contrato insertarLugares

Nombre: insert_Places(placenames, countrycodes, placelinks, clusterId)
Responsabilidades: Almacenar los datos de los lugares relacionados con el clúster **clusterId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El clúster **clusterId** está almacenado en la base de datos.
Postcondición: Por cada name de la lista placenames, cada link de la lista placelinks y cada countryCode de la lista countrycodes se ha añadido una tupla a la tabla PLACE con los valores (sha1(link), name, link, latituide, longitude, countryCode) en las columnas (id, name, link, latituide, longitude, countryCode) y una tupla en la tabla PLACE_CLUSTER con los valores (clusterId, sha1(link)) en las columnas (clusterId, placeId), donde latitude y longitude son dos valores que forman parte del link. Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

Contrato insertarOtrosNombres

Nombre: insert_Others(othernames, otherfreqs, otherlinks, clusterId)
Responsabilidades: Almacenar los datos de los otros nombres relacionadas con el clúster **clusterId** en la base de datos.
Precondición: Los parámetros de entrada deben estar correctamente inicializados. El clúster **clusterId** está almacenado en la base de datos.
Postcondición: Por cada name de la lista othernames, cada link de la lista otherlinks y cada frequency de la lista otherfreqs se ha añadido una tupla a la tabla OTHER con los valores (name, link) en las columnas (name, link) y una tupla en la tabla OTHER_CLUSTER con los valores (clusterId, name, frequency) en las columnas (clusterId, otherName, frequency). Sólo se insertará una tupla en una tabla si dicha tupla no se encuentra ya en la tabla.
Salida: Ø

7 – Diseño

En este apartado se muestra el diagrama Entidad-Relación de la base de datos, explicando la razón de cada tabla, atributo y relación. Asimismo, también veremos el pseudocódigo de las operaciones anteriormente introducidas, que además de ser la base para la implementación, es de gran utilidad para comprender lo que hace nuestro script.

7.1 – Base de Datos

En la siguiente figura aparecen las tablas junto con sus atributos y relaciones. El diseño es casi idéntico al que aparece en el apartado 5.3 - Modelo de Dominio5.3 - Modelo de Dominio.

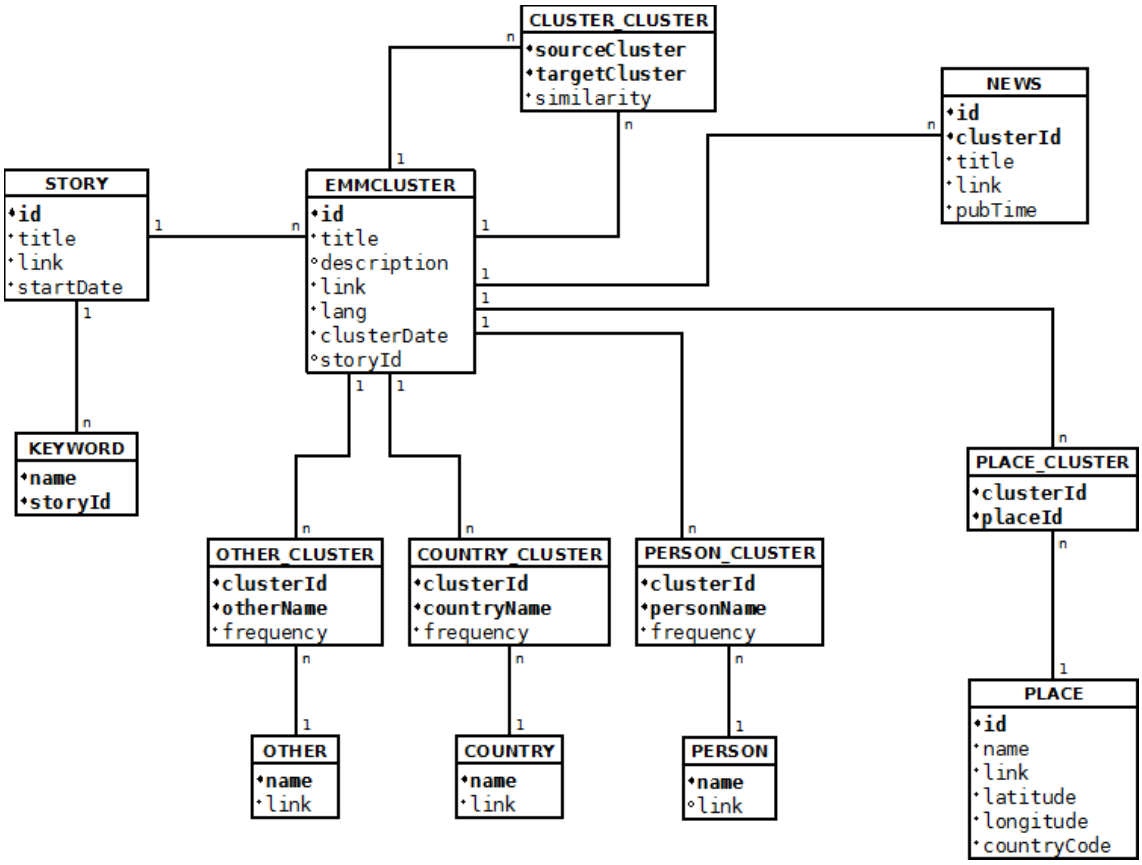


Figura 22: Diagrama Entidad-Relación de la base de datos

Tabla EMMCLUSTER

Descripción:

Ésta es la tabla principal de la base de datos porque representa los clústers, nuestro punto de entrada al recopilar los datos del News Explorer. Sobre ella se construyen las demás tablas mediante relaciones. Un clúster tiene muchas noticias, lugares, personas, países y otros nombres relacionados.

Atributos:

- Id: Identificador unívoco de clúster y clave primaria. Es una secuencia de 40 caracteres hexadecimales resultado de aplicar la función SHA1 al enlace del mismo. Se utiliza en otras tablas como clave foránea.
- Title: Título del clúster. Es el título de la noticia más representativa del clúster.
- Description: Descripción del clúster. Es la descripción de la noticia más representativa del clúster.
- Link: Enlace de la página del clúster.
- Lang: Idioma del clúster. Secuencia de dos caracteres: es, en, de, fr, it,…
- ClusterDate: Fecha de creación del clúster.
- StoryId: Identificador de la historia a la que pertenece el clúster. Los clústers no siempre pertenecen a una historia, por lo que este atributo es opcional. Es clave foránea de la tabla STORY.

Tabla STORY

Descripción:

En esta tabla se guardan las historias que se van recopilando del News Explorer. Además de estar relacionada con la tabla EMMCLUSTERS también lo está con la tabla KEYWORDS, porque una historia tiene cero o más palabras clave. Como decimos en el apartado 5.3 - Modelo de Dominio, aunque una palabra clave puede aparecer en más de una historia no vamos a guardar esa relación.

Atributos:

- Id: Identificador unívoco de historia y clave primaria de la tabla. Es una secuencia de 40 caracteres hexadecimales resultado de aplicar la función SHA1 al enlace de la misma. Se utiliza en otras tablas como clave foránea.

- Title: Título de la historia. Es el título del clúster más representativo de la historia.
- Link: Enlace de la página de la historia.
- StartDate: Fecha de inicio de la historia. Es la fecha de inicio del primer clúster que pertenece a la historia.

Tabla KEYWORD

Descripción:

En esta tabla se almacenan todas las palabras clave que aparecen en las páginas de clústers. Como las palabras clave aparecen en la página solamente si el clúster pertenece a una historia, hemos relacionado la tabla con STORY y no con EMMCLUSTER. Esta tabla tiene una clave primaria compuesta por la propia palabra y el identificador de historia en la que aparece.

Atributos:

- Name: Palabra clave. Forma parte de la clave primaria.
- storyId: Identificador de historia. Forma parte de la clave primaria y es clave foránea de la tabla STORY.

Tabla NEWS

Descripción:

En esta tabla guardamos los datos de las noticias recuperadas en News Explorer. Está relacionada con la tabla EMMCLUSTER su clave primaria está compuesta por el identificador de noticia y el identificador del clúster en el que aparece.

Atributos:

- Id: Identificador unívoco de noticia y parte de la clave primaria de la tabla. Es una secuencia de 40 caracteres hexadecimales resultado de aplicar la función SHA1 al enlace de la misma.
- ClusterId: Identificador de clúster y parte de la clave primaria de la tabla. Además es clave foránea de la tabla EMMCLUSTER.
- Title: Título de la noticia.
- Link: Enlace de la página de la noticia.

- PubTime: Hora de publicación de la noticia.

Tabla OTHER

Descripción:

En esta tabla se almacenan los otros nombres relacionados con el clúster. A menudo suelen ser nombres de entidades, eventos u organizaciones. La frecuencia con la que aparece el nombre en el clúster estará en la tabla intermedia OTHER_CLUSTER, como se describe más adelante.

Atributos:

- Name: Nombre de la entidad que queremos guardar. Es la clave primaria de la tabla.
- Link: Enlace de la página del otro nombre.

Tabla COUNTRY

Descripción:

Aquí guardamos la información de los países relacionados con el clúster. Como en el caso anterior, su frecuencia será guardada en una tabla intermedia. Esta vez en COUNTRY_CLUSTER.

Atributos:

- Name: Nombre del país. Es la clave primaria de la tabla.
- Link: Enlace de la página del país.

Tabla PERSON

Descripción:

En esta tabla guardaremos el nombre de la persona relacionada con el clúster junto con el enlace a la página de la persona. No todas las personas disponen de una página propia

en News Explorer, por lo tanto el atributo enlace será opcional. La frecuencia con la que aparece en el clúster se guardará en la tabla PERSON_CLUSTER.

Atributos:

- Name: Nombre de la persona. Es la clave primaria de la tabla.
- Link: Enlace de la página de la persona. Como hemos dicho es opcional.

Tabla PLACE

Descripción:

Esta tabla almacena los datos de los lugares relacionados con un clúster. Sin embargo, como un lugar es único en el mundo y queremos extraer un gran número de datos, lo vamos a representar una sola vez en la base de datos, creando una tabla intermedia PLACE_CLUSTER para guardar la relación entre lugares y clústers.

Atributos:

- Id: Identificador unívoco de lugar y clave primaria de la tabla. Es una secuencia de 40 caracteres hexadecimales resultado de aplicar la función SHA1 al enlace del mismo. Aparece en PLACE_CLUSTER como clave foránea.
- Name: Nombre del lugar. Es la clave primaria de la tabla.
- Link: Enlace a la ubicación del lugar en MapQuest.
- Latitude: Latitud del lugar. Se extrae del enlace.
- Longitude: Longitud del lugar. Se extrae del enlace.
- CountryCode: Junto con el nombre del lugar aparece un código de dos caracteres que representa el país en el que éste se encuentra.

Tabla CLUSTER_CLUSTER

Descripción:

Aquí guardamos las relaciones entre clústers multilingües. Tenemos dos atributos con los identificadores del clúster origen y el clúster destino y un tercero con la semejanza

que guardan entre ellos. La tabla está relacionada con EMMCLUSTER por doble partida.

Atributos:

- SourceCluster: Identificador del clúster origen. Es el identificador de un clúster accedido a través del canal RSS de News Explorer. Forma parte de la clave primaria y es clave foránea de EMMCLUSTER.
- TargetCluster: Identificador del clúster destino. Es el identificador de un clúster relacionado con otro al que se ha accedido a través del canal RSS. Forma parte de la clave primaria y es clave foránea de EMMCLUSTER.
- Similarity: Semejanza existente entre ambos clústers. Es un número real entre 0 y 1.

Tabla OTHER_CLUSTER

Descripción:

En esta tabla intermedia guardamos la relación entre clústers y otros nombres junto con la frecuencia con la que aparece un nombre en un clúster.

Atributos:

- ClusterId: Identificador del clúster de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla EMMCLUSTER.
- OtherName: Nombre del 'otro nombre' de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla OTHER.
- Frequency: Frecuencia con la que el otro nombre aparece en el clúster.

Tabla COUNTRY_CLUSTER

Descripción:

Aquí es donde se almacena la relación entre países y clústers. Igualmente se almacena la frecuencia con la que aparece un país en un clúster.

Atributos:

- ClusterId: Identificador del clúster de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla EMMCLUSTER.
- CountryName: Nombre del país de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla COUNTRY.
- Frequency: Frecuencia con la que el país aparece en el clúster.

Tabla PERSON_CLUSTER

Descripción:

Es la tabla intermedia entre EMMCLUSTER y PERSON. En ella guardamos la relación entre una entidad de cada tabla junto con la frecuencia con la que una persona aparece en un determinado clúster.

Atributos:

- ClusterId: Identificador del clúster de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla EMMCLUSTER.
- PersonName: Nombre de la persona de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla PERSON.
- Frequency: Frecuencia con la que la persona aparece en el clúster.

Tabla PLACE_CLUSTER

Descripción:

Esta tabla intermedia no contiene información adicional a parte de la relación entre lugares y clústers. Pero la hemos creado para no tener toda la información de un lugar repetida varias veces.

Atributos:

- ClusterId: Identificador del clúster de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla EMMCLUSTER.

- PlaceId: Identificador del lugar de la relación. Forma parte de la clave primaria de la tabla y es clave foránea de la tabla PLACE.

7.2 – Pseudocódigo por operaciones

7.2.1 – Programa principal

Comprobación de los parámetros de entrada → **check_Params**
Se obtienen los enlaces de los clústers que aparecen en el RSS del primer lenguaje pasado como parámetro → **process_RSS**
Por cada enlace de clúster del RSS:
{
    Se obtiene el id del clúster a partir de su enlace
    Si el clúster existe → **exists_Cluster**: Pasamos al siguiente enlace
    Dormir programa 1 segundo para no sobrecargar el servidor EMM
    Obtener la página del clúster actual
    Obtener y almacenar datos del clúster →**getsave_Cluster**
    Obtener enlaces, idiomas y semejanzas de los clústers relacionados
    → **get_Related**
    Por cada clúster relacionado:
    {
        Se obtiene el id del clúster a partir de su enlace
        Si el clúster existe → **exists_Cluster**: Siguiente clúster
        Dormir programa 1 segundo para no sobrecargar el servidor EMM
        Obtener la página del clúster actual
        Obtener y almacenar datos del clúster → **getsave_Cluster**
    }
    Almacenar relaciones entre el clúster del bucle exterior y los clústers del bucle interior con sus semejanzas → **insert_CCs**
}
Guardar contadores de registros insertados en la base de datos en un fichero
→ **print_Log**

7.2.2 – Funciones que aparecen en el programa principal

Check_Params(@ARGV):

Si no nos han pasado parámetros terminamos la ejecución con un mensaje de error
Si nos han pasado menos de 3 parámetros terminamos la ejecución con un mensaje de error
Si el último parámetro (umbral de semejanza) no está entre 0 y 1 terminamos la ejecución con un mensaje de error
Si alguno de los demás parámetros (idiomas) es algo distinto a una cadena de dos caracteres alfabéticos, terminamos la ejecución con un mensaje de error

Process_RSS(INlang):

Accedemos al canal RSS de News Explorer en el idioma INlang
Obtenemos una lista de enlaces de páginas de clúster a partir del código XML del RSS
Devolvemos la lista

exists_Cluster(clusterId):

Inicializar la variable resultado a 0
Llamar a la función **db_Connect**
Se prepara la consulta a la base de datos
Se ejecuta la consulta con el parámetro id, correspondiente al id del clúster que se quiere consultar
Si se ha encontrado dicho clúster se cambia la variable resultado a 1
Llamar a la función **db_Disconnect**
Devolver el resultado

Getsave_Cluster(uri, html, clusterId, cluster_lang):

Obtener fecha → **get_ClusterDate**
Obtener descripción → **get_Description**
Obtener noticias → **get_News**
Obtener países → **get_Countries**
Obtener lugares → **get_Places**
Obtener personas → **get_People**
Obtener otros nombres → **get_Other**
Obtener keywords → **get_Keywords**
Obtener historia → **get_Story**
Si se ha obtenido la historia:
{
    Calcular el id de la historia a partir de su enlace
    Si no existe la historia en la base de datos →not **exists_Story**:
    {
        Insertar historia → **insert_Story**
    }
    Insertar Keywords → **insert_Keywords**
}
Insertar Clúster → **insert_Cluster**
Insertar Personas → **insert_People**
Insertar Noticias → **insert_News**
Insertar Otros Nombres → **insert_Other**
Insertar Lugares → **insert_Places**

Get_Related(html, langs, similarity, uri):

Por cada lenguaje en langs:
{
    Obtener una lista con todos los enlaces, los lenguajes y las semejanzas a partir del código html mediante una expresión regular
}
Pasar los enlaces, los lenguajes y las semejanzas a tres listas diferentes
Sólo conservar los que cumplen la restricción de tener una semejanza mayor o igual a la pasada como parámetro similarity
Convertir los enlaces relativos en absolutos a partir del parámetro uri
Devolver las listas de enlaces, lenguajes y semejanzas

insert_CCs(sourceCluster, targetClusters, similarities):

Llamar a la función **db_Connect**
Por cada clúster relacionado de la lista targetClusters:
{
Se ejecuta la inserción de la relación entre los clústers sourceCluster y targetCluster con los parámetros sourceCluster, targetCluster y similarity
}
Llamar a la función **db_Disconnect**

print_Log():

Escribir en la salida estándar los contadores de registros insertados en las tablas EMMCLUSTER, NEWS, STORY, KEYWORD, PERSON, COUNTRY, PLACE y OTHER (En la ejecución se redirecciona todo a un fichero).

7.2.3 – Funciones que aparecen en la función getsave_Cluster

get_ClusterDate(html):

Obtener la fecha del clúster a partir del código html mediante una expresión regular
Si se ha obtenido dicha fecha:
{
Sustituir los códigos html para caracteres no-ASCII (á) por sus respectivos caracteres (á).
}
Devolver la fecha decodificada

get_Description(html):

Obtener la descripción del clúster a partir del código html mediante una expresión regular
Si se ha obtenido dicha descripción:
{
Sustituir los códigos html para caracteres no-ASCII (á) por sus respectivos caracteres (á)
}
Devolver la descripción decodificada

get_News(html, uri):

Obtener en una lista todos los enlaces relativos y los títulos de noticia del clúster a partir del código html mediante una expresión regular
Obtener en una lista todas las horas de publicación de las noticias a partir del código html mediante una expresión regular
Pasar los enlaces y los títulos a dos listas diferentes
Convertir los enlaces relativos en absolutos a partir del parámetro uri
Aplicar la función **decode_list** a la lista de títulos de noticia
Devolver las listas de títulos, horas de publicación y enlaces

get_Countries(html):

Obtener una lista con todos los nombres de país junto con la frecuencia con la que aparecen en el clúster a partir del código html mediante una expresión regular
Obtener una lista con todos los enlaces absolutos de países a partir del código html mediante una expresión regular
Pasar los nombres y las frecuencias a dos listas diferentes
Aplicar la función **decode_list** a la lista de nombres de país
Devolver las listas de nombres, frecuencias y enlaces

get_Places(html):

Obtener el fragmento del código donde se encuentra la información de los lugares a partir del código html mediante una expresión regular
A partir de dicho fragmento se obtiene una lista con el enlace absoluto, nombre y código de país de cada lugar
Pasar los nombres, los códigos de lugar y los enlaces a tres listas diferentes
Aplicar la función **decode_list** tanto a la lista de nombres de lugar como a la de enlaces
Devolver las listas de nombres, códigos de país y enlaces

get_People(html, uri):

Obtener el fragmento del código donde se encuentra la información de las personas a partir del código html mediante una expresión regular
A partir de dicho fragmento se obtiene una lista con el enlace relativo, nombre y frecuencia de cada persona mediante una expresión regular
A partir de dicho fragmento también se obtiene una lista con el nombre y la frecuencia de personas que carecen de enlace
Pasar los nombres, las frecuencias y los enlaces a tres listas diferentes
Aplicar la función **decode_list** a la lista de nombres de persona
Convertir los enlaces relativos en absolutos a partir del parámetro uri
Devolver las listas de nombres, frecuencias y enlaces

get_Other(html, uri):

Obtener el fragmento del código donde se encuentra la información de los otros nombres a partir del código html mediante una expresión regular
A partir de dicho fragmento se obtiene una lista con el enlace relativo, nombre y frecuencia de cada otro nombre mediante una expresión regular
Pasar los nombres, las frecuencias y los enlaces a tres listas diferentes
En cada nombre sustituir los códigos html para caracteres no-ASCII (á) por sus respectivos caracteres (á)
Convertir los enlaces relativos en absolutos a partir del parámetro uri
Devolver las listas de nombres, frecuencias y enlaces

get_Keywords(html):

Obtener el fragmento del código donde se encuentran las palabras clave a partir del código html mediante una expresión regular
Dividir el fragmento cuando haya una coma ‘,’ o un slash ‘/’, que son los elementos que separan las keywords, y guardar el resultado en una lista
Aplicar la función **decode_list** a la lista de keywords
Devolver la lista de keywords

get_Story(html, uri):

Obtener el título y el enlace relativo de la historia a partir del código html mediante una expresión regular
Obtener la fecha de inicio de la historia a partir del código html mediante una expresión regular
Tanto en el título como en la fecha de inicio sustituir los códigos html para caracteres no-ASCII (á) por sus respectivos caracteres (á)
Convertir el enlace relativo en absolutos a partir del parámetro uri
Devolver el nombre, la fecha de inicio y el enlace de la historia

exists_Story(storyId):

Su código es el mismo que el de existe_Cluster, sólo que cambiando la consulta para buscar en la tabla STORY

insert_Story(storyId, storytitle, storystartdate, storylink):

Llamar a la función **db_Connect**
Se ejecuta la inserción de la historia con los parámetros storyId, storytitle, storystartdate y storylink
Llamar a la función **db_Disconnect**

insert_Keywords(Keywords,storyId):

Llamar a la función **db_Connect**
Por cada keyword de la lista Keywords:
{
    Se ejecuta la inserción del keyword con los parámetros keyword y storyId
}
Llamar a la función **db_Disconnect**

insert_Cluster(clusterId, title, description, link, cluster_lang, clusterDate, storyId):

Llamar a la función **db_Connect**
Se ejecuta la inserción del clúster con los parámetros clusterId, title, description, link, cluster_lang, clusterDate y storyId
Llamar a la función **db_Disconnect**

insert_Countries(Countries, countryfreqs, countrylinks, clusterId):

Llamar a la función **db_Connect**
Por cada país de la lista Countries:
{
    Se comprueba si ya existe el país en la base de datos
    Si no existe:
    {
        Se ejecuta la inserción del país con los parámetros countryname y countrylink
    }
    Se ejecuta la inserción de la relación entre país y clúster con los parámetros clusterId, countryname y countryfreq
}
Llamar a la función **db_Disconnect**

insert_People(peoplenames, peoplefreqs, peoplelinks, clusterId):

Llamar a la función **db_Connect**
Por cada persona de la lista peoplenames:
{
    Se comprueba si ya existe la persona en la base de datos
    Si no existe:
    {
        Se ejecuta la inserción de la persona con los parámetros personName y personlink
    }
    Se ejecuta la inserción de la relación entre persona y clúster con los parámetros clusterId, personName y personfreq
}
Llamar a la función **db_Disconnect**

insert_News(newstitles, newspubtimes, newslinks, clusterId):

Llamar a la función **db_Connect**
Por cada noticia de la lista newstitles:
{
    Se aplica la función **sha1** al link de la noticia para obtener su id
    Se ejecuta la inserción de la noticia con los parámetros id, title, link, pubtime y clusterId
}
Llamar a la función **db_Disconnect**

insert_Others(othernames, otherfreqs, otherlinks, clusterId):

Llamar a la función **db_Connect**
Por cada otro nombre de la lista othernames:
{
    Se comprueba si ya existe el otro nombre en la base de datos
    Si no existe:
    {
        Se ejecuta la inserción del otro nombre con los parámetros otherName y otherlink
    }
    Se ejecuta la inserción de la relación entre otro nombre y clúster con los parámetros clusterId, otherName y otherfreq
}
Llamar a la función **db_Disconnect**

insert_Places(placenames, countrycodes, placelinks, clusterId):

Llamar a la función **db_Connect**
Por cada lugar de la lista placenames:
{
    Se comprueba si ya existe el lugar en la base de datos
    Si no existe:
    {
        Se aplica la función **sha1** al link del lugar para obtener su id
        Se ejecuta la inserción del lugar con los parámetros placeId, placeName, countrycode, placelink, latitud y longitud
    }
    Se ejecuta la inserción de la relación entre lugar y clúster con los parámetros clusterId y placeId
}
Llamar a la función **db_Disconnect**

7.2.4 – Funciones auxiliares

db_Connect:

Contiene los datos de conexión a la base de datos
Se conecta a la base de datos

db_Disconnect:

Se desconecta de la base de datos.

sha1(item):

Calcula el resumen hash hexadecimal SHA1 y lo devuelve.

decode_list(list):

Decodifica los códigos html (á   etc.) de los elementos de una lista y los devuelve en otra lista.

8 – Implementación

En este apartado describiremos algunas de las decisiones tomadas en las etapas anteriores, con el fin de facilitar las cosas a cualquier persona que vaya a leer o modificar el código.

Entre esas decisiones se encuentran los módulos Perl utilizados, el método de extracción empleado, los identificadores de clúster escogidos y el tiempo de espera entre accesos al servidor.

8.1 – Módulos Perl utilizados y alternativas

LWP::Simple[23]

Descripción

Este módulo permite simular el comportamiento de un navegador web obteniendo el código html de una página web a partir de una dirección web mediante el método:

```
$html = get($uri);
```

Donde **$uri** contiene la dirección de la página web y en html se deja el código html de la página. Su funcionamiento es muy sencillo pero en todo caso resulta suficiente para el propósito de nuestro proyecto.

Alternativas

LWP::UserAgent[24]

Con este módulo podemos simular mucho mejor el comportamiento de un navegador, ya que podemos modificar el nombre del navegador que estamos utilizando, utilizar cookies, modificar cabeceras de una petición HTTP, especificar usuario y contraseña en páginas que lo requieran, elegir el método de envío de datos (GET, POST), trabajar con SSL, FTP, etc.

Como hemos dicho antes, en nuestro proyecto no necesitamos todas estas características y nos es suficiente con obtener el código HTML de las páginas.

---

[23] http://search.cpan.org/~gaas/libwww-perl-6.04/lib/LWP/Simple.pm
[24] http://search.cpan.org/~gaas/libwww-perl-6.04/lib/LWP/UserAgent.pm

WWW::Mechanize[25]

Este complemento es una subclase de LWP::UserAgent, por lo que permite utilizar todos sus métodos. No obstante, está más enfocado hacia la automatización de recuperación de contenido web. Desde enlaces y texto hasta imágenes y documentos.

Para ello cuenta con funciones más adaptadas a este menester, como volver atrás, obtener el título, seguir un enlace, obtener todos los enlaces y obtener los enlaces de todas las imágenes.

Aunque al principio elegimos este módulo por estar específicamente dirigido a la extracción de datos de sitios web, más tarde nos dimos cuenta de que podíamos hacer lo mismo con LWP::Simple.

XML::RSS

Descripción

Este módulo permite crear y actualizar ficheros RSS especificando los enlaces, los títulos y las descripciones de cada elemento o cogiéndolos de las páginas web que nos interesen. También permite estructurar en ítems cada uno con su enlace, título y descripción el contenido de un fichero RSS guardado en una cadena de caracteres.

```
$rss->parse($html);
```

En el ejemplo llamamos al método parse del objeto $rss de la clase XML::RSS, pasándole como parámetro el código de un fichero RSS contenido en la variable $html.

De ese modo obtenemos el RSS dividido por la descripción del mismo, fecha de creación y lista de elementos, cada uno con su título, descripción y enlace.

Queda claro que este módulo es el más adecuado para la descomposición de un fichero RSS pero existen otras alternativas.

Alternativas

XML::Parser

Este módulo es parecido al anterior pero permite obtener los elementos de cualquier fichero XML siempre que especifiquemos las etiquetas de lo que queremos recuperar.

[25] http://search.cpan.org/~jesse/WWW-Mechanize-1.72/lib/WWW/Mechanize.pm

Utilizarlo requiere ciertos conocimientos de XML, razón principal por la que en su momento desestimamos utilizarla en nuestro programa.

Expresiones regulares

Utilizar expresiones regulares siempre es una alternativa efectiva a tener en cuenta en la extracción de datos y el caso de los RSS no es una excepción. Resulta muy fácil extraer todos los enlaces de los clústers del RSS con una sencilla expresión regular y aunque al principio utilizamos el módulo XML::RSS por estar específicamente diseñado para ello, al final nos decantamos por las expresiones regulares.

URI[26]

Descripción

Este módulo implementa la clase URI, cuyos objetos representan "Identificadores Uniformes de Recursos" (Uniform Resource Identifier) tal y como se especifica en el RFC 2396. Es decir, una cadena de caracteres corta que identifica inequívocamente un recurso en una red o sistema.

Una dirección URI está compuesta por esquema, autoridad, ruta, consulta y fragmento.

- El **esquema** suele ser el protocolo que se utiliza para acceder al recurso. Por ejemplo: http:, ftp:, mailto:
- La **autoridad** se refiere al elemento que identifica la autoridad de nombres.
  Por ejemplo: //es.wikipedia.org
- La **ruta** a menudo identifica la ubicación de un recurso dentro de la máquina.
  Por ejemplo: /wiki/Uniform_Resource_Identifier
- Después de la ruta, se pueden indicar los valores de elementos de un formulario (método GET) separados entre sí por el carácter & y que por lo general son pares "clave=valor". A este componente se le llama **consulta** y viene precedido por el carácter '?'.
- El **fragmento** permite identificar una vista de una representación del recurso principal. Su comienzo se indica con el carácter '#'.

La clase dispone de métodos para comparar direcciones URI, convertir direcciones relativas en absolutas, obtener el directorio actual y obtener cada uno de los componentes antes mencionados de una dirección URI, entre otros.

---

[26] http://es.wikipedia.org/wiki/Uniform_Resource_Identifier
http://search.cpan.org/~gaas/URI-1.60/URI.pm

En nuestro caso, utilizamos la operación de convertir una dirección web relativa en absoluta indicando una dirección base.

```
$enlace_absoluto = URI->new($enlace_relativo)->abs($enlace_base);
```

HTML::Entities[27]

Descripción

Este módulo nos permite codificar y decodificar cadenas de caracteres en el formato que se utiliza en HTML. Por ejemplo, si decodificamos la cadena 'descripción' obtendremos la cadena 'descripción'.

En nuestro caso lo necesitamos porque trabajamos directamente con el código HTML y queremos guardar el texto decodificado.

DBI[28]

Descripción

Las siglas DBI son el acrónimo de Database Independent Interface for Perl, o lo que es lo mismo a decir que el módulo funciona como una interfaz entre Perl y el driver de bases de datos que utilicemos.

Esto quiere decir que no nos tenemos que preocupar de las especificidades de cada motor de bases de datos, sino simplemente de realizar la conexión y trabajar sobre nuestras tablas.

En nuestro programa lo utilizamos constantemente tanto para comprobar si el clúster actual ya está almacenado en la base de datos, como para insertar o actualizar registros de las diferentes tablas.

Digest::SHA1

Descripción

[27] http://search.cpan.org/~gaas/HTML-Parser-3.69/lib/HTML/Entities.pm
[28] http://search.cpan.org/~timb/DBI-1.622/DBI.pm

La clase Digest implementa los métodos más conocidos para calcular resúmenes de datos. Generalmente estos resúmenes son utilizados para comprobar la integridad de los datos a posteriori. Los más conocidos son MD5 y SHA1.

Sus aplicaciones van desde codificar contraseñas para que no sean interceptadas por la red, como método de comprobación de integridad de ficheros descargados por Internet, como función hash que sirve como clave en bases de datos, tablas hash y ficheros XML. Este último es el uso que le hemos dado al módulo en nuestro programa. En concreto hemos creado claves primarias para cada tabla a partir de un enlace u otro elemento que a priori sea único para cada registro de la misma.

Por ejemplo: EMMCLUSTER.Id = SHA1(EMMCLUSTER.link)

La función SHA1 crea un resumen de 40 caracteres hexadecimales del fichero o la cadena de caracteres que reciba. Como consecuencia, todas las claves primarias tienen longitud fija y tienen un código que no se repite en elementos diferentes.

8.2 – Método de extracción de datos

Expresiones regulares

Como hemos comentado antes, las expresiones regulares son uno de los más importantes métodos de extracción de datos. Son muy eficientes y nos permiten extraer exactamente lo que buscamos. Ya sea aplicando una expresión regular a todo el texto o reduciendo el fragmento en el que buscar, lo cierto es que en muchos casos resultan una opción más acertada que algunos módulos diseñados para extraer información.

No obstante también existen algunos problemas asociados al uso de esta herramienta. El primero es la curva de aprendizaje. A diferencia de otros métodos más intuitivos, si no se ha trabajado antes con expresiones regulares y autómatas puede costar un tiempo entender su funcionamiento, aprender a leer expresiones y a plasmar una idea mediante expresiones regulares.

El segundo problema y quizás el más importante es que las páginas web de hoy en día suelen actualizarse muy a menudo. El contenido de la mayoría se actualiza a diario y generalmente no pasan tres años sin que el webmaster modifique su estructura.

Esto quiere decir que si utilizamos expresiones regulares en un programa que vamos a utilizar durante mucho tiempo, es necesario estar al tanto de las actualizaciones en la estructura de la página.

Por suerte para nosotros, el EMM News Explorer tiene una estructura bien organizada que si bien su contenido es actualizado a diario, es muy probable que su estructura interna no sufra cambios a medio plazo, ni tampoco la forma de mostrar esos datos.

HTML::TreeBuilder[29]

Este módulo es uno de los más utilizados para extraer información de una página web.

Representa los documentos HTML en forma de árbol, de modo que resulta muy fácil navegar a través de él. Los árboles son una buena manera de representar código HTML, un ejemplo claro de ello es que el elemento <head> es hijo de <html>, mientras que el elemento <title> es hijo de <head>.

Un objeto TreeBuilder coge un código html de una variable o un fichero y lo convierte en un árbol de nodos HTML::Element. A cada nodo se le puede preguntar por su padre, sus hermanos y sus hijos. Además en el caso de los hijos se puede limitar la consulta a los que cumplan ciertas restricciones. Por ejemplo podríamos consultar sólo los enlaces que tengan un determinado valor en el atributo class:

```
my $tree = HTML::TreeBuilder->new_from_content( $page );

my ($pubinfo) = $tree->look_down(
                                    _tag => 'a',
                                    class => 'books'
    );
```

Aunque pueda parecer la mejor solución, lo cierto es que las facilidades que ofrece este módulo tienen un coste: el tiempo de ejecución. Mientras que con las expresiones regulares trabajamos directamente con el texto de un fichero, con la clase TreeBuilder tenemos que analizar todo el documento, crear la estructura en árbol y guardar cada elemento en un nodo.

Si sólo analizásemos un par de páginas esta sería la solución adecuada, pero da la casualidad de que a priori comprobamos unos 360 enlaces cada día (unos 60 enlaces del rss en cada uno de los 6 idiomas) además de los clústers relacionados. Si tomamos como premisa que recogemos de media un clúster relacionado por cada clúster de un RSS, nuestro número se duplica, pero lo cierto es que el promedio de clústers relacionados que recogemos es mayor.

Por último, en las primeras iteraciones se hizo una comparación de tiempos de ejecución entre la extracción de clústers mediante expresiones regulares y por medio de árboles y comprobamos que utilizando los últimos el programa tardaba tres veces más.

[29] http://search.cpan.org/~cjm/HTML-Tree-5.03/lib/HTML/TreeBuilder.pm

HTML::TokeParser[30]

Este módulo a diferencia del anterior nos permite navegar por el código HTML almacenando únicamente el contenido de los nodos visitados a medida que nos acercamos a la información que queremos extraer.

Si quisiéramos extraer todos los enlaces de una página pondríamos algo así:

```
use HTML::TokeParser;
$p = HTML::TokeParser->new(shift);
while (my $token = $p->get_tag("a")) {
      my $url = $token->[1]{href};
      my $text = $p->get_trimmed_text("/a");
      print "$url\t$text\n";
}
```

En un principio estuvimos tentados a utilizar este módulo por su facilidad de uso, pero entonces descubrimos la flexibilidad de las expresiones regulares y decidimos sacrificar la legibilidad del código en pos de un mayor control en la extracción de datos.

8.3 – Estructuras de datos

Del mismo modo que en lugar de utilizar una representación en árbol de las páginas HTML hemos utilizado expresiones regulares para agilizar la ejecución de nuestro programa, a la hora de guardar temporalmente los datos extraídos en memoria, hemos preferido utilizar listas en lugar de estructuras más complejas.

En este caso la velocidad de ejecución no ha sido tan determinante como la facilidad de uso que ofrece Perl con las listas. Perl implementa las listas como conjuntos de elementos y nos provee de funciones para trabajar con ellas como si fueran pilas, colas o vectores.

8.4 – Tiempo de espera entre accesos

Al simular el comportamiento de un navegador accediendo a las páginas de un servidor, debemos tener en cuenta la carga que supone nuestro programa para los recursos de ese servidor. Si lo sobrecargamos con muchas peticiones en un corto período de tiempo, estamos impidiendo el acceso a otras personas y lo más probable es que su administrador se ponga en contacto con nosotros para llamarnos la atención.

[30] http://search.cpan.org/~gaas/HTML-Parser-3.69/lib/HTML/TokeParser.pm

Una buena práctica en el mundo del Web Scraping es pausar el programa unos segundos entre acceso y acceso. De este modo nuestro robot puede seguir descargando contenido y no asfixiamos los recursos del servidor.

8.5 – Ficheros del programa

Hemos dividido nuestro programa en tres ficheros:

1. **NewsMonitor.pm:** Es un módulo Perl que contiene la definición de todas las funciones utilizadas en el programa principal además de los contadores de registros nuevos insertados y la importación de los módulos antes descritos.
2. **Emm.pl:** Contiene el programa principal y está enlazado únicamente con el fichero NewsMonitor.pm para poder utilizar las funciones que allí se definen.
3. **Emm_launcher.pl:** Este programa se encarga de ejecutar una tras otra las diferentes configuraciones del programa estableciendo los parámetros de entrada. En este momento el programa se ejecuta comprobando seis idiomas (inglés, español, francés, italiano, alemán y holandés) y con un umbral de semejanza de 0,5. Los idiomas se van rotando de modo que en cada una de las seis ejecuciones sea distinto el lenguaje principal.

9 – Pruebas

Las pruebas son un componente fundamental en todo proceso de desarrollo de software. Su finalidad, más allá de ser la de comprobar que todo funciona correctamente, es la de descubrir errores. Cuantos más errores descubramos en esta fase, menos problemas tendremos una vez instalemos el programa en el ordenador del cliente.

Para descubrir dichos errores se diseñan casos de prueba, cuya idoneidad viene determinada por la probabilidad que éstos nos brindan de encontrar un nuevo error.

Existen varios tipos de prueba que podemos realizar. Las más importantes son: las pruebas unitarias, las de integración, las de sistema y las de aceptación. Pero no son las únicas utilizadas, en algunos entornos es bueno realizar también pruebas de instalación y de estrés.

Dicho esto, también tenemos que aclarar que no todos los sistemas son iguales y puede que los tipos de prueba que se realizan en unos casos, en otros casos resulten inútiles.

9.1 – Pruebas Unitarias

En estas pruebas, las comprobaciones se dividen por componentes, que pueden ser desde casos de uso hasta funciones. En algunos sistemas, como en el nuestro, esta división es clave a la hora de determinar el origen de los errores. Es más fácil encontrar un error en 20 líneas de código que en 200.

En nuestro caso, hemos elegido realizar pruebas unitarias a cada función antes de incluirla en el programa principal. Las pruebas realizadas han sido sobre todo de caja negra, es decir, validamos si el comportamiento de cada función es el de su especificación.

Pero además, hemos tenido en cuenta el único punto del programa en el que el usuario puede introducir datos: los parámetros de entrada.

En el contrato de la operación que comprueba los parámetros, check_Params, especificamos que los idiomas especificados tenían que ser cadenas de 2 caracteres y que la semejanza que le especificásemos tendría que ser un valor entre 0 y 1. Si alguna de estas condiciones no se cumplía, el programa terminaba con un mensaje de error.

A continuación se describen los casos probados que nos han dado otro resultado al que en principio esperábamos.

9.1.1 – Caso de prueba Idioma Principal

En este caso de prueba, comprobamos los resultados que obtenemos de la función process_Rss, si le pasamos cadenas de dos caracteres que han pasado la comprobación realizada en check_Params, pero no se encuentran entre los idiomas contemplados por el EMM News Explorer. La función process_Rss recibe una cadena de dos caracteres, accede al canal RSS en dicho idioma, y devuelve la lista de enlaces a clústers encontrada en el canal.
Variable: $INlang (Lengua Principal)
Función: process_Rss

Representantes	Resultado esperado	Resultado obtenido
en	Lista de clústers que aparecen en el canal RSS en inglés.	Lista de clústers que aparecen en el canal RSS en inglés.
1a, b2, ?a	Lista vacía.	Lista vacía
&&	Lista vacía.	Error Perl: La sintaxis del comando no es correcta.
??, $?	Lista vacía.	Error al obtener el canal RSS.
&a, .a,	Lista vacía.	Error Perl: "a" no se reconoce como un comando, programa o archivo por lotes ejecutable.

Tabla 2: Caso de Prueba Idioma Principal

Solución: Añadir restricción en check_Params, que compruebe que la cadena de caracteres $INlang correspondiente al idioma principal está compuesta por dos caracteres alfabéticos en minúscula.

9.1.2 – Caso de prueba Idiomas Secundarios

En este caso de prueba comprobamos la función get_Related, que es donde se utilizan los parámetros de programa @OUTlangs (Idiomas secundarios). Las comprobaciones que se realizan aquí son similares a las del caso de prueba anterior, con una diferencia: en el caso anterior, los errores se daban porque el parámetro aparecía en el enlace de un canal RSS que teníamos que obtener. En este caso, sin embargo, los parámetros aparecen dentro de una expresión regular. Esto quiere decir que tenemos que probar los casos límite que modifiquen el efecto de dicha expresión. Esta función recibe como parámetros una dirección web, una cadena de caracteres con el código html de dicha dirección, el umbral de semejanza y una lista de idiomas. Se devuelven tres listas: una lista de enlaces, una lista de idiomas y una lista de semejanzas. Para esta prueba hemos introducido datos de clústers determinados de modo que todos los errores estén producidos por los idiomas secundarios.

Variables: @OUTlangs
Función: get_Related

Representantes	Resultado esperado	Resultado Obtenido
es, en, it, fr, …	Listas de clústers relacionados encontrados en cada idioma.	Lista de clústers relacionados en cada lenguaje
a1, bj, ri	Listas vacías	Listas vacías
$a, @b	Listas vacías	La expresión regular interpreta que tiene que utilizar la variable 'a' o la lista 'b', que no están definidas. Termina con un error.
..	Listas vacías	El punto '.' en una expresión regular representa cualquier carácter excepto el salto de línea. La expresión regular recopila todas las combinaciones de dos caracteres y devuelve los clústers relacionados en todos los idiomas.

Tabla 3: Caso de prueba Idiomas Secundarios

Hemos aplicado la misma solución que en el caso de prueba anterior.

9.1.3 – Caso de prueba Umbral de Semejanza

Por último hemos puesto a prueba el parámetro umbral de semejanza, que hemos definido como un valor entre 0 y 1. Al realizar las pruebas hemos detectado un error muy importante: no hemos comprobado si el parámetro de entrada es un número real, por lo que de no serlo, al realizar la comparación el programa se cerrará con un error.

Variables: $similarity (Umbral de semejanza)
Función: check_Params

Representantes	Resultado esperado	Resultado Obtenido
a, "hola", …	No se contemplaba	La ejecución termina con un error.

Tabla 4: Caso de prueba Umbral de Semejanza

En este caso hemos realizado una comprobación adicional en la función check_Params creando la función is_Num, que determina mediante una expresión regular si el último parámetro pasado al programa tiene la estructura de un número real.

9.2 – Pruebas de Integración

Estas pruebas se suelen realizar una vez comprobado cada componente por separado y los errores que descubren están relacionados con las dependencias existentes entre módulos.

En nuestro programa hemos vuelto a esta etapa cada vez que terminamos una nueva función, y los errores obtenidos nos han ayudado a determinar los puntos críticos de la aplicación.

Ejemplos de errores encontrados son los relacionados con obtener y almacenar una lista de elementos de un clúster, como las palabras clave. Aunque por separado las funciones get_Keywords e insert_Keywords funcionaban bien, en algunos casos aislados, lo que devolvía la primera daba error al pasárselo a la segunda.

Uno de estos errores sucedió al intentar extraer las keywords de una página sin palabras clave. La función get_Keywords devolvía caracteres extraños que al pasárselos a insert_Keywords los insertaba en la base de datos.

La solución en la mayoría de los casos fue la misma: añadir un caso básico que devolviese una lista vacía en los casos en los que no hubiese keywords.

9.3 – Pruebas de Sistema

Estas pruebas se realizan una vez hemos integrado todas las funciones en nuestro sistema y queremos simular el comportamiento en un entorno de producción. Descubrir errores en esta fase resulta de suma importancia porque evita que le sucedan al cliente una vez instalemos nuestro sistema en su máquina.

En nuestro caso se han realizado las pruebas en un PC con Linux, un sistema gestor de bases de datos mysql y el intérprete Perl instalados. Para realizar las pruebas se han ejecutado varias configuraciones diferentes del mismo programa y se han analizado los datos almacenados en la base de datos comparándolos con los que había en las páginas de clústers en las que se ha mirado.

9.4 – Pruebas de Explotación

Estas pruebas se realizan una vez que el sistema está en funcionamiento en el ordenador del cliente. En nuestro caso, el reloj del servidor en el que lo hemos instalado ejecuta una vez al día el fichero emm_launcher.pl, que a su vez ejecuta varias configuraciones diferentes del programa con diferentes idiomas principales y secundarios. Los datos se almacenan en una base de datos de un servidor mysql que está accesible mediante la herramienta phpMyAdmin.

Algunos de los problemas encontrados en esta fase son los siguientes:

Al exportar la base de datos a otra máquina, mysql informaba de un error diciendo que en la tabla PERSON había claves primarias repetidas.

Aún no sabemos cuál es la causa, porque nuestro programa comprueba antes de intentar insertar un registro, si éste ya existe. Además, la columna NAME de la tabla PERSON está definida como PRIMARY KEY. Es decir, en cada tupla, el valor de la columna es único y obligatorio y funciona como índice para las búsquedas.

Si intentásemos insertar un nombre que ya existe en la tabla, mysql nos devolvería un error. Pero no lo hace porque como hemos dicho, antes de insertar un nombre comprobamos si existe o no.

Conseguimos atajar el problema cambiando el tipo de datos de la columna NAME de utf8-general-ci a utf8-bin.

Otro problema encontrado es que pueden existir dos personas diferentes con el mismo nombre. En estos casos nuestro sistema interpreta que son la misma persona, y si ya hemos registrado el nombre una vez, las siguientes veces no lo tendremos en cuenta.

La solución pasaría por elegir como clave primaria el enlace a la página de persona en lugar del nombre. El problema radica en que no todas las personas tienen una página propia ni un enlace, por lo que de momento hemos decidido dejarlo para futuras versiones.

En anteriores apartados hemos comentado otras dos pruebas realizadas a nuestro sistema. La primera está relacionada con el tiempo de ejecución del programa utilizando expresiones regulares frente a estructuras más complejas como árboles HTML. La segunda tuvo como objetivo determinar el tamaño máximo utilizado en cada columna de cada tabla, con el fin de reducir su tamaño máximo y ahorrar espacio en disco. En ambas ocasiones se realizó la comprobación mediante sendos scripts escritos en Perl.

10 – Implantación

En este apartado describimos los pasos que hemos seguido para instalar el programa en el servidor. Asimismo, también mostramos los resultados obtenidos desde que implantamos el sistema.

10.1 – Manual de Instalación

10.1.1 – Requisitos

Ficheros

Para realizar la instalación del programa se requieren cuatro ficheros:

- **emm_clusters.sql:** Contiene las instrucciones SQL de creación de la base de datos junto con sus tablas, columnas relaciones y restricciones.
- **NewsMonitor.pm:** Módulo Perl que guarda todas las funciones utilizadas en el programa, junto con la importación de todos los módulos requeridos.
- **emm.pl:** Contiene el programa principal e importa el fichero NewsMonitor.pm para poder hacer uso de las funciones que en él se definen.
- **emm_launcher.pl:** Redirecciona la salida estándar al fichero log /var/log/emm.log y realiza tantas ejecuciones seguidas del programa emm.pl como idiomas se definan en la lista @langs. El comando recibe como parámetro el umbral de semejanza 0.5, pero se puede cambiar en función de nuestras necesidades.

Intérprete Perl

Del mismo modo, es necesario tener instalada la última versión de Perl y los cinco módulos que utiliza nuestro programa.

- LWP::Simple: Este módulo se encarga de obtener el código HTML de una página web a partir de su enlace.
- URI: Utilizamos este módulo para convertir direcciones web relativas en absolutas.
- HTML::Entities: Gracias a este módulo podemos decodificar las secuencias de escape HTML como “ ”, “á”, etc.
- DBI: Este módulo sirve como puente entre nuestro programa y el driver mysql con el fin de conectarnos e interactuar con nuestra base de datos.
- Digest::SHA1: Nuestro programa realiza un resumen SHA1 hexadecimal del enlace de cada clúster y lo utiliza como clave primaria en su respectiva tabla.

Servidor mysql

También es necesario tener instalado un servidor de bases de datos myql en alguna máquina accesible desde el ordenador que vaya a ejecutar el programa.

10.1.2 – Pasos a seguir

1- Instalar Perl y sus módulos en el cliente.
2- Instalar servidor mysql y crear la estructura de la base de datos en el servidor.
3- Copiar programa en el sistema de archivos del cliente y darle permisos de ejecución.
4- Cambiar los datos de conexión a la base de datos por los datos reales del servidor.
5- Programar cron para que ejecute el programa automáticamente.

Instalar Perl

Como vamos a instalar el programa Perl en un ordenador con Sistema Operativo Linux, podemos descargar el **intérprete Perl** del repositorio principal de nuestra distribución. También debemos instalar un componente llamado **CPAN**, que nos permite descargar e instalar todos los módulos Perl que necesitamos.

Para instalar los módulos desde el cliente CPAN es necesario tener privilegios de administrador, por lo que en una nueva terminal de consola ejecutamos:

>sudo cpan

Una vez introducida la contraseña del superusuario aparecerá una linea de comandos con prefijo "cpan>" en la que tendremos que ir introduciendo las siguientes instrucciones a medida que vayan terminando:

- install LWP::Simple
- install URI
- install HTML::Entities
- install DBI
- install Digest::SHA1

Si hay algún problema siempre podemos descargar los módulos del repositorio Linux o probar en CPAN los comandos force install o fforce install.

Instalar la Base de Datos

El servidor mysql también se encuentra en la mayoría de repositorios Linux por lo que lo podemos descargar e instalar sin mayor problema.

Para crear la estructura de la base de datos, desde el ordenador donde tenemos instalado el servidor mysql ejecutamos la interfaz de línea de comandos mysql, que también requiere permisos de administrador:

>sudo mysql

mysql>source ruta_del_fichero

Donde "mysql>" es el prompt de la interfaz de línea de comandos mysql y ruta_del_fichero es una cadena de caracteres que representa el camino absoluto para llegar al fichero emm_clusters.sql.

Por ejemplo:

source /home/usuario/Desktop/emm_clusters.sql

Copiar programa en el cliente

Los programas que se deben copiar en la máquina en la que hemos instalado el intérprete Perl son NewsMonitor.pm, emm.pl y emm_launcher.pl. Además les daremos permisos de ejecución con el comando:

sudo chmod +x nombre_del_fichero

Cambiar los datos de conexión con la base de datos

Al principio del fichero NewsMonitor.pm tenemos definida una función llamada db_Connect. En ella debemos cambiar los datos de conexión a nuestra base de datos.

En la variable host pondremos la IP de la máquina en la que hemos instalado el servidor mysql, o en caso de que sea la misma que el cliente, dejaremos su valor como "localhost". Del mismo modo tenemos que especificar el puerto, el usuario y la contraseña de un usuario con permisos para insertar, eliminar y actualizar los contenidos de la base de datos EMM_CLUSTERS.

Programar cron para que ejecute el programa automáticamente

Abrimos crontab en un editor mediante el comando:

>crontab –e

Añadimos una línea por cada ejecución que queramos realizar.

Formato:

- **Minuto =** El minuto de 00 a 59. Un * indica cada minuto.
- **Hora =** Hora del día en formato de 24 horas: de 00 a 23. Un * indica cada hora.
- **Día =** Día del mes de 1 a 31. Un * indica cada día.
- **Mes =** Mes del año de 1 a 12. Un * indica cada mes.
- **Día de la semana =** 3 caracteres en inglés (sun, mon,..) o númerico empezando el domingo por 0 (0=domingo, 1=lunes,...). Un * indica cada día de la semana.
- **Comando =** El comando que queremos que se ejecute.

Ejemplo:
0 14 * * * perl -w /home/usuario/Desktop/emm_launcher.pl
0 15 * * * perl -w /home/usuario/Desktop/emm_launcher.pl
0 16 * * * perl -w /home/usuario/Desktop/emm_launcher.pl

Estas tres líneas ejecutarían a las 14:00, a las 15:00 y a las 16:00 de todos los días el lanzador del programa emm.pl, que a su vez se ejecuta tantas veces como idiomas se hayan especificado en el fichero emm_launcher.pl.

10.2 – Resultados obtenidos a 19 de octubre de 2012

Nuestro programa lleva recopilando clústers del EMM News Explorer en seis idiomas desde el 31 de enero de 2012 hasta el 19 de octubre de 2012. Es decir, en 262 días.

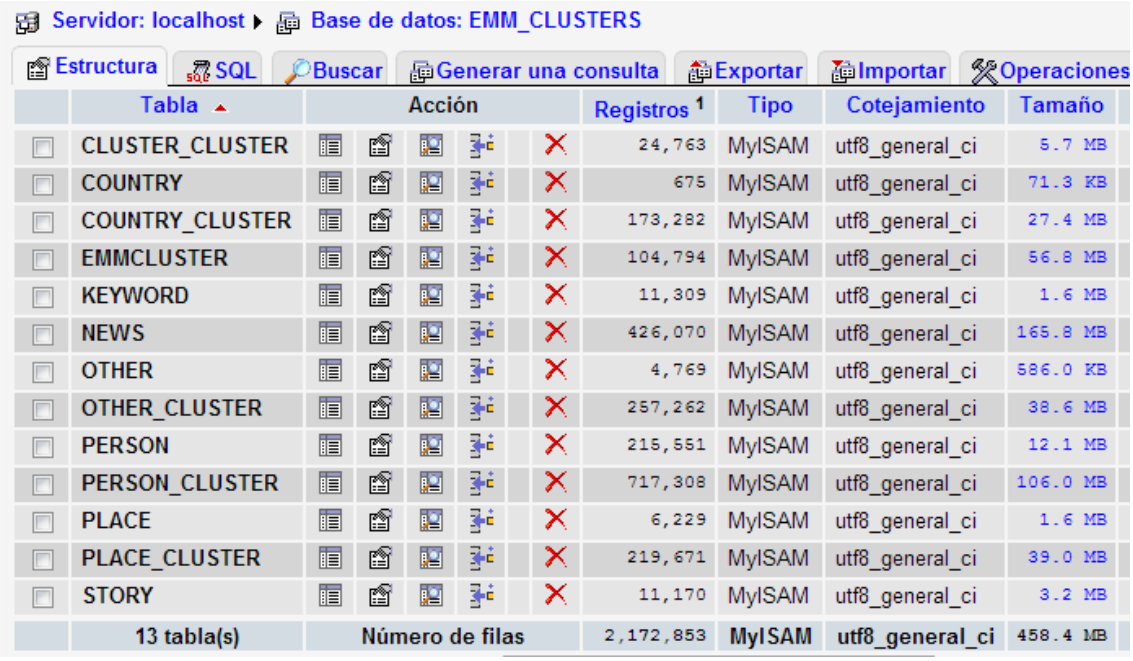
Hasta el momento ha obtenido 104.794 clústers, 426.070 noticias y 11.170 historias, a parte de países, lugares, personas y otros nombres. En la actualidad tiene un tamaño de unos 500 MB.

Si interpolamos estos resultados en el tiempo transcurrido, obtenemos que el programa ha extraído unos 400 clústers, 1626 noticias y 43 historias cada día, ocupando 1954 KB.

Del mismo modo, podemos extrapolar estos números a un año y predecir cuantos registros esperamos tener en ese tiempo. Si el número de registros crece de manera lineal, podemos asegurar que el 31 de enero de 2013 la base de datos tendrá 146.400 clústers, 595.116 noticias y 15.738 historias y ocupará 699 MB.

Asimismo, si el programa sigue funcionando como hasta ahora, dentro de un año es muy probable que en la base de datos tengamos 250.794 clústers, 1.019.560 noticias y 26.865 historias. Lo que ocupará 1 GB y 173MB más o menos.

En la Figura 23 se muestran los registros que tiene cada tabla, junto con el espacio que ocupan en el disco duro.


Servidor: localhost ▸ Base de datos: EMM_CLUSTERS

Estructura | SQL | Buscar | Generar una consulta | Exportar | Importar | Operaciones

Tabla	Acción	Registros [1]	Tipo	Cotejamiento	Tamaño
CLUSTER_CLUSTER		24,763	MyISAM	utf8_general_ci	5.7 MB
COUNTRY		675	MyISAM	utf8_general_ci	71.3 KB
COUNTRY_CLUSTER		173,282	MyISAM	utf8_general_ci	27.4 MB
EMMCLUSTER		104,794	MyISAM	utf8_general_ci	56.8 MB
KEYWORD		11,309	MyISAM	utf8_general_ci	1.6 MB
NEWS		426,070	MyISAM	utf8_general_ci	165.8 MB
OTHER		4,769	MyISAM	utf8_general_ci	586.0 KB
OTHER_CLUSTER		257,262	MyISAM	utf8_general_ci	38.6 MB
PERSON		215,551	MyISAM	utf8_general_ci	12.1 MB
PERSON_CLUSTER		717,308	MyISAM	utf8_general_ci	106.0 MB
PLACE		6,229	MyISAM	utf8_general_ci	1.6 MB
PLACE_CLUSTER		219,671	MyISAM	utf8_general_ci	39.0 MB
STORY		11,170	MyISAM	utf8_general_ci	3.2 MB
13 tabla(s)	Número de filas	2,172,853	MyISAM	utf8_general_ci	458.4 MB

Figura 23: Resultados obtenidos

11 – Gestión

En este apartado presentaremos la información más relevante de la gestión realizada en el proyecto. Además, dejaremos constancia de las incidencias principales que nos han hecho re-planificar el proyecto y daremos una justificación razonada de dichos cambios en la planificación.

11.1 – Procesos Tácticos

En la Tabla 5 mostramos una comparativa por procesos tácticos del tiempo estimado y el tiempo empleado.

Proceso	Esfuerzo Estimado	Esfuerzo Real
Planificación	20h	25h
Gestión	10h	10h
Reuniones	15h	20h
Instalación de herramientas	5h	5h
TOTAL:	**50h**	**60h**

Tabla 5: Comparativa Planificado VS Real - Procesos Tácticos

Como podemos ver si comparamos los gráficos Gráfico 1 y Gráfico 2, no se aprecia gran diferencia entre la proporción de tiempo asignado a cada proceso y la proporción de tiempo utilizado. Esto quiere decir que aunque al final se dedicaran más horas, se hizo una estimación realista del tiempo de dedicación a cada proceso. Por otra parte, aunque una diferencia de 10 horas entre el esfuerzo estimado y el real no parezca mucho, en realidad estamos hablando de una desviación del 20%.

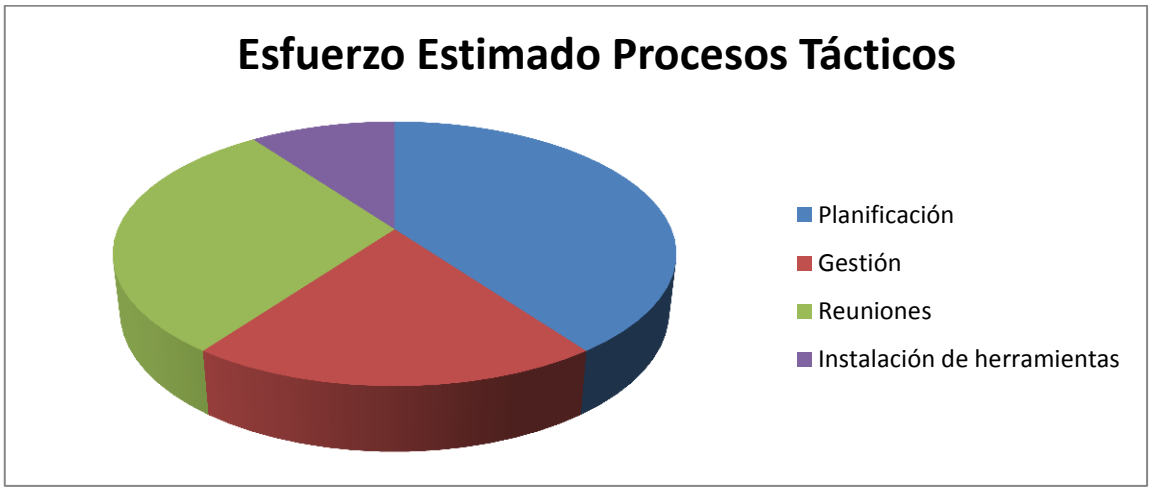


Gráfico 1: Esfuerzo estimado en los procesos tácticos

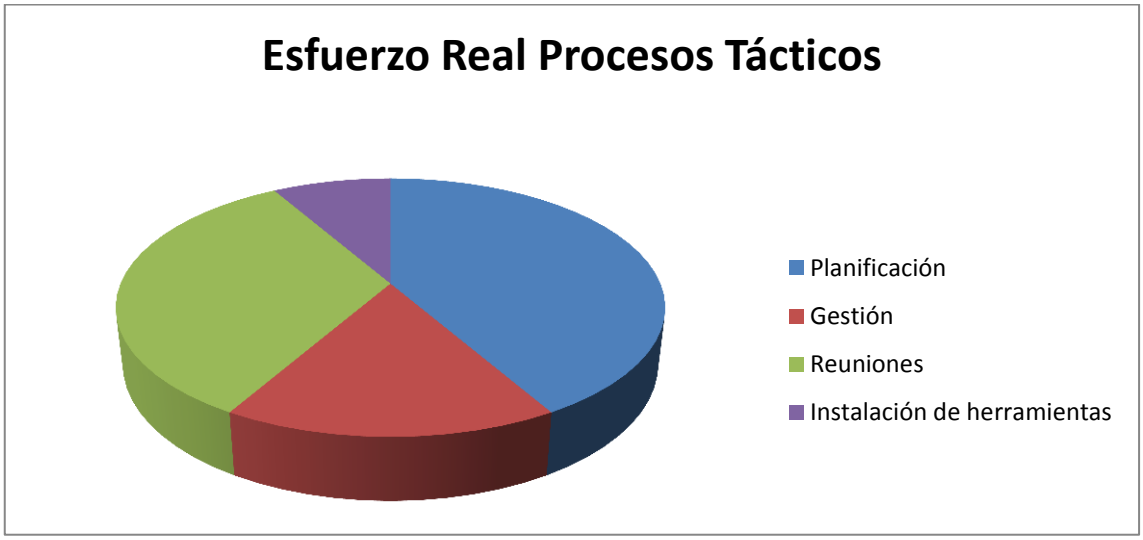


Gráfico 2: Esfuerzo real en los procesos tácticos

11.2 – Procesos Formativos

En la Tabla 6 se muestra la diferencia entre el esfuerzo estimado y el esfuerzo real en los procesos formativos. Como se puede apreciar, existe una gran diferencia entre la planificación de horas a invertir en la redacción de la memoria y el tiempo real empleado. Esto es debido a varios factores, que explicaremos al final del apartado.

Proceso	Esfuerzo Estimado	Esfuerzo Real
News Explorer	15h	15h
Perl	10h	10h
Expresiones Regulares	5h	5h
Memoria	80h	115h
Presentación	5h	-
TOTAL:	**115h**	**145h**

Tabla 6: Comparativa Planificado VS Real - Procesos Formativos

En este caso los gráficos Gráfico 3 y Gráfico 4 nos muestran que aunque se estimó correctamente que la memoria sería el proceso formativo que más carga tendría en el proyecto, realmente se ha dedicado bastante más tiempo a realizarla de lo que al principio se pensó. Esta vez vemos que la diferencia de 30 horas entre esfuerzo estimado y real se traduce en una desviación del 26%.

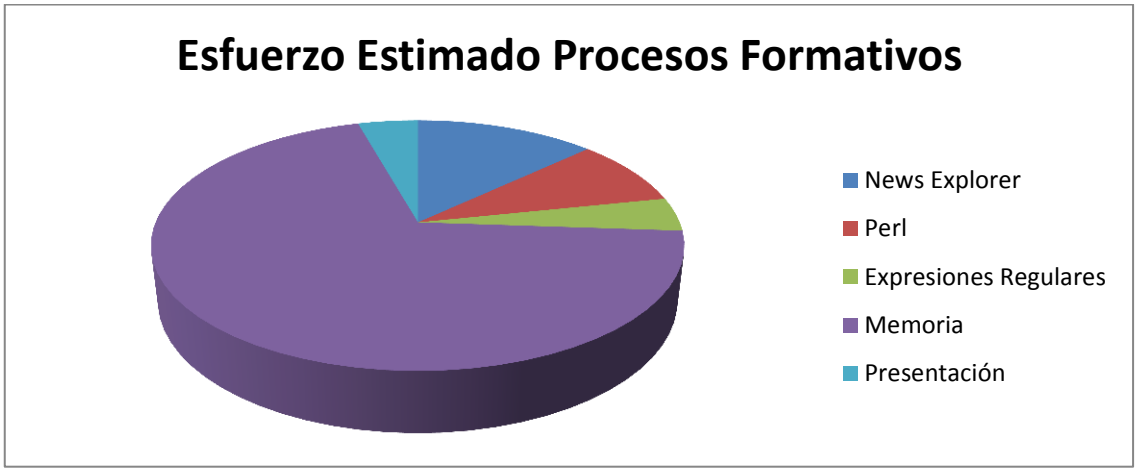


Gráfico 3: Esfuerzo estimado en los procesos formativos

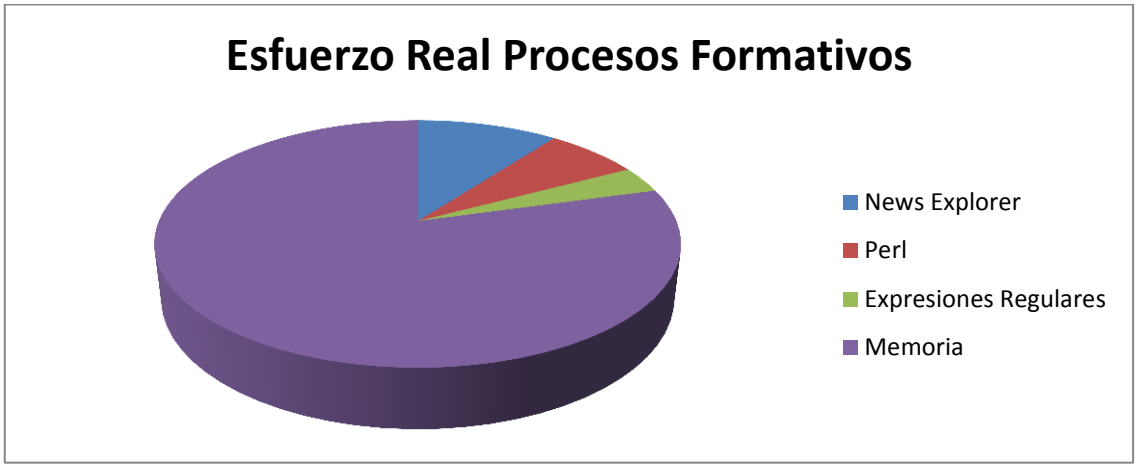


Gráfico 4: Esfuerzo real en los procesos formativos

11.3 – Procesos Operativos

Como podemos ver en la Tabla 7, si comparamos los totales no se aprecia un desajuste muy grande. Pero si miramos proceso a proceso vemos que en realidad he utilizado más tiempo que el planificado en las primeras fases y casi la mitad de tiempo en las Pruebas. Esto es debido a que tuvimos que añadir una iteración más al proyecto, que se corresponde con la última versión del programa.

Proceso	Esfuerzo Estimado	Esfuerzo Real
Captura de Requisitos	13h	15h
Análisis	7h	15h
Diseño	13h	15h
Implementación	28h	30h
Pruebas	19h	10h
TOTAL:	**80h**	**85h**

Tabla 7: Comparativa por procesos Planificado VS Real - Procesos Operativos

En los gráficos 5 y 6 se observa que donde más cambio ha habido entre la planificación y el esfuerzo real es en el análisis. Esto se debe principalmente a que en una ocasión se tuvieron que rehacer prácticamente todos los contratos de las operaciones y los diagramas de secuencia del sistema. También se observa que se ha dedicado relativamente menos tiempo al proceso de Pruebas que el que se esperaba.

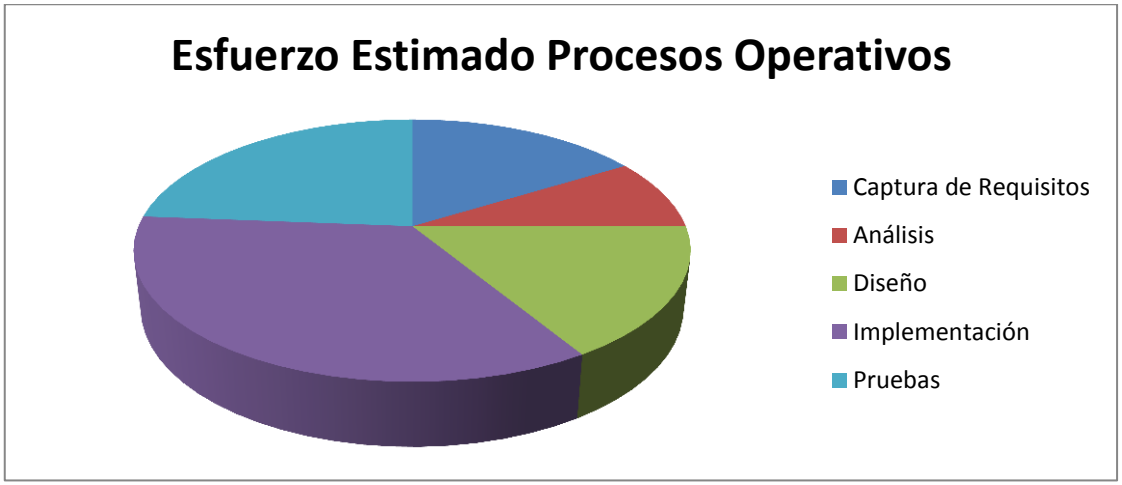


Gráfico 5: Esfuerzo estimado en los procesos operativos

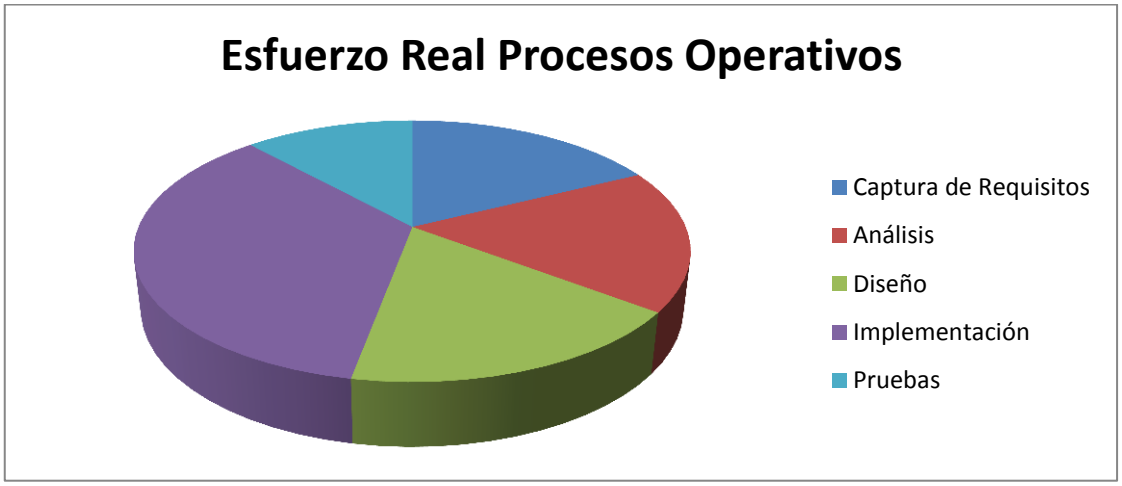


Gráfico 6: Esfuerzo real en los procesos operativos

En la Tabla 8 mostramos la diferencia de dedicación entre iteraciones. También aparece la séptima y última iteración, relacionada con reordenar el código fuente y añadir algunas comprobaciones que antes no se hacían.

Iteración	Esfuerzo Estimado	Esfuerzo Real
Primera	2h	2h
Segunda	2h	3h
Tercera	40h	35h
Cuarta	6h	10h
Quinta	18h	20h
Sexta	12h	10h
Séptima	-	5h
TOTAL:	**80h**	**85h**

Tabla 8: Comparativa por iteraciones Planificado VS Real - Procesos Operativos

En el Gráfico 7 podemos comprobar que asignamos la mitad del tiempo de desarrollo del proyecto a la tercera iteración, relacionada con la extracción de la información. Aunque ha sido la iteración más larga del proyecto, el Gráfico 8 muestra que no hemos necesitado todo el tiempo que al principio se estimó.

También se puede ver que utilizamos más tiempo que el estimado para completar la cuarta iteración. Como hemos dicho antes esto es debido a problemas en los contratos y los diagramas de secuencia del sistema.

Por último, hacia el final se realizó una última iteración relacionada con reordenar el código y añadir algunas comprobaciones adicionales de los parámetros de entrada.

Gráfico 7: Esfuerzo estimado por iteraciones

Gráfico 8: Esfuerzo real por iteraciones

Resumiendo, en la planificación inicial se estimó que en total se necesitarían alrededor de 245 horas y en realidad se han necesitado 290 horas. Es decir, se ha producido una desviación del 18%.

Algo curioso que no reflejan los gráficos es que entre las iteraciones 6 y 7 pasaron ocho meses. La explicación de ello se describe en el siguiente apartado.

11.4 – Justificación de las desviaciones

La mayor desviación del proyecto ha sido en la entrega de la memoria. Por una parte he empleado mucho más tiempo del que pensaba que necesitaría para redactarla. Por otra parte, tenía intención de entregarla en Enero del 2012 y la voy a entregar en Octubre del mismo año.

En el primer caso, la razón de la desviación ha sido una mala planificación. En parte se debe a que ésta es la primera memoria que he tenido que redactar.

En el segundo caso, tuvieron que ver varios factores:

1. Entre Septiembre del 2011 y Febrero del 2012 he estado asistiendo a clases on-line sobre Ingeniería Informática que me han restado tiempo para emplear en otras cosas. No obstante considero que los conocimientos adquiridos me serán de gran ayuda en un futuro cercano.
2. Desde Marzo hasta Junio he estado estudiando inglés en una academia. Proceso que culminé con la obtención del FCE.
3. En verano he estado trabajando y no le he dedicado mucho tiempo al proyecto.
4. En la mayoría de los casos no he tenido en cuenta el tiempo que tenía que dedicar a otras asignaturas.

11.5 – Incidencias Principales

Cabe destacar que a la hora de realizar la introducción de la memoria el alumno no disponía de material suficiente para presentar el News Explorer. Como se propuso en su momento en el plan de contingencia, el director del proyecto facilitó nueva documentación donde consultar toda la información requerida.

Por otra parte, en varias ocasiones nos ha sido imposible llevar a cabo una reunión y en la mayoría de los casos la hemos aplazado o hemos solucionado la razón de la reunión por correo electrónico.

Como hemos dicho en este mismo apartado, el alumno se ha retrasado en varias ocasiones en la entrega de la memoria. Estos incidentes se han resuelto aplazando varias veces la finalización de la memoria y del proyecto.

En los casos en los que la memoria no se ajustaba a las características que se esperan de un documento formal como éste, el director del proyecto ha informado de ello al alumno y éste ha corregido las diferencias.

En cuanto a la pérdida de datos o la avería del ordenador, si bien es cierto que el alumno no ha tenido ningún problema de este tipo, hay que decir que durante el transcurso del proyecto se han realizado copias de seguridad a diario y se ha guardado una copia de

cada prototipo por separado. Lo que quiere decir que aún si se hubiese producido algún extravío de información, el alumno podría haber seguido trabajando sin contratiempos.

11.6 – Gantt Real

En la Figura 24 podemos ver que se han cumplido las fechas de finalización previstas para la mayoría de procesos exceptuando la memoria. El retraso en la memoria ha producido que se postergue la finalización del proyecto, lo que ha derivado en más reuniones y una última iteración.

Como hemos expuesto en este mismo apartado, la duración del proyecto no refleja las horas invertidas, dado que entre febrero y septiembre el alumno estuvo inmerso en otros proyectos.

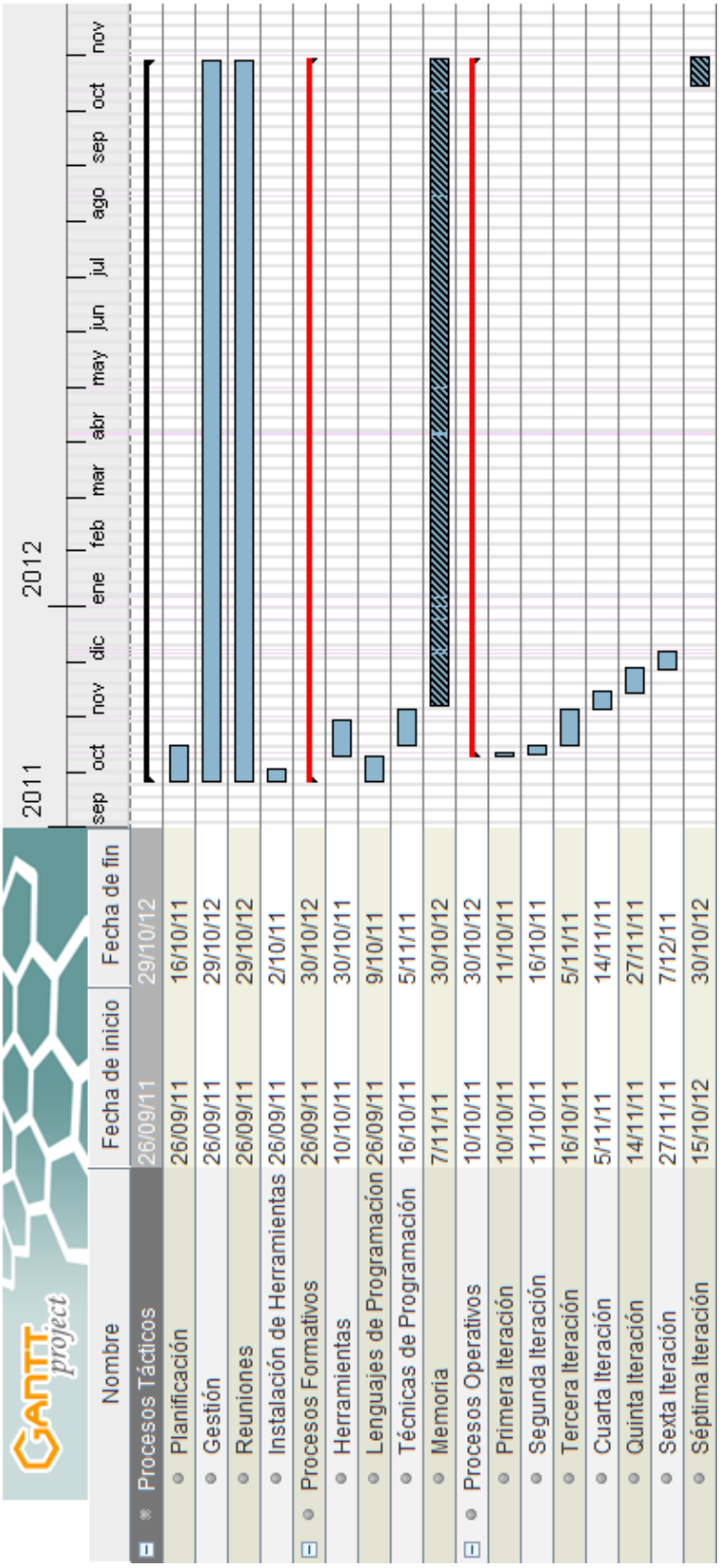


Figura 24: Diagrama de Gantt - Duración Real

12 – Conclusiones

A pesar de las innumerables re-planificaciones que ha sufrido el proyecto y de la larga duración del mismo, la impresión de conjunto que me queda del proyecto es muy satisfactoria.

Por una parte hemos alcanzado con creces los objetivos planteados al inicio del proyecto desarrollando un programa que se ejecuta a diario y que obtiene grandes cantidades de información sobre las noticias del día en múltiples idiomas. Como hemos comentado en el apartado 10.2, ya se han extraído los datos de 104.794 clústers y se ha procedido a analizarlos como proyecto de fin de máster de otro alumno.

Otro factor de gran importancia para calificar de éxito este proyecto es la cantidad de herramientas nuevas que el alumno ha aprendido a utilizar. Desde el lenguaje de programación Perl, pasando por las expresiones regulares y la extracción de datos de una página web, hasta la propia redacción de la memoria.

12.1 – Mejoras futuras

Hemos conseguido completar los objetivos principales pero a lo largo del proyecto han ido surgiendo posibles mejoras que sería interesante realizar en un futuro cercano. En este apartado recogemos las más significativas y describimos los objetivos a cumplir en cada una de ellas.

12.1.1 – Extraer más datos de interés

Hemos recogido los enlaces a las páginas de historias, lugares, países, personas y otros nombres. Sin embargo estaría bien que accediésemos a dichas páginas para obtener otros datos, como un enlace a la página de Wikipedia de la entidad en cuestión, los nombres que se le asocian a la misma persona en diferentes idiomas, o cómo está relacionada con otras personas u organizaciones.

12.1.2 – Obtener los clústers relacionados del mismo idioma

De momento sólo extraemos los clústers relacionados con cada uno de los clústers del RSS en otros idiomas, pero existe una sección de la página de clúster en la que se recogen los enlaces a todos los clústers pertenecientes a la misma historia que el actual.

Sería interesante recopilar también esos clústers y guardar la relación entre ellos.

12.1.3 – Guardar los datos en otro formato

Uno de los mayores problemas que deriva del uso de bases de datos relacionales es que si en las tablas se utilizan campos de mayor tamaño que el de los elementos que se guardan en ellos, el tamaño que la base de datos ocupa en disco crece muy rápido a medida que se añaden datos.

Hemos utilizado bases de datos relacionales por ser una herramienta conocida por el alumno pero es probable que en relativamente poco tiempo la base de datos ocupe más de lo que debería.

Existen otras tecnologías para el almacenamiento organizado de los datos. Entre ellas que se encuentran las dos siguientes alternativas:

- Bases de datos XML
- Una base de datos relacional con las claves de clústers más una colección de documentos XML organizados en directorios.

Ambas soluciones tienen sus pros y sus contras pero en las dos sus campos ocupan sólo lo que ocupa su contenido.

12.1.4 – Recuperar todo el cuerpo de las noticias

Hemos comentado en repetidas ocasiones que la finalidad de este proyecto no termina con la extracción de datos de EMM, sino que luego esos datos se van a utilizar para realizar análisis relacionados con el procesamiento del lenguaje natural.

Una de las áreas en la que se quería trabajar con los datos es la de comparar los clústers de una misma historia en diferentes idiomas para sintetizar estructuras del lenguaje natural que luego se pueden utilizar en la traducción de textos.

Por el momento sólo recogemos el título y la descripción de cada noticia pero lo realmente interesante sería poder comparar los textos completos de las noticias. Del mismo modo que hemos extraído datos de la página del News Explorer también podríamos acceder a la página de cada noticia y extraer su contenido, teniendo cuidado de seleccionar sólo el contenido que nos interesa.

Existen varias herramientas que nos facilitarían este trabajo, como por ejemplo la librería boilerpipe de Java[31].

[31] http://boilerpipe-web.appspot.com/

12.1.5 – Normalizar las fechas

En la actualidad las fechas se extraen tal cual en el idioma en el que esté el clúster. Esto produce que sea algo complicado consultar a la base de datos por los clústers de un día determinado. El objetivo de esta mejora sería el de normalizar las fechas de diferentes idiomas a un único formato para poder realizar búsquedas por fecha más fácilmente.

13 – Bibliografía

[1]. Sean M. Bourke (2004). *LWP &Perl*. O'Reilly
- ISBN-10: 0596001789
- ISBN-13: 978-0596001780
- http://lwp.interglacial.com/

[2]. Tara Calishain, Kevin Hemenway (2003). *Spidering Hacks*. O'Reilly
- ISBN-10: 0596005776
- ISBN-13: 978-0596005771

[3]. Tom Christiansen, Nathan Torkington (2003). *Perl Cookbook 2nd Edition*. O'Reilly
- ISBN-10: 0596005776
- ISBN-13: 978-0596005771

[4]. Ben Hammersley (2005). Developing Feeds with RSS and Atom. O'Reilly
- ISBN-10: 0596005776
- ISBN-13: 978-0596005771

[5]. Jeffrey Friedl (2002). Mastering Regular Expressions 2nd Edition. O'Reilly
- ISBN-10: 0596005776
- ISBN-13: 978-0596005771