TALLER 1 Descubrimiento de información utilizando crawling y Utilización de fuentes de sindicación/suscripción, RegEX y XQuery

Erik Baumert, Carlos Montealegre, Areli Moreno
Análisis de Información sobre Big Data
Universidad de los Andes, Bogotá, Colombia
{am.morenof, cj.carvajal, e.baumert }@uniandes.edu.co

Fecha de presentación: marzo 6 de 2017

Tabla de contenido

1 Introducción

Este documento describe brevemente los métodos y tecnologías utilizadas para el desarrollo de los objetivos del taller y el análisis de los resultados.

2 Descubrimiento de información utilizando *crawling*

2.1 Métodos y Tecnologías

La figura 1 presenta el diseño de arquitectura para la construcción del Directorio de personas vinculadas a la Universidad de los Andes. El Worker y el Web Server se comunican a través de Servicios REST y se ejecutan independientemente, aunque están en la misma máquina, en este caso el Web Server funciona como Proxy. El proyecto se desarrolló principalmente en lenguajes de programación JavaScript y Python. Las aplicaciones en Worker y Web Server están desarrolladas en el framework Flask. El servicio *crawling* usa las librerías Scrapy y Request de Python. Adicionalmente para ejecutar los procesos Flask se hizo uso de la herramienta *virtualenv* que está diseñada para poder desarrollar proyectos en Python trabajando con diferentes versiones de paquetes que, de otra forma, entrarían en conflicto. Por otro lado, la interfaz gráfica se desarrolló con los frameworks AngularJS y Bootstrap, haciendo usó el generador de proyectos de Angularjs, Yeoman.

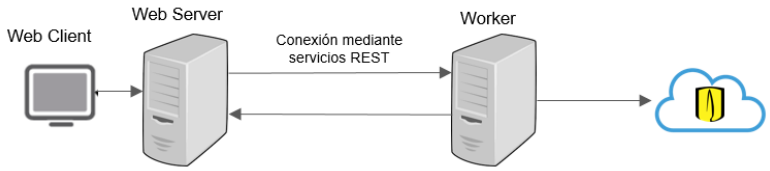

Figura 1. Arquitectura Crawling

2.2 Algoritmo básico

El cliente envía la solicitud de búsqueda a través de la interfaz gráfica pulsando el botón "Buscar", el Web Client reenvía la petición GET al Web Server y este solicita al Worker el proceso de *crawling*, el Worker ejecuta y retorna la información al web server quien reenvía la información capturada en un archivo en formato JSON para que finalmente pueda ser presentada en la interfaz gráfica. El Worker ejecuta el *crawler* una vez que el cliente realiza la petición de búsqueda, el *crawler* vuelve a ejecutarse solo si el cliente lanza la petición pasados por lo menos dos minutos de la anterior, de lo contrario el Worker envía el último archivo almacenado. El algoritmo utilizado en el *crawling* se presenta en la figura 2.

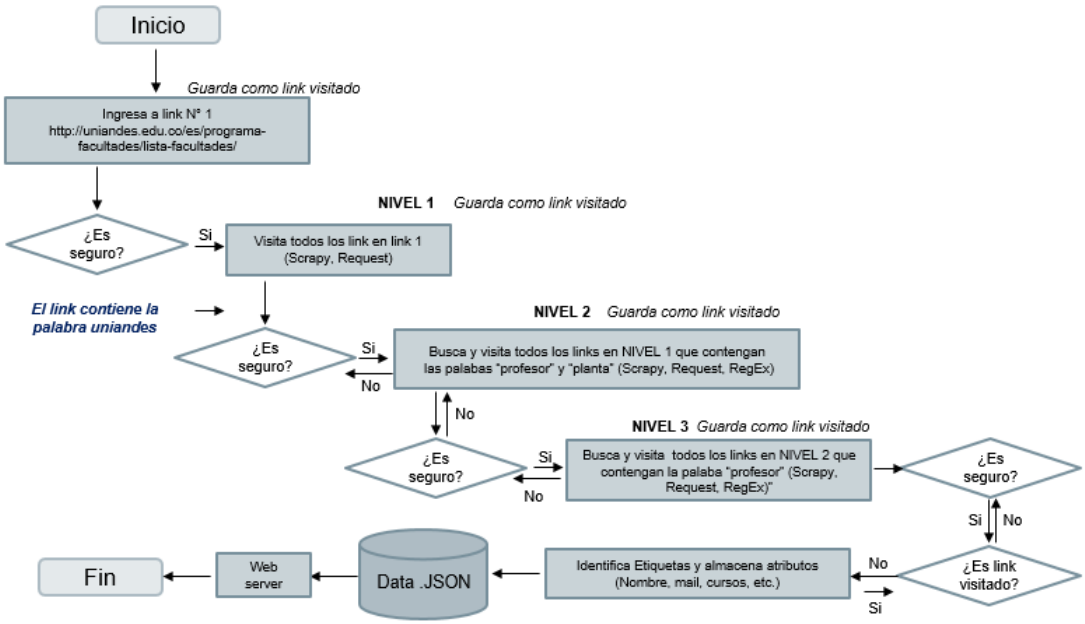

Figura 2. Diagrama Algoritmo crawler Nephila

3 Utilización de fuentes de sindicación/suscripción, RegEx y XQuery

3.1 Métodos y Tecnologías

La figura 3 presenta el diseño de arquitectura para el ejercicio 2. La tecnología usada en este ejercicio es similar a la descrita en la sección 2.1. En este caso el Web Client envía una petición al Web Server quien funciona en este caso también como Proxy, este reenvía la petición POST al Worker quien realiza el proceso de suscripción al RSS, y recibe un *string* con el que efectua el filtrado de los datos del RSS usando RegEx, de acuerdo a la petición para finalmente devolver la respuesta.

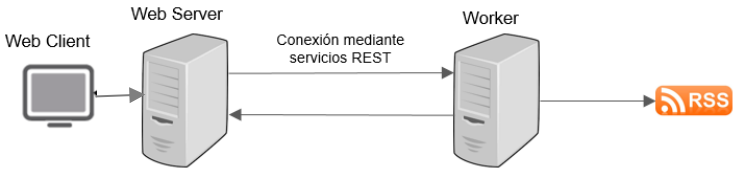

Figura 3. Arquitectura fuentes RSS

3.2 Algoritmo básico

El usuario realiza por medio de la interfaz gráfica una búsqueda a partir de tres campos “Título”, “descripción” y “Categoría”, entonces el Web Server realiza la petición POST redirigiendo los tres elementos en formato JSON al Worker. Este último recibe la petición y realiza la búsqueda en los RSS de Mashable y Google News de acuerdo a tres categorías: Entretenimiento, Tecnología y Negocios. Cuando el cliente envíe una búsqueda, el Worker visitará y traerá la información de cuatro principales fuentes Mashable, Google News Entertaiment, Google News Bussines y Google News Tecnology. Sin embargo, el Worker consultará las fuentes RSS solo si han pasado por lo menos dos minutos de la creación del último archivo almacenado, en caso contrario el Worker envía al Web Server la información que se encuentre almacenada. En el siguiente diagrama se encuentra el algoritmo básico que realiza el Worker:

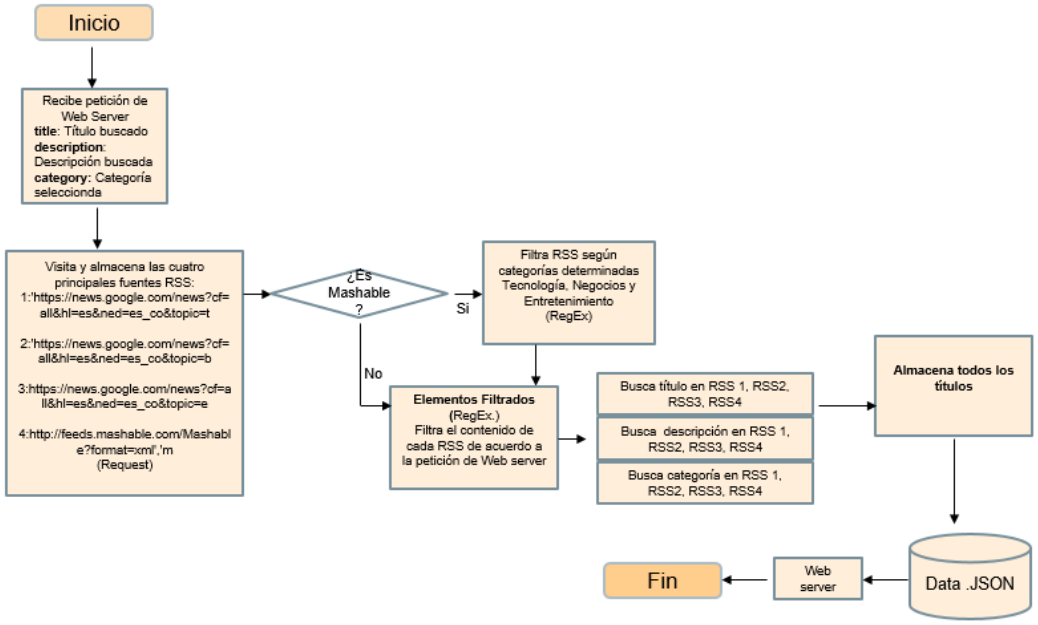

Figura 4. Algoritmo búsqueda en fuentes RSS

4 Expresiones XQuery y RegEx usadas

En Nephila como llamamos al crawler se usaron expresiones RegEx como: `re.search(r'.*profesores',next_page),re.search(r'.*uniandes.*',next_page),re.search(r'.*administracion.*',next_page).` Y se usó XML Path Language (Xpath) en casos como:
`response.xpath('//div[@id="collapseTwo"]').css('li::text').extract().`

En el ejercicio de búsqueda en fuentes RSS , se usaron expresiones como:

```
re.findall(r'<item>(.*?)</item>',str(data))
re.search(r'.*<category>Business</category>.*',item)
re.search(r'.*<category>Tech</category>.*',item)
re.search(r'.*<category>Entertaiment</category>.*',ítem)

re.research(r'.*<title>.*+title+r'.*<title>.*,ítem)
re.research(r'.*<title>.*+description+r'.*<title>.*,ítem)
re.research(r'.*<title>.*+category+r'.*<title>.*,ítem)
```

En AngularJS se usaron expresiones como `s.replace(/[\[\]']+/g, ''), s.replace(/\\t+/g, '')` y `s.replace(/\s\s+/g, ' ').` Con el objetivo de editar la información

5 Forma de acceso a la aplicación web que muestra los resultados

Para realizar el proceso completo se deben ejecutar las aplicaciones `app.py` de Web server y Worker respectivamente. La aplicación Angularjs que contiene la interfaz gráfica, se ejecuta a través del comando de Yeoman `grunt serve` desde la consola Unix. El comando abre la publicación en la url: http://172.24.101.144:9000. Allí existen dos páginas web, una con el Directorio de personas vinculadas a la Universidad de los Andes y la otra con las fuentes RSS de Mashable y Google News.

En la página del directorio se encuentra el botón "Buscar" para iniciar la aplicación y campos de texto asociados a facultad, nombre, correo electrónico, títulos obtenidos, currículo y cursos ofrecidos. En estos campos el usuario pude ingresar cualquier palabra y la página retornará solo el contenido asociado a su búsqueda. ¡Por otro lado, en la página RSS existe e botón "Filtra noticias!" que envía la petición, existen dos campos de Título y descripción para que el cliente ingrese el texto que desea encontrar, teniendo en cuenta que esta aplicación es *case sensitive*, y botones asociados a las categorías.

6 Análisis de resultados

Algunas de las dificultades en el proceso fueron:

- Aprender y hacer uso de nuevas herramientas como Flask y AngularJS
- Trabajar con Xquery, configurar las librerías
- Identificar y entender la arquitectura para solucionar el problema
- Manejo del Encoding para guardar fuentes RSS

Algunos de los aspectos que se lograron fueron:

- Aprender un poco de los nuevos frameworks
- Implementar una arquitectura desacoplada
- Realizar un crawler eficiente manejando el tema de caché y en una interfaz moderna
- Aprender y utilizar expresiones RegEx para el filtro de RSS
- Trabajar con fuentes con diferentes estructuras

Desde el punto de vista de la información entregada al usuario:

- Se logró construir un directorio de profesores de dos facultades con información básica
- La respuesta al servicio de *crawler* se demora menos de 3 segundos y el proceso de filtrado se da en menos de 1 segundo.
- Se presenta la información de una forma visualmente fácil e intuitiva.

Algunos de los problemas encontrados para hacer un mejor trabajo de entrega:

- No fue posible abarcar todas las unidades académicas debido a la diversidad de estructuras
- No fue posible guardar el caché de Mashable debido a un problema de encoding, por ello se disminuye la eficiencia.

Mejoras, retos:

- Implementar los filtros XQuery para los filtros de las fuentes RSS
- Hacer implementaciones individuales para cada unidad académica faltante
- Incluir personal diferente a los profesores
- Incluir imágenes de las personas
- Permitir descarga de reportes de búsqueda

Finalmente, si se quisiera construir el directorio completo de la Universidad, incluyendo las publicaciones de los profesores, de la librería o unidades de investigación y responsables de los órganos de gobierno y de administración de la Universidad, la solución propuesta debería poder aumentar la cobertura del *crawler* en términos de unidades académicas y atributos y, prepararse para procesar otras fuentes con estructuras distintas.