# Arquitectura de un Crawler para Extraer las Estructuras y Contenidos de Recursos Electrónicos

F. de la Rosa T., R M. Gasca, C. Del Valle y R. Ceballos
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
{ffrosat, gasca, carmelo, ceballos}@lsi.us.es

**Abstract.** Hoy en día Internet se nos presenta como el gran escaparate de la información, la cual podemos observar y consultar. Sin embargo, esta información es difícil de manipular o procesar para asimilarla adecuadamente o adquirir nuevos conocimientos. En este trabajo presentamos una arquitectura que permite la extracción de las estructuras y los contenidos de los recursos electrónicos, con el objeto de facilitar el procesamiento y/o la representación de dicha información mediante otros paradigmas para mejorar su comprensión.

**Palabras Claves.** Arquitectura Crawler, Extracción de Información, Navegación, Recursos Electrónicos, Web Semántica, Wrapper Inductivo, Base de Datos Semiestructurada, Digestión de Información, WebL.

## 1. Introducción

La enorme cantidad de información disponible en Internet es fácil de observar y consultar, pero difícilmente podemos convertirla en conocimiento. Se está realizando un gran esfuerzo en algunas disciplinas para desarrollar el concepto de *Web Semántica* [1][2], cuyo objeto es proporcionar un orden en el caos que gobierna la información almacenada en Internet. Se han conseguido varios logros con el desarrollo del metalenguaje de marcas XML [3]. El objetivo de este metalenguaje es que la información que hoy se encuentra en los recursos electrónicos en formato HTML sea accesible en el futuro a través de una sintaxis común para su automatización. Para conseguir esta automatización el lenguaje XML asocia a los datos una información semántica, lo que permite el posterior cruce de distintas fuentes electrónicas. El metalenguaje XML no impide el acceso a la información como se ha venido realizando hasta la fecha, pudiéndose utilizar una plantilla XLST, donde se describe el proceso de transformación de los repositorios de información XML en algún formato interpretable por los navegadores, como puede ser el HTML.

Como hemos visto, el metalenguaje XML permite asociar información semántica a los datos. Sin embargo, esto no es suficiente para que varios actores independientes

puedan colaborar para conseguir sus objetivos. Para ello es necesario definir una ontología, es decir, debemos describir los conceptos y las relaciones de un dominio común para que pueda existir una comunicación entre los distintos actores. Para realizar las definiciones de las ontologías existen varios lenguajes, como pueden ser XOL, Topic Maps, SHOE y DAM+OIL [2]. Pero entre todos destaca RDF [3][4], un estándar de propósito general para describir ontologías, auspiciado por el consorcio W3C [3] y que está definido sobre XML. El concepto de ontología se ha fraguado en diferentes disciplinas, como en la Ingeniería del Conocimiento con el objeto de desarrollar sistemas expertos, o en el área de Documentación para el desarrollo de estándares en la catalogación de recursos, no sólo electrónicos o bibliográficos, sino también otros tipos de recursos. Destacamos como estándar de catalogación *DublinCore* [5].

Una de las pruebas de que la Web Semántica no es aún una realidad es que uno de los campos más activos de investigación es el de la extracción de información Web mediante el uso de *wrappers inductivos* [6]. En los trabajos actuales se están desarrollando sistemas capaces de aprender a extraer de páginas HTML la información que nos interesa. Para ello utilizan una pequeña porción de los datos que quieren extraer para conseguir que el sistema aprenda a recuperarlos. Estos esfuerzos nos hacen pensar que en el futuro una gran parte de la información a la que podremos acceder seguirá siendo accesible solamente mediante páginas HTML.

En este trabajo presentamos la arquitectura de una herramienta que permite la extracción de información de los recursos electrónicos, tanto de su contenido como de su estructura y además con un bajo coste tanto en recursos humanos como en medios. A diferencia de los sistemas inductivos, donde se busca extraer la información de forma automática, nuestra herramienta es programada para navegar por los recursos electrónicos y extraer la información que nos interesa como lo harían los programas rastreadores de la red, también conocidos como *crawlers* [7][8]. Nuestra propuesta no debe ser entendida como contrapuesta a éstas propuestas, sino en simbiosis con ellas, ya que una de las aplicaciones más interesantes, como veremos posteriormente, es la generación de corpus para entrenar sistemas inductivos de forma general (como pueden ser los sistemas para extraer información de páginas Web o algunos modelos de procesamiento de lenguaje natural).

El resto del trabajo queda estructurado de la siguiente forma: en la sección 2 presentamos la arquitectura primitiva de nuestro sistema y los nuevos requisitos que surgieron tras la etapa de pruebas. En la sección 3 presentamos la arquitectura actual del sistema surgida a partir del sistema primitivo y el modelo conceptual subyacente para la descripción de la navegación y extracción de la información. Por último en la sección 4 presentamos las conclusiones, centrándonos en dos aspectos fundamentales: (i) las características particulares de nuestra arquitectura y (ii) la aplicabilidad de ésta.

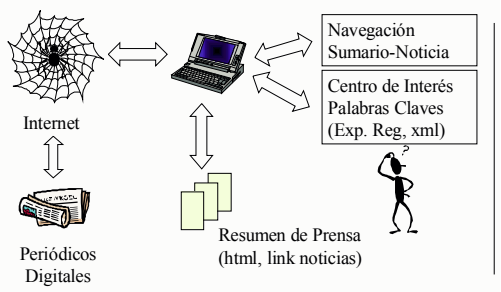
## 2. Arquitectura Primitiva: Generación de Resúmenes de Prensa a partir de Periódicos Electrónicos

La motivación de la herramienta que presentamos en este trabajo fue la creación de un periódico electrónico a medida, que hiciese llegar diariamente a cada uno de los

usuarios del servicio un resumen de prensa personalizado, con el objeto de *facilitar al usuario la asimilación* de información. Los resúmenes estaban personalizados con la relación de las noticias publicadas en los medios de comunicación que eran de interés para el usuario y por un estudio estadístico del impacto que ellas tenían en los medios.

Para la generación de los resúmenes la herramienta debía conocer cómo navegar por los periódicos electrónicos y cuáles eran los centros de interés de los usuarios. El paradigma utilizado para la descripción de la navegación se basaba en la especificación de dos elementos: *secciones* y *noticias*. Las *secciones o semillas* son las páginas Web donde se inicia el proceso de navegación y la forma de especificar las semillas se hacía indicando su dirección URL. A diferencia de las secciones, las *noticias* eran páginas desconocidas para el sistema, pero podían describirse utilizando un rango de URLs. Con estas descripciones el núcleo de la herramienta extraía las referencias almacenadas en las páginas semillas y en caso de encontrar una referencia que perteneciese a algún rango de noticias, realizaba una descarga de la noticia.

Para poder generar los resúmenes de prensa era necesario que el usuario definiera sus centros de interés. Como lenguaje para definir estos centros de interés se utilizaban las expresiones regulares que proporciona WebL [9] y que se utilizaban para la búsqueda de patrones en ficheros planos. De esta forma la herramienta asociaba una noticia al centro de interés de un usuario si encontraba en ella alguno de los patrones definidos por dicho usuario. En la *Ilustración 1* se muestra el esquema de la arquitectura y la implementación del núcleo del sistema en pseudocódigo.




```
Semillas <- {Referencias Iniciales}
MIENTRAS Semillas.NoVacia() HACER
    refS <- Semillas.PrimerElemento()
    refS.Descarga()
    refNs <- refS.Referencias()
    MIENTRAS refNs.NoVacia() HACER
        refN<-refNs.PrimerElemento()
        SI EsNoticia(refN) ENTONCES
           refN.Descarga()
           refN.Clasifica()
        FIN SI
        refNs = refNs - {refN}
    FIN MIENTRAS
    Semillas = Semillas - {refS}
FIN MIENTRAS
```

***Ilustración 1:*** *Arquitectura del sistema y pseudocódigo del núcleo del sistema*

La herramienta que hemos descrito arriba fue la primera versión del sistema y tras un periodo de pruebas se detectaron dos inconvenientes importantes en su arquitectura:

- El sistema de generación de informes era poco modular y resultaba muy costoso cambiar el contenido de los resúmenes de prensa.
- Un porcentaje alto de noticias eran clasificadas incorrectamente, ya que durante el proceso de clasificación, no se separaba el contenido de la noticia del resto de la página Web. Esto generaba ruido en el proceso de clasificación pues por norma general las páginas que albergaban las noticias hacen referencia a otras noticias que no tienen relación alguna con el contenido de la misma.

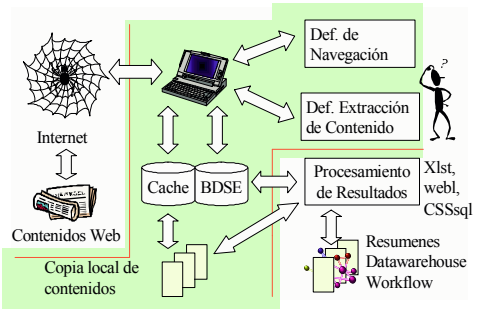
Como resultado de la fase de prueba obtuvimos una lista de nuevos requisitos que debía contemplar la nueva arquitectura:

- Era necesario mantener una cache de URL descargadas, para *evitar descargas duplicadas y bucles infinitos* en el procesamiento de las noticias y secciones.
- Resultaba imprescindible cambiar el paradigma de navegación para a*mpliar el dominio de procesamiento de los recursos electrónicos,* de forma que pudiésemos procesar cualquier estructura encontrada como recurso electrónico y no solamente periódicos digitales.
- Era conveniente reducir el tiempo de descarga, aprovechando al máximo el ancho de banda y evitar ataques DoS (Denegación de Servicios) sobre los servidores. Por ello surge como nuevo requisito el realizar *descargas de forma concurrente*.
- Finalmente, para resolver los dos inconvenientes antes mencionados se creyó justificada la necesidad de *extraer información tanto de las estructuras como de los contenidos* de los recursos electrónicos y almacenar esta información en una *base de datos*. No bastaba con descargar las páginas Web, era necesario extraer parte de su contenido. En el caso de las noticias, es posible la extracción de items tales como: los autores, los resúmenes, los títulos, la localización de los sucesos, etc. Esta solución permite: 1) eliminar los problemas de ruidos detectados, al obtener la noticia sin referencias falsas a otras noticias; 2) alimentar una base de datos para mantener un sistema modular de generación de informes utilizando un lenguaje de consultas.

Para satisfacer todos estos requisitos no bastaba con disponer de funciones para la búsqueda de expresiones regulares en los textos, necesitábamos además otros tipos de servicios, que nos permitiesen implementar procesos concurrentes y manipular páginas HTML o XML. Hemos encontrado en el lenguaje WebL todos los servicios necesarios para implementar la nueva arquitectura, consiguiendo además una plataforma de programación muy homogénea. Esto diferencia la arquitectura que proponemos en este trabajo de otros sistemas como LORE [10], ARANEUS [11] o AKIRA [12], que necesitan utilizar varios lenguajes para describir su funcionamiento.

## 3. Arquitectura Actual: Extracción de Estructuras y Contenidos a partir de Recursos Electrónicos en General

En esta sección describimos la arquitectura actual del sistema y por su interés dedicaremos dos subapartados específicos donde serán desarrollados: primero el modelo utilizado para navegar y extraer información, y finalmente la interfaz utilizada para desacoplar la herramienta de otros sistemas de información. En el esquema que aparece en la *Ilustración 2* podemos observar una visión global de la arquitectura que iremos exponiendo a lo largo de la sección.




```
Semillas <- {Referencias Iniciales}
MIENTRAS Semillas.NoVacia() HACER
    ref <- Semillas.PrimerElemento()
    ref.Descarga()
    refNs <- ref.Referencias()
    MIENTRAS refNs.NoVacia() HACER
        refN<-refNs.PrimerElemento()
        SI EsInterior(refN) ENTONCES
            Semillas = Semillas + {refN}
        FIN SI
        SI EsHoja(refN) ENTONCES
            refN.Descarga()
            refN.Procesa()
        FIN SI
        refNs = refNs - {refN}
    FIN MIENTRAS
    Semillas = Semillas - {ref}
FIN MIENTRAS
```

***Ilustración 2:*** *Arquitectura del sistema y pseudocódigo del núcleo del sistema*

Uno de los subsistemas más importantes de la herramienta es la *cache de descargas.* La función de este subsistema es la de gestionar las descargas de ficheros y páginas Web, evitando las descargas y los tratamientos duplicados. Para realizar esta gestión, el subsistema mantiene en un fichero XML la información de las descargas realizadas, almacenando la dirección URL, la localización de las descargas y la información que los servidores proporcionan sobre el estado de los ficheros descargados. Entre la información proporcionada por los servidores nos encontramos con la fecha de actualización del fichero. Esta información es fundamental y permite al subsistema de cache decidir en el futuro si es necesario o no realizar de nuevo la descarga. En la *Ilustración 3* podemos ver un ejemplo del fichero XML que mantiene la cache de descargas.

```
<Cache objeto="CacheAlbumes.xml">
  <EntradaBck url="http://www.efrance.fr/articulos/basedonnee/albumescompletos/estadisticas.asp">
    <URL> http://www.efrance.fr/articulos/basedonnee/albumescompletos/estadisticas.asp</URL>
    <Fichero>d:\tmp\Albumes\trash\ind16112.html</Fichero>
    <Dia> 4</Dia><Mes>3</Mes><Anyo>2002</Anyo>
  </EntradaBck>
  <EntradaBck url="http://www.efrance.fr/articulos/basedonnee/albumescompletos/listaasp">
    <URL>http://www.efrance.fr/articulos/basedonnee/albumescompletos/listar.asp</URL>
    <Fichero>d:\tmp\Albumes\trash\ind16113.html</Fichero>
    <Dia>4</Dia><Mes>3</Mes><Anyo>2002</Anyo>
  </EntradaBck>
  ...
</Cache>
```

***Ilustración 3:*** *Ejemplo de fichero XML mantenido por el cache de descarga*

El sistema realiza la descarga y el tratamiento de las páginas de forma concurrente. Tras la descarga de la página, ésta pasa al *subsistema de procesamiento.* Como veremos en próximos subapartados, el procesamiento de las páginas se particulariza para cada recurso electrónico, mediante la implementación de dos tareas*: la navegación y la extracción de información,* siendo ésta última opcional.

Si existe la tarea de extracción para una página, el sistema la utiliza para obtener un objeto que mantiene la información relevante. Y mediante la introspección del objeto el proceso de almacenamiento consigue guardar la información que contiene en una *base de datos semiestructurada (BDSE)*. Podemos apreciar esta dinámica en la *Ilustración 4.*

Aunque el formato utilizado para implementar la base datos es XML, no se define una estructura semántica fija, ya que ésta se adapta al contenido de cada objeto almacenado, obteniendo por tanto una base de datos de objetos heterogéneos. Debido a la importancia de la BDSE dentro de la arquitectura, como interfaz para alimentar otros sistemas de información, le dedicamos un subapartado dentro de esta sección.

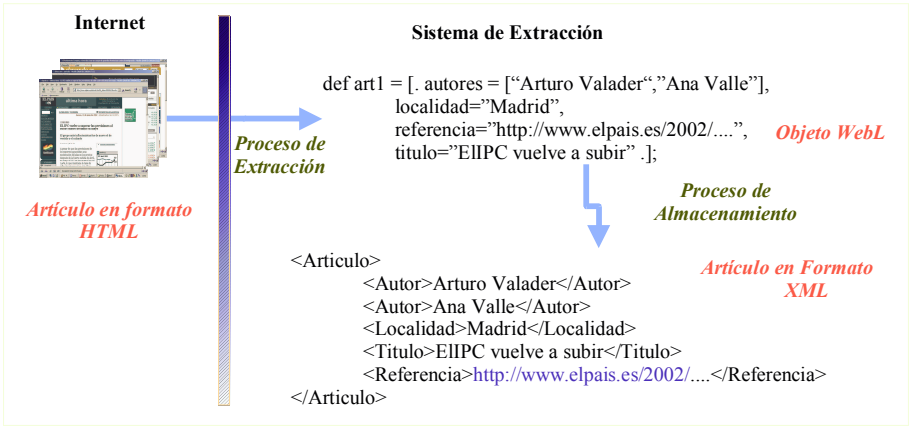



***Ilustración 4:*** *Etapas en el procesamiento de un artículo*

Además, podemos recalcar que la arquitectura propuesta genera dos tipos de productos: por un lado mantiene una *copia local de las páginas* visitadas, operación que se realiza automáticamente, y por otro lado se puede configurar para generar una *BDSE* con información extraída de los recursos visitados.

### 3.1. Paradigma de Navegación y Extracción de Contenidos

En el proceso de navegación el usuario define cómo la herramienta debe navegar a través del recurso electrónico a visitar. El paradigma utilizado para navegar tiene su base en la catalogación de las referencias según dos características: su comportamiento o modo de navegación y su contenido. Debido a que la catalogación de las referencias de forma individual consumiría muchos recursos, la catalogación se realiza por *rangos de referencias*, describiendo éstos mediante expresiones regulares. Por tanto, una referencia pertenece a un rango si la referencia es reconocida por la expresión regular que representa al rango. Dentro de los rangos podemos definir *islas de rangos de referencias*, que determinan las referencias que son excluidas del rango principal.

En el modelo de navegación se han detectado tres posibles modos de navegación según el comportamiento de las referencias:

**Referencias Hojas:** Son referencias que tienen asociado un proceso de extracción de información para los contenidos a los que hacen referencia. La información extraída será posteriormente almacenada en la base de datos semiestructurada.

**Referencias Intermedias:** Son referencias usadas para extraer otras referencias que realimenten el proceso de navegación. En algunos casos, el conjunto de referencias

extraídas se verá influido por su contenido. Nada impide que una referencia intermedia sea definida también como referencia hoja .

**Referencias Semillas:** Son las referencias iniciales, desde las cuales comienza el proceso de navegación. En el modelo de navegación de la arquitectura primitiva de la herramienta, éstas referencias eran conocidas como referencias a sumarios. La forma de procesar estas referencias es similar a las referencias intermedias.

Como ya hemos comentado al principio de la sección, además de realizar una catalogación de los rangos según su comportamiento, existe también una catalogación subjetiva dependiente de los contenidos de los recursos electrónicos. Por ejemplo, si estamos navegando en un servidor de ficheros de música, se podría definir una categoría de referencias que distinguiesen los siguientes contenidos: tendencias musicales, grupos musicales, miembros de grupos, álbumes y canciones de los grupos. Una vez catalogado los rangos el usuario deberá implementar los procesos de navegación y de extracción para cada uno de ellos. Muchos procesos de navegación y extracción serán reutilizables gracias a su catalogación. En la *Ilustración 5* podemos ver las relaciones entre las diferentes categorías y de procesos.

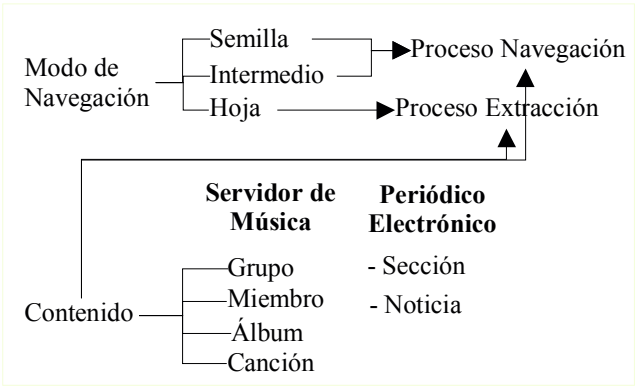



***Ilustración 5:*** *Categorización de los procesos según modo de navegación y contenidos de los rangos*

### 3.2 Extracción de Información a partir de Recursos Electrónicos

En la primera arquitectura que hemos presentado hacíamos uso de las expresiones regulares como lenguaje para definir los centros de interés y generar resúmenes de prensa personalizados. Aunque los lenguajes regulares eran mecanismos adecuados para nuestros propósitos, en esta segunda arquitectura que presentamos resultan insuficientes para extraer información manipulando los contenidos de las páginas Web. Los objetivos han cambiado y nuestra intención es la extracción de los campos específicos de esos contenidos. Por ejemplo, para el caso de las noticias podemos extraer: el autor, la localidad donde se produjo el suceso, el texto de la noticia, etc. A este conjunto compacto de campos extraídos de una página Web y que hacen referencia a una misma entidad se le denomina *esquema.* En el ejemplo anterior la entidad sobre la cual se construye el esquema sería la noticia.

A diferencia de otras arquitecturas *wrappers* como LORE[10], ARANEUS[11] y AKIRA[12], que utilizan analizadores léxicos y sintácticos para construir *parsers* que extraen esquemas predefinidos, nuestra herramienta utiliza un *álgebra de etiquetas*. Los lenguajes que implementan álgebras de etiquetas tienen un gran potencial para extraer esquemas de los lenguajes de marcas como HTML y XML. Este es el caso del lenguaje WebL.

Las álgebras de etiquetas permiten recuperar de las páginas Web conjuntos de etiquetas y conseguir la localización de la información que queremos extraer mediante la aplicación sucesiva de los operadores básicos de los conjuntos: unión, intersección, selección y diferencia. Como se puede observar en el ejemplo de la *Ilustración 6*, en el que se imprimen las referencias contenidas en una página Web, las álgebras de etiquetas proporcionan un modelo menos genérico pero más potente que el modelo utilizado por los analizadores léxicos y sintácticos para manipular textos etiquetados.

```
var P = GetURL("http://www.elmundo.es/nacional.html");
every e in Elem(P,"a") do
   PrintLn(e.href);
end;
```

***Ilustración 6:*** *Ejemplo de extracción de referencias mediante un álgebra de etiquetas*

Nuestra arquitectura utiliza el álgebra de etiquetas que implementa WebL para construir los *procesos de extracción* utilizados para alimentar la BDSE de contenidos. Cada objeto extraído pertenecerá implícitamente a un esquema y este esquema se almacenará total o parcialmente en la BDSE.

```
// Funciones recolectoras de información.
var Autores = defun(obj,HTML)
   .... // Código extracción.
end;
var Titulo = defun(obj,HTML)
   .... // Código extracción.
end;
var Resumen = defun(obj,HTML)
   .... // Código extracción.
end;
   ... ...
export var Constructor = fun(HTML)
   // Recopilación de los datos.
   var datos = Autores([..],HTML);
   datos = Titulo(datos,HTML);
   datos = Resumen(datos,HTML);
   datos = Contenido(datos,HTML);
   datos = FechaSuceso(datos,HTML);
   datos = Localizacion(datos,HTML);
   return datos;
end;
```

***Ilustración 7:*** *Ejemplo de biblioteca WebL para extraer un esquema de noticias*

### 3.3 Integración del Sistema en otros Procesos

Una de las características más interesantes de la arquitectura es el *desacoplamiento* del proceso de extracción de información del proceso de explotación de los datos extraídos. Esto se ha conseguido a través de la BDSE. Esta característica permite integrar nuestro sistema en otros sistemas más complejos, como pueden ser un *Datawarehouse*, un *Workflow*, un generador de resúmenes de prensa o algún paradigma de representación visual. De esta forma se permite la construcción *escalar* de un sistema de información. Por ejemplo, se pueden ir actualizando los datos de un *Datawarehouse*, a medida que se vayan obteniendo resultados de los recursos electrónicos o ir generando nuevos contenidos como los resúmenes de prensa.

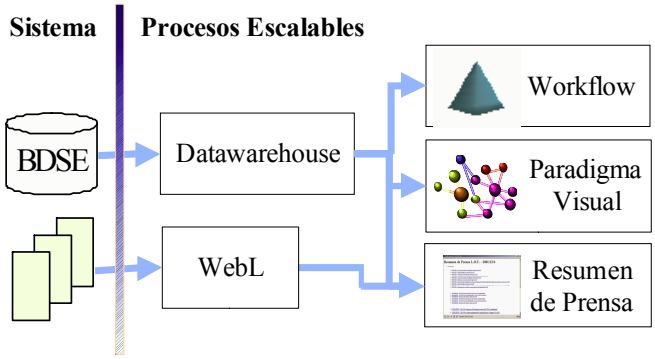



***Ilustración 8:*** *Posibilidades de los procesos escalables*

No hay que olvidar que el sistema siempre realiza una descarga local de los contenidos visitados, aunque llegado el caso estas descargas puede ser eliminadas, por problemas de falta de espacio. La información descargada localmente puede resultar útil para extraer nuevos esquemas que anteriormente no han sido tenidos en cuenta o disminuir el tiempo de acceso de los usuarios que visiten nuestros contenidos. A diferencia de la descarga local, la extracción del contenido a una BDSE es opcional.

## 4. Conclusiones

De la arquitectura propuesta en este trabajo podemos destacar las siguientes características:

- La generación automática de una base de *datos semiestructurada* en XML de las estructuras y los *contenidos* de los recursos electrónicos distribuidos por Internet.
- El *desacoplamiento de la estructura y el contenido* de los recursos electrónicos, del paradigma o de los procesos utilizados para digerir adecuadamente la información.
- *La robustez* alcanzada al permitir el cruce de la información de varios recursos electrónicos mediante el mantenimiento de un *Datawarehouse*. Esto permite completar las descargas parciales o fallidas de información y evitar que el usuario final se quede sin servicio.

- Es un sistema que necesita *poco esfuerzo de mantenimiento,* ya que es fácil detectar cuándo se extrae de forma incorrecta la información y por tanto como corregir estas situaciones anómalas.

Todo ello conduce a una alta aplicabilidad del sistema. Podemos poner como ejemplos de aplicabilidad:

- La generación de *corpus de noticias* de prensa para entrenar modelos de procesamiento del lenguaje natural, o para entrenar sistemas inductivos de extracción de información Web.
- La generación de *periódicos a la carta,* donde podemos generar tanto resúmenes de noticias catalogadas, como realizar estudios estadísticos de los impactos de las noticias.
- La realización de *copias locales* de recursos electrónicos o la *monitorización de recursos* mediante el envío de avisos en caso de producirse cambios en los contenidos.

Como conclusión final podemos decir que la construcción de este sistema consigue mostrar la viabilidad de la extracción de información de Internet de forma semiautomática, siempre y cuando estos sistemas mantengan una cierta redundancia sobre los recursos, que evite la siempre indeseable caída de los servicios. Por tanto para asegurar el éxito de estos sistemas, creemos que es necesario realizar una planificación detallada de los recursos a explotar, que permita alcanzar las expectativas de los servicios ofertados.

## Referencias


1. Tim Berners-Lee, James Hendler and Ora Lassila. "The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.". May 2001. The Semantic Web, Scientific American.
2. Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W.Fergerson,a Mark A. Musen. "Creating Semantic Web Contents with Protégé-2000". Sandford University. 2001 IEEE Intelligent Systems.
3. W3C Word Wide Web Consortium: *http://www.w3.org/*
4. Pierre-Antonie Chaping "RDF Tutorial": *http://www710.univ-lyon1.fr/~champin/rdf-tutorial/*
5. Dublin Core Metadata Iniciative: http://dublincore.org/
6 Ariadne: *http://www.isi.edu/ariadne/demo/index.html*
7. Prasanaa Thati, Po-Hao Chang, and Gyl Agha. "Crawlets: Agents for High Performace Web Search Engines" University of Illinois at Urbana-Champaign USA. 5th International Conference, Mobile Agents 2001. Springer.
8. Goerge Chang, Marcus J. Healey, James A. M. Mc Hugh and Jason T. L. Wang. "Mining the World Wide Web. An Information Search Approach." Kluwer Academic Publishers.
9. Hannes Marais. "Compaq's Web Language. A Programming Language for the Web". Compaq Systems Resarch Center (SRC).
10. Lore: *http://www-db.stanford.edu/lore/*
11. Araneus*: http://www.difa.unibas.it/araneus/index.html*
12. Akira: http://www.cis.upenn.edu/~lacroix/AKIRA/SIGMOD/akira_home.html