Universidad Zaragoza
1542

Trabajo Fin de Grado

GeoCrawler: sistema de crawler web enfocado al descubrimiento de información geográfica

Autor

Jorge Cáncer Gil

Director

Francisco Javier López Pellicer

Escuela de Ingeniería y Arquitectura

2016

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

GeoCrawler: sistema de crawler web enfocado al descubrimiento de información geográfica

RESUMEN

Actualmente, los videojuegos están experimentando un gran auge tanto a nivel industrial como de volumen de jugadores. Uno de los formatos más populares es, sin duda, el de los videojuegos multijugador online.

En este tipo de juegos, el modelo de interacción entre jugadores consiste en que cada jugador controla una entidad dentro del juego, normalmente independiente del resto de jugadores, por ejemplo, un personaje, un vehículo o un ejército.

Este trabajo explora la idea de que varios jugadores controlen de forma simultánea a la misma entidad mediante un sistema que ejecute en tiempo real la voluntad de la mayoría.

Con este sistema de "control compartido", se persigue que los jugadores tengan que adaptarse al curso de acción decidido por la mayoría en cada momento y tengan que reevaluar constantemente su estrategia en función de lo que está sucediendo y no de los planes que tuvieran previamente.

Para lograr esto, se ha desarrollado un sencillo videojuego de plataformas en dos dimensiones, en el cual los jugadores controlan a un robot que recolecta mineral en un planeta desconocido.

El mundo por el que se mueve el robot se genera de forma distinta en cada partida, para evitar que los jugadores puedan memorizar los caminos, de forma que no se pierda el componente de incertidumbre, que es esencial en este modelo de interacción.

Una vez se terminaron el videojuego y el sistema de control compartido, se realizaron pruebas con usuarios para comprobar si el sistema funciona de forma adecuada y si este tipo de interacción es atractiva para los jugadores.

Contenido

1. Introducción

Descubrir y recolectar servicios de información geográfica, colecciones de datos y recursos geográficos en la web mediante los buscadores comerciales tradicionales es muy complicado.

La mejor estrategia es desarrollar una *crawler* web enfocado al descubrimiento y recuperación de información geográfica.

1.1. Objetivo

El principal objetivo de este trabajo de fin de grado o TFG es la realización de un *crawler* enfocado al descubrimiento de información geográfica en la web.

En este trabajo se ha profundizado en la recuperación de documentos correspondientes a las especificaciones de OGC (Open Geospatial Consortium).

Estos documentos son documentos de descripción de diferentes tipos de servicios de red definidos por OGC (WPS, WCTS, WMS, WMTS, WCS y CSW). Todas estas variantes serán explicadas posteriormente en esta memoria. También se realiza la recuperación de servicios de descarga Atom.

Gracias al *crawler* se deben recuperar gran cantidad de documentos que describen servicios geográficos. Estos documentos tecogidos pueden ser útiles para tareas de *big data* y análisis de datos.

1.2. Contexto

Este trabajo ha sido completado como parte del grupo de Sistemas de Información Avanzados (IAAA).

1.3. Resolución del problema

Para conseguir llegar al objetivo expuesto anteriormente se va a partir de Apache Nutch. Apache Nutch es un *crawler* genérico desarrollado por Apache.

Se ha desarrollado una extensión de Nutch, el *crawler* genérico de *Apache Software Foundation*, para convertirlo en un *crawler* enfocado con el propósito antes indicado.

Se han utilizado técnicas algorítmicas para extraer los términos relevantes de una página web para detectar si puede o no ser relevante para el descubrimiento de información geográfica, además de diferentes heurísticas de

búsqueda basadas en los usos y costumbres de publicación de información geográfica en la web.

Nutch basa su arquitectura en Hadoop, esto le permite trabajar con un gran volumen de datos, como el que se genera en el proceso de *crawling*, utilizando operaciones de MapReduce.

Nutch por defecto está preparado para indexar la información recogida en el motor de búsqueda Apache Solr, el cual está basado en Lucene.

No obstante, Nutch permite la adición de *plugins* que modifican su comportamiento. Estos *plugins* actúan sobre los llamados **puntos de extensión**. Los puntos de extensión corresponden con partes del proceso de *crawling* como pueden ser las fases de *fecth, parse, scoring* y *indexing*.

Es en estos puntos de extensión donde se ha desarrollado el trabajo necesario para convertir Nutch en un crawler que se dedique a buscar y guardar ficheros de descripción de servicios de información geoespacial.

1.4. Estructura de la memoria

En esta memoria se explica qué son los *crawlers*, para qué se utilizan y por quién. También se mencionan sus componentes y los elementos que rodean a los *crawlers*.

Por otra parte, se ahonda en qué es Nutch, cómo funciona, las herramientas que utiliza y su arquitectura.

Posteriormente se incluye el trabajo realizado para preparar Nutch para la tarea que se ha explicado anteriormente, así como las herramientas y tecnologías utilizadas para la realización de este proyecto.

Se presentan los resultados a los que se ha llegado con el trabajo, la calidad de estos y la rapidez con la que son obtenidos.

También se habla de posibles mejoras y ampliaciones que pueden darse en este trabajo con el objetivo de aumentar la capacidad y el alcance del *crawler.*
Se pueden consultar anexos con información sobre el tipo de ficheros que se pretenden recuperar y para qué se utilizan.

También existe un manual de usuario en el cual se explica paso por paso cómo utilizar GeoCrawler.

2. Crawlers

Un crawler es una herramienta que recorre e inspecciona páginas web. Lo hace, generalmente, de manera automática y de acuerdo a un método que fija su comportamiento.

También son conocidas como arañas web o simplemente arañas en las zonas hispanoparlantes.

Tienen múltiples usos, pero el más usado es el de recorrer todas las páginas web que pueda recogiendo información y datos del contenido de cada una de ellas para posteriormente indexar toda esa información recogida en un motor de búsqueda.

Esta sería la definición de un crawler genérico dedicado a recoger la mayor cantidad de información posible. Sin ir más lejos, este sería el método que usa Alphabet para dar vida a su famoso buscador Google.

2.1. Crawlers genéricos

Estos crawlers se utilizan principalmente para darle vida a un buscador web.

Su funcionamiento básico se basa en extraer y seguir la mayor cantidad de URLs posibles extrayendo información de cada una de las páginas visitadas para posteriormente indexarla. Esto permite realizar búsquedas por términos. Recogen y siguen todos los links de salida de una web para llegar al mayor número posible de ellas.

Por lo tanto, estos *crawlers* se extiende de manera circular o de mancha de aceite ya que no siguen ningún criterio especifico que modifique el comportamiento de este.

En la **Ilustración 1** se puede observar una representación esquemática de cómo sería una expansión de un *crawler* genérico desde una semilla.

Como se puede observar la expansión se produce de manera circular al recoger y procesar todos los enlaces de salida que tiene una página web.

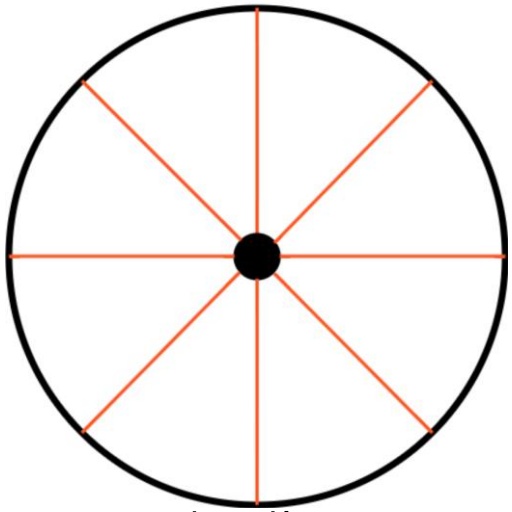

Ilustración 1

2.2. Crawlers enfocados

El término *crawler* enfocado es muy amplio y puede abarcar a un elevado número de tipos de *crawlers*. Existen muchas maneras para convertir un *crawler* en enfocado, esto depende en la medida en la que se quiera alterar el comportamiento de un *crawler* genérico.

Un *crawler* enfocado está preparado para no seguir el esquema de uno genérico. Es decir, no va a procesar todos los enlaces de salida de una web, sino que va a asignarles un *score* para priorizar qué enlaces va a expandir en la siguiente iteración.

Con esto, el flujo de trabajo del *crawler* prioriza unos enlaces antes que otros. De esta manera el *crawler* es más eficiente en la búsqueda de elementos concretos que uno genérico.

Los crawlers enfocados tienen muchos usos. Se pueden utilizar para indexar información de un tema específico buscando y priorizando enlaces que estén relacionados con este tema.

También se pueden utilizar para extraer un tipo de información o documentos de la web. Este es el caso que se ha utilizado en este trabajo. Nos vamos a centrar en la búsqueda y extracción de ficheros que definan información geográfica.

En la Ilustración 2 se puede observar una representación esquemática de un *crawler* enfocado. Estos se extienden de manera irregular expandiendo y

procesando enlaces que son más relevantes según se haya preparado al *crawler*.

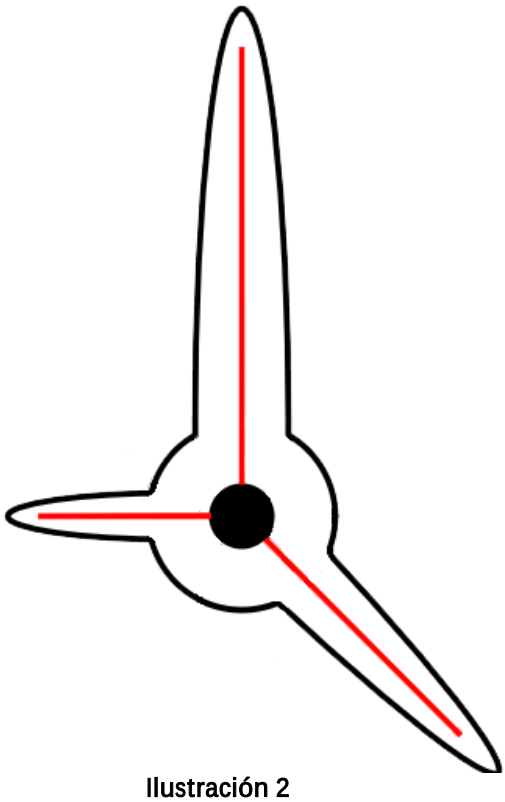

Ilustración 2

2.3. Funcionamiento de un crawler

El funcionamiento de los *crawlers* tiene, como norma general, un comportamiento iterativo. Es decir, que un crawler repite un conjunto de operaciones el número deseado de veces para desarrollar su trabajo.

Cada *crawler* puede tener sus peculiaridades en su funcionamiento. No obstante, en este apartado, se va a explicar la situación más típica con las operaciones más comunes.

En la primera ejecución de un *crawler* se realiza una operación denominada **inyección**. Esta operación inserta las URLs semilla. Es a partir de estas páginas web desde donde el *crawler* va a partir. La calidad de estas semillas influye mucho en la rapidez con la que se van a empezar a encontrar resultados significativos.

Esta operación, mencionada anteriormente, no está dentro del bucle principal ya que solo debe ejecutarse una vez en la vida de un *crawler*.

Una vez entrado en el bucle la primera operación realizada es el ***fetch***.

Durante esta fase el crawler selecciona un número determinado de URLs que estén sin visitar. El número de URLs que se seleccionan depende de la configuración que se le haya puesto al *crawler* este parámetro permite configurar la rapidez con la que se ejecutaran cada una de las iteraciones.

Una vez seleccionados los enlaces que se van a seguir se procede a descargar la información correspondiente a cada uno. Esta información se preserva para la siguiente fase.

En esta segunda operación el bucle, llamada ***parse***, se procede a analizar la información descargada en la anterior operación. Se extraen metadatos y se procesa el contenido de la página web para el fin que se le haya dado al *crawler*.

Durante esta fase también se extraen nuevos enlaces presentes en las páginas que se están procesando, los cuales se añaden a la lista de enlaces para ser tenidos en cuenta en la próxima iteración.

A continuación, se procede a guardar toda la información recolectada en algún tipo de almacén. A este proceso se le conoce por ***indexación***.

En el caso de *crawlers* enfocados, existe una operación trasversal que actúa durante el resto de las operaciones. Esta operación se conoce como ***scoring***. Durante esta operación, se puede modificar el *score* dado a una URL, dependiendo de diversos factores, para conseguir su prioridad en el flujo de trabajo del *crawler*.

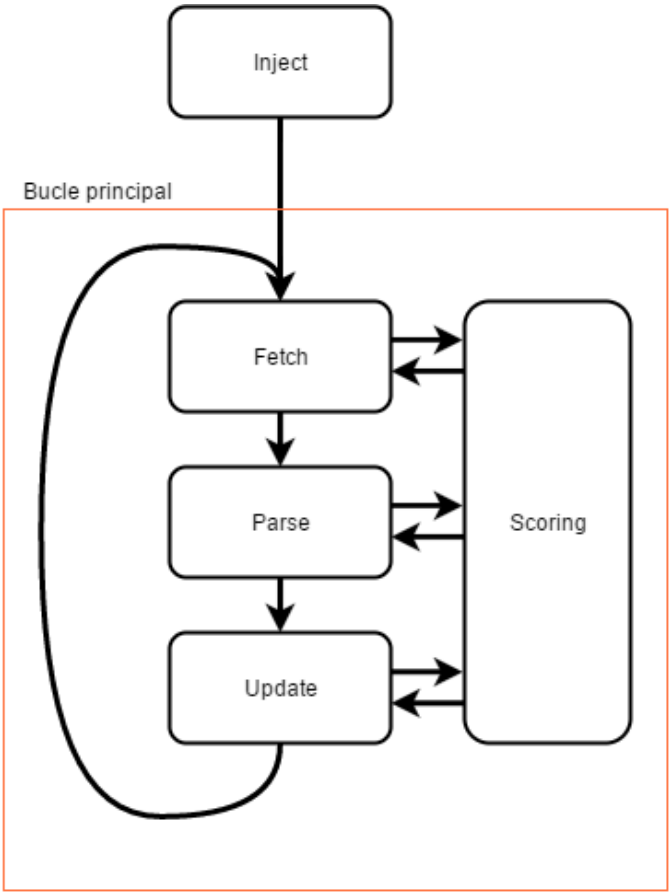

Ilustración 3

En la Ilustración 3 se puede observar el esquema del flujo habitual de un *crawler* incluyendo todas las secciones mencionadas en este apartado.

2.3.1. Educación

Cuando pones en marcha un *crawler* puedes elegir ser respetuoso con los *hosts* que visitas o no. Normalmente los *crawlers* están configurados para no hacer muchas peticiones seguidas a un sitio para no mermar su capacidad de respuesta para otras peticiones.

Se suele esperar unos segundos entre petición y petición para ser respetuoso. Por supuesto esto depende de la configuración del *crawler*. Por lo tanto, se puede ser tan respetuoso como el que diseña el *crawler* quiera.

Existe otra medida para intentar regular a los *crawlers*. Por defecto los *crawlers* deben mirar el fichero *robots.txt*. En este dichero se pueden encontrar limitaciones a los *crawlers*.

Con el fichero robots.txt se puede limitar el acceso a toda o a una parte de una web. También se pueden establecer límites a algunos *crawlers,* pero a otros no. A continuación, se pueden observar algunos ejemplos de robots.txt:

En este se indica que cualquier robot puede acceder a cualquier parte de la web.

```
User-agent: *
Disallow:
```

En el siguiente ejemplo todos los robots tendrían el acceso restringido a todas las secciones de la web.

```
User-agent: *
Disallow: /
```

El siguiente corresponde a la página web de la Casa Real de España. Se ha decidido que los *crawlers* no indexen según que partes de la web.

```
User-agent: *
Disallow:
Disallow: /_*/
Disallow: /ES/FamiliaReal/Urdangarin/
Disallow: /CA/FamiliaReal/Urdangarin/
Disallow: /EU/FamiliaReal/Urdangarin/
Disallow: /GL/FamiliaReal/Urdangarin/
Disallow: /VA/FamiliaReal/Urdangarin/
Disallow: /EN/FamiliaReal/Urdangarin/
Sitemap: http://www.casareal.es/sitemap.xml
```

Por último, podemos ver un ejemplo de una web donde se restringe el acceso de los robots excepto uno concreto. Al *crawler* de Spatineo se le permite acceder a los recursos WMS, pero al resto de robots no.

```
User-agent: spatineo
Allow: /wms/request.php
Disallow: /

User-agent: *
Disallow: /
```

2.4. Crawlers famosos

Grandes empresas hacen uso de *crawlers* para ofrecer productos y servicios. En la mayoría de casos estos son usados para alimentar un motor de búsqueda. No obstante, existen empresas que hacen uso de los *crawlers* para otros fines como recolección de información para diversos usos.

Entre los más conocidos están:

- Googlebot – Es el *crawler* del que más gente se ve beneficiada en el mundo. Como se puede averiguar por el nombre, se trata del *crawler* que usa Google.
- Baiduspider - El homólogo en Baidu del *crawler* anterior. Tiene el mismo fin, pero para alimentar el buscador de Baidu. Baidu es el buscador más usado en China.
- Bingbot – El equivalente para el motor de búsqueda de Microsoft Bing.
- Yandexbot – Usado por Yandex. Yandex es el buscador principalmente usado en Rusia y algunos países de la antigua Unión Soviética.

- TODO
- Spatineo - Un

3. Nutch

Nutch es un producto realizado Apache Software Foundation para complementarse con Apache Solr, Apache Lucene y Apache Hadoop.

Es un *crawler* genérico ampliamente extendido de código abierto. Al ser de código abierto permite modificaciones en su estructura central de manera que se puede cambiar de cualquier manera siendo posible cualquier variación.

Apache Nutch está implementado en el lenguaje de programación Java.
TODO

A parte de poder modificar el código del núcleo de Nutch, existen los denominados puntos de extensión. Los puntos de extensión son puntos del flujo de ejecución de Nutch que pueden ser ampliados mediante la implementación de *plugins*.

TODO

PUNTOS DE EXTENSION

3.1. Directorio base

La estructura de ficheros de Nutch está compuesta por carpetas con distinta finalidad.

A continuación, se va a explicar la finalidad de ellas:

- bin – Contienen dos scripts para Shell. Estos son:
  - Nutch.sh – Este fichero permite ejecutar cualquier sección de Nutch. Consultar Anexo A – Operaciones de Nutch para más información.
  - Crawl.sh – Permite la ejecución entera del flujo completo de Nutch. Se deben introducir una serie de parámetros para la configuración básica de la ejecución.
- build – Carpeta generada automáticamente al compilar Nutch con Ant. Contiene todos los ficheros necesarios para la ejecución de Nutch. Para más información se puede consultar la sección Compilación de Nutch.
- conf – Contiene todos los ficheros de configuración de Nutch. Con estos ficheros se puede alterar el comportamiento de Nutch como se desee. Se pueden editar los plugins a usar, el tipo de URLs aceptadas, etc. Se pueden consultar detalles de los ficheros de configuración de Nutch en la sección Configuración de Nutch.
- docker – Ficheros necesarios para desplegar Nutch en Docker.

- ivy – Configuración de la compilación de Nutch. Se usan Ivy como gestor de dependencias y **Ant** para compilar. Para más información consultar la sección Compilación de Nutch.
- src – Contiene todos los ficheros fuente de Nutch. Desde los ficheros originales del núcleo de Nutch hasta los plugins que vienen implementados o se implementan.
- urls – Dentro de este directorio se encuentra un único documento llamado seeds.txt. En este documento se encuentra la lista de URLs semilla.

3.2. Principales componentes

La representación del **almacén** que utiliza Nutch para su ejecución e infraestructura se llama **CrawlDB**. En este almacén se guarda información relativa a cada URL como pueden ser la dirección web, el nombre, el anchor, el estado, el score, metadatos, etc.

Existe una operación de Nutch que permite la lectura en un fichero txt del almacén anteriormente mencionado. Esta operación se llama ***readdb.***

A continuación, se van a presentar ver tres ejemplos reales extraídos de una de las ejecuciones de **GeoCrawler**.

El primero de ellos corresponde a una petición a un servicio que devuelve el código 404 de error de HTTP. Esto corresponde a un recurso no encontrado. Se puede observar en el campo ***Status*** el valor db_gone con valor numérico 3 indicando el error al intentar acceder a este recurso.

Los valores del *status* en Nutch pueden tomar los siguientes valores:

- db_unfetched – Representa un enlace que se ha almacenado en la lista, pero no ha sido procesado.
- db_fetched – Representa un enlace que ya ha sido procesado.
- db_gone – Con este estado se marcan aquellos enlaces que llevan a recursos no encontrados o que ya no existen.
- db_redir_temp – Indica que la página redirige temporalmente a otra.
- db_redir_perm – Indica que la página redirige permanentemente a otra.
- db_notmodified – Indica que una página ha sido recuperada correctamente pero no se han encontrado modificaciones.
- db_duplicate – Indica que el enlace ya existía.

http://www.opengis.net/spec/WCS_service-model_scaling+interpolation/1.0/conf/scaling+interpolation
Version: 7
Status: 3 (db_gone)
Fetch time: Tue Jun 21 20:07:53 CEST 2016
Modified time: Thu Jan 01 01:00:00 CET 1970
Retries since fetch: 0
Retry interval: 3888000 seconds (45 days)
Score: 140000.0
Signature: null
Metadata:
  anchor_context=
pst=notfound(14), lastModified=0: http://www.opengis.net/spec/WCS_service-model_scaling+interpolation/1.0/conf/scaling+interpolation
_rs_=424
anchor=

El segundo ejemplo representa un caso de éxito. Se encuentra un recurso de tipo WMS, se analiza y se indexa.

Como se puede ver, el **tipo de contenido** que hay en esa URL es de tipo application/xml. Este tipo de documento es en lo que se centra en buscar GeoCrawler.

También aparece el ***anchor***[1] y algunos otros elementos que serán explicados más adelante en esta memoria.

```
http://www.opengis.uab.es/cgi-bin/IdoneitatPI/MiraMon5_0.cgi?request=GetCapabilities&service=WMS
Version: 7
Status: 2 (db_fetched)
Fetch time: Mon Jun 06 19:46:59 CEST 2016
Modified time: Thu Jan 01 01:00:00 CET 1970
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 1.221E7
Signature: 5e59a5d17e7b9c6aa45711779e6c942b
Metadata:
  anchor_context=ática de las especies leñosas http://www.opengis.uab.es/cgi-
_pst_=success(1), lastModified=0
_rs_=963
Content-Type=application/xml
anchor=http://www.opengis.uab.es/cgi-bin/IdoneitatPI/MiraMon5_0.cgi
nutch.protocol.code=200
```

Ejemplo 2

```
http://www.opengis.uab.es/cgi-bin/Educacio/MiraMon.cgi?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.3.0 Version: 7
Status: 7 (db_duplicate)
Fetch time: Mon Jun 06 20:06:00 CEST 2016
Modified time: Thu Jan 01 01:00:00 CET 1970
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 3333000.0
Signature: 59d80766354f3ab52ab4cf8186b2a539
Metadata:
  anchor_context=ut Cartogràfic de Catalunya ( Capabilities ) WMS Base topogr
_pst_=success(1), lastModified=0
_rs_=573
Content-Type=application/xml
anchor=Capabilities
nutch.protocol.code=200
```

Ejemplo 3

3.3. Flujo principal de Nutch

El flujo de trabajo principal de Nutch sigue el esquema de un crawler general explicado con anterioridad en esta memoria. No obstante, tienen sus particularidades debido a que Nutch posee un mayor número de operaciones disponibles.

Nutch también trabaja por iteraciones, es decir, repite un conjunto de operaciones un numero especificado de veces.

3.3.1. Inject

Previo al inicio de la fase iterativa Nutch **inyecta** las URL semilla. Estas son las URLs de las cuales va a partir el *crawler.* Estas direcciones se recogen del fichero seeds.txt, el cual está localizado en la carpeta urls. La clase que implementa esta operación es `org.apache.nutch.crawl.Injector`.

3.3.2. Generate

Una vez dentro del bucle, la primera operación es **generate**. En esta operación se seleccionan los documentos para continuar con el proceso. Nutch permite Especificar en la configuración cuantos documentos van a ser seleccionados para ser procesados. Así, Nutch seleccionará los N enlaces con más score para su procesamiento.

Esta operación genera un ***segment***, este es el elemento de almacenamiento temporal que se utiliza para las distintas fases del flujo de trabajo.

En un *segment* se guarda:

- El contenido de las páginas en crudo (texto sin procesar con las etiquetas de html).
- Contenido después de pasar por la fase de *parse*.
- Metadatos descubiertos o extraidos.
- Enlaces de salida de la web (*outlinks).*
- Texto plano para la indexación.

3.3.3. Fetch

Después de haber generado un *segment* se procede a realizar la operación de ***fetch***. Durante esta operación se recupera el contenido de las páginas web

seleccionadas para ser procesadas. Extrae tanto el contenido como los metadatos de la misma.

3.3.4. Parse

El proceso de *parse* es uno de los más importantes en el proceso de *crawling* para conseguir los resultados deseados.

¿Qué es?
El proceso de *parse* no guarda una gran relación con el procedimiento homologa referida a los compiladores de lenguajes. En los dos casos se trabaja con conjuntos de caracteres con el objetivo de interpretarlos. No obstante, en el caso que nos ocupa el proceso se relaciona más con la extracción del texto perteneciente a la página que se está procesando.

Dependiendo del tipo de contenido que tienen una web, se debe usar un *parser* u otro. Cada tipo de contenido necesita que se tengan en cuenta sus peculiaridades con el objetivo de extraer la mayor cantidad de información y de manera correcta.

Un ejemplo sería una página que contiene un documento XML[2]. Para realizar el *parse* de ese tipo de documentos se utilizan analizadores específicos basados en la arquitectura DOM[3].

Cuando se creó el lenguaje XML también surgió la necesidad de procesar y manipular el contenido de los archivos XML mediante los lenguajes de programación tradicionales. XML es un lenguaje sencillo de escribir, pero complejo para procesar y manipular de forma eficiente. Por este motivo, surgieron algunas técnicas entre las que se encuentra DOM.

DOM es un conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML. Se puede utilizar DOM para manipular las páginas XHTML de forma rápida y eficiente.

Antes de poder utilizar sus funciones, DOM transforma internamente el archivo XML original en una estructura más fácil de manejar formada por una jerarquía de nodos. De esta forma, DOM transforma el código XML en una serie de nodos interconectados en forma de árbol.

El árbol generado no sólo representa los contenidos del archivo original (mediante los nodos del árbol) sino que también representa sus relaciones (mediante las ramas del árbol que conectan los nodos).

[2] e*Xtensible Markup Language*
[3] Document Object Model

Aunque en ocasiones DOM se asocia con la programación web y con JavaScript, la API de DOM es independiente de cualquier lenguaje de programación. De hecho, DOM está disponible en la mayoría de lenguajes de programación comúnmente empleados.

En la Ilustración 4 de puede ver un esquema de una jerarquía generada por un documento XHTML al ser transformado en DOM.

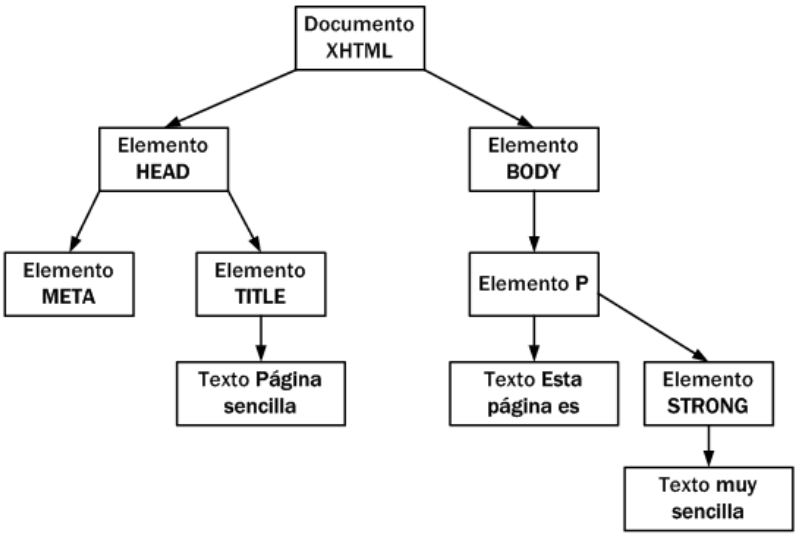

Ilustración 4

Para el desarrollo de este trabajo este tipo de documentos son los más importantes.

Nutch provee un *parser* para este tipo de contenidos. Viene implementado en *Tika Parser*. *Tika Parser* es otra herramienta de *Apache* que permite realizar el proceso de *parse* para muchos tipos de documentos entre ellos XML, HTML, PDF… .

3.3.5. Crawldb

3.3.6. LinkDb

3.3.7. Index

3.4. Map-Reduce

Hadoop

3.5. Puntos de extensión

Comentar que se implementan “implementando” la interfaz correspondiente

3.6. Compilación de Nutch

3.7. Configuración de Nutch

Nutchsite.xml
Selección de parser por tipo de contenido
URLFilter
Pim Pam

4. GeoCrawler

GeoCrawler es el nombre del *crawler* que ha sido desarrollado a lo largo de este trabajo. GeoCrawler es un *crawler* enfocado al descubrimiento de información geográfica por una parte y su posterior análisis y almacenamiento.

El objetivo de este *crawler* enfocado es el de encontrar archivos de descripción de servicios geoespaciales. Estos servicios ofrecen una gran cantidad de información.

En concreto, en este trabajo se ha tratado la búsqueda de documentos de descripción de servicios definidos por OGC[4]. Estos documentos son WPS, WCTS, WMS, WMTS, WFS, WCS y CSW.

A continuación, se va a explicar brevemente que particularidades tiene cada uno. Para complementar esto se puede consultar información más completa en Anexo C – Tipos de recursos OGC.

4.1. Documentos OGC

La OGC es una organización internacional sin ánimo de lucro que se encarga de la realización de estándares abiertos de calidad para la comunidad geoespacial.

Estos estándares son realizados a través de un proceso de consenso y están abiertos gratuitamente para cualquier persona que quiera mejorar el mundo de los datos geoespaciales.

Los estándares de OGC son usados en una gran cantidad de dominios incluidos medio ambiente, defensa, salud, agricultura, meteorología, desarrollo sostenible y muchos más.

Estos estándares son documentos técnicos que detallan interfaces o codificaciones. Los desarrolladores de software usan este tipo de documentos para construir interfaces abiertas para sus productos y servicios.

[4] Open Geo Spatial - http://www.opengeospatial.org

4.1.1. WPS

4.1.2. WCTS

4.1.3. WMS

4.1.4. WFS

4.1.5. WMTS

4.1.6. WCS

4.1.7. CSW

4.2. Modificaciones

En esta memoria se ha explicado la posibilidad de la inclusión de *plugins* en el flujo normal de Nutch. Se han incluido varias modificaciones para conseguir el comportamiento deseado del crawler.

A continuación, se va a explicar qué puntos de extensión se han usado, por qué y cómo.
En un apartado anterior se han explicado los distintos puntos de extensión a través de los cuales Nutch puede ser modificado.

Para la realización de este trabajo se han modificado los siguientes puntos de extensión:

- IndexWriter
- ParseFilter
- IndexingFilter
- Scoring Filter

4.2.1. OgcParseFilter

Vamos a empezar con el plugin que explota el punto de extensión *ParseFilter*. Esta extensión se ha utilizado por dos motivos muy importantes pero que no están relacionados entre sí.

Debido a que actúa en una sección del flujo de trabajo de Nutch que es clave para poder aprovechar y procesar en tiempo de ejecución los datos, se decidió utilizarla para detectar si el documento que se está procesando es un fichero OGC y, si lo es, extraer qué tipo de documento es y la versión del mismo.

La otra tarea que es realizada aquí es la de la extracción del *anchor* de los enlaces de salida que están contenidos en el documento que está siendo procesado, así como el contexto del mismo. Nosotros entendemos por contexto del *anchor* a los caracteres más cercanos a este. Esta información será utilizada posteriormente en el cálculo del *score.*

Como se ha comentado previamente, estos plugins se desarrollan implementando un punto de extensión, el cual viene representado por una interfaz de Java.

En este caso la interfaz que se extiende es HtmlParseFilter.

```
/**
 * Extension point for DOM-based HTML parsers. Permits one to add
 * additional metadata to HTML parses. All plugins found which
 * implement this extension point are run sequentially on the parse.
 */
public interface HtmlParseFilter extends Pluggable, Configurable {
  /** The name of the extension point. */
  final static String X_POINT_ID = HtmlParseFilter.class.getName();

  /**
   * Adds metadata or otherwise modifies a parse of HTML content,
   * given the DOM tree of a page.
   */
  ParseResult filter(Content content, ParseResult parseResult,
      HTMLMetaTags metaTags, DocumentFragment doc);

}
```

Esta interfaz tiene un método llamado *filter*. Dentro de este método se realiza la implementación que se desee. Para ello proporciona unos parámetros para ello.

Los más relevantes son el *Content* y el *ParseResult*. A través del *Content* se puede acceder al contenido en texto plano de la página. Con *ParseResult* contiene datos relevantes sobre la página que está siendo procesado como la URL.

Este objeto también contiene los metadatos, a través de los cuales vamos a conseguir la persistencia de lo que extraigamos a lo largo de todo el proceso de *crawling*. Es decir, este objeto también actúa como la salida de *filter.*

Detector de OGC

Como se ha comentado antes la primera de las tareas que se realizan en este plugin es la detección del tipo de documento OGC.

Esta detección solo se realiza si el tipo de contenido de la página es XML.

Para ellos se extrae el nodo raíz del fichero XML para analizarlo en busca de datos que nos permitan establecer de qué tipo se tratan.

A continuación, se pueden ver varios ejemplos de cabeceras para los distintos formatos de ficheros OGC.

Este es un ejemplo para WMS, en su versión 1.3.0.

```
<WMS_Capabilities
xmlns:esri_wms="http://www.esri.com/wms"
xmlns:inspire_vs="http://inspire.ec.europa.eu/schemas/inspire_vs/1.0"
xmlns:gml="http://schemas.opengis.net/gml"
xmlns:inspire_common="http://inspire.ec.europa.eu/schemas/common/1.0"
xmlns="http://www.opengis.net/wms"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3.0"
updateSequence="2619"
xsi:schemaLocation="http://inspire.ec.europa.eu/schemas/inspire_vs/1.0
http://inspire.ec.europa.eu/schemas/inspire_vs/1.0/inspire_vs.xsd">
```

Ejemplo de CSW con versión 2.0.2.

```
<csw:Capabilities
xmlns:csw=http://www.opengis.net/cat/csw/2.0.2
xmlns:gml=http://www.opengis.net/gml
xmlns:gmd=http://www.isotc211.org/2005/gmd
xmlns:ows=http://www.opengis.net/ows
xmlns:ogc=http://www.opengis.net/ogc
xmlns:xlink=http://www.w3.org/1999/xlink
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance”
xmlns:inspire_ds="http://inspire.ec.europa.eu/schemas/inspire_ds/1.0"
xmlns:inspire_com=http://inspire.ec.europa.eu/schemas/common/1.0
version="2.0.2"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd">
```

Ejemplo de WFS con versión 1.1.0.

```
<wfs:WFS_Capabilities
xmlns:wfs=http://www.opengis.net/wfs
xmlns:ogc=http://www.opengis.net/ogc
xmlns:gml=http://www.opengis.net/gml
xmlns:ows=http://www.opengis.net/ows
xmlns:xsi=http://www.w3.org/2001/XMLSchema-istance
xmlns:xlink=http://www.w3.org/1999/xlink
xmlns:ComarcasAgrarias=http://wms.magrama.es/sig/WFS/ComarcasAgrarias/
wfs.aspx version="1.1.0"
xsi:schemaLocation="http://www.opengis.net/gml
http://schemas.opengis.net/gml/3.1.1/base/gml.xsd
http://www.opengis.net/ogc
http://schemas.opengis.net/filter/1.1.0/filter.xsd
http://www.opengis.net/ows
http://schemas.opengis.net/ows/1.0.0/owsAll.xsd
http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
```

Ejemplo WCS con versión 2.0.1.

```
<wcs:Capabilities
xmlns:wcs=http://www.opengis.net/wcs/2.0
xmlns:ows=http://www.opengis.net/ows/2.0
xmlns:ogc=http://www.opengis.net/ogc
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:xlink=http://www.w3.org/1999/xlink
xmlns:gml=http://www.opengis.net/gml/3.2
xmlns:gmlcov=http://www.opengis.net/gmlcov/1.0
xmlns:swe=http://www.opengis.net/swe/2.0
xsi:schemaLocation="http://www.opengis.net/wcs/2.0
http://schemas.opengis.net/wcs/2.0/wcsAll.xsd "
version="2.0.1">
```

Ejemplo de WPS con versión 0.4.0.

```
<wps:Capabilities
xmlns:wps=http://www.opengeospatial.net/wps
xmlns:ows=http://www.opengeospatial.net/ows
xmlns:xlink=http://www.w3.org/1999/xlink
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="0.4.0"
xsi:schemaLocation="http://www.opengeospatial.net/wps..\wpsGetCapabili
ties.xsd">
```

Como estas existen otras cabeceras distintas dependiendo del tipo de documento. Analizándolas se puede saber de qué tipo se tratan.

Anchor y contexto

Anchor

4.2.2. OgcIndexingFilter

4.3. Algoritmo búsqueda

4.4. Tesauro

4.5. Modificaciones extra

Robots.txt ignorado
Content length
Filter de url modificado
…
Acelerador del fetch

5. Resultados

6. Futuro

7. Bibliografía

A. Anexo A – Operaciones de Nutch

B. Anexo B – Manual de GeoCrawler

C. Anexo C – Tipos de recursos OGC