Facultad de Ingeniería

Universidad de la República - Uruguay

Recuperación de Información y Recomendaciones en la Web (WEBIR)

2015

PROYECTO FINAL
Grupo 5

Integrantes del grupo:

Juan José Cardarello	5.330.612-2
Guillermo Leopold	5.485.678-6
Felipe Garcia	4.521.936-7
Ignacio Chiazzo	4.995.790-5

Índice

Introducción

Cada vez que se aproxima un nuevo período de pases en el fútbol mundial ocurre un fenómeno muy especial: los rumores sobre posibles traspasos de jugadores. Algunos de ellos se concretan, pero su gran mayoría quedan en eso, sólo rumores.

Es por eso que en este proyecto nos enfocamos en buscar, en páginas relacionadas al mundo del fútbol y las transferencias, todas las noticias (o la gran mayoría) que involucren algún rumor de traspaso, y mostrarle al usuario una tabla con el resumen de todos esos rumores.

Optamos por plantear esta propuesta debido a que nos resultó de gran interés yaque en la actualidad, durante y en los meses previos a la apertura del mercado de fichajes en el fútbol, la cantidad de artículos relacionados con posibles traspasos es enorme, siendo éste uno de los principales temas de la mayoría de los sitios dedicados al fútbol, debido al atractivo que representan el mismo para el público y la gran cantidad de noticias que se publican con el fin de vender.

En el presente documento se aborda una solución de un prototipo de obtención de rumores de las principales ligas de fútbol. Se brinda también una aplicación Web para que los usuarios puedan ver los distintos rumores captados de un conjunto selecto de páginas, con la posibilidad de filtrar tanto por página, como por jugador o club y hasta por fecha.

Enfoque de solución

El sistema de obtención de las distintas noticias fue desarrollado utilizando el lenguaje Python y el framework Scrapy [1]. También se utilizó una Base de Datos no relacional (MongoDB[2]) para persistir los datos, y para el frontend se utilizaron AngularJS 1.2.18 [3], HTML5, CSS3 y Jquery 1.11 [4].

La solución que planteamos para resolver este problema consiste en el siguiente procedimiento:

Nos conectamos a una api [5], de la cual sacamos datos sobre equipos y sus respectivas plantillas, las persistimos en una base de datos local. Decidimos persistir esta información porque estos datos son relativamente rígidos, esto es, cambian una vez cada 6 meses aproximadamente, reduciendo así el costo en tiempo que implica conectarse con otro servicio externo, así como en ancho de banda. Paralelamente, mediante scrapy hicimos scraping de dos fuentes distintas de noticias, lo cual es expandible dado que el formato en el que se almacenan las noticias obtenidas es el mismo (JSON genérico), y con los resultados de ambas tareas paralelas hacemos un trabajo de procesamiento de lenguaje para encontrar la siguiente relación en cada una de las noticias: hallar el nombre de equipos comparándolos con los persistidos en la base de datos, hallar nombres de jugadores que pertenezcan a aquellos clubes que hallamos en el primer paso para ahorrar procesamiento y no buscar en toda la bd, y asumimos que el resto de los equipos mencionados son los que están interesados en contratarlo. Para futuro, se puede mejorar el procesamiento de texto de forma de detectar ciertas expresiones de lenguaje normalmente utilizadas en este tipo de noticias, que indican interés o posibilidad de fichaje de un club con un jugador, con el fin de disminuir los falsos positivos que ocurren en los clubes destino.

Diseño e implementación

El diseño del sistema corresponde a un proyecto Python, implementado con el framework Scrapy [6], el cual tiene como principales componentes a los Items [7] y Spiders [8].

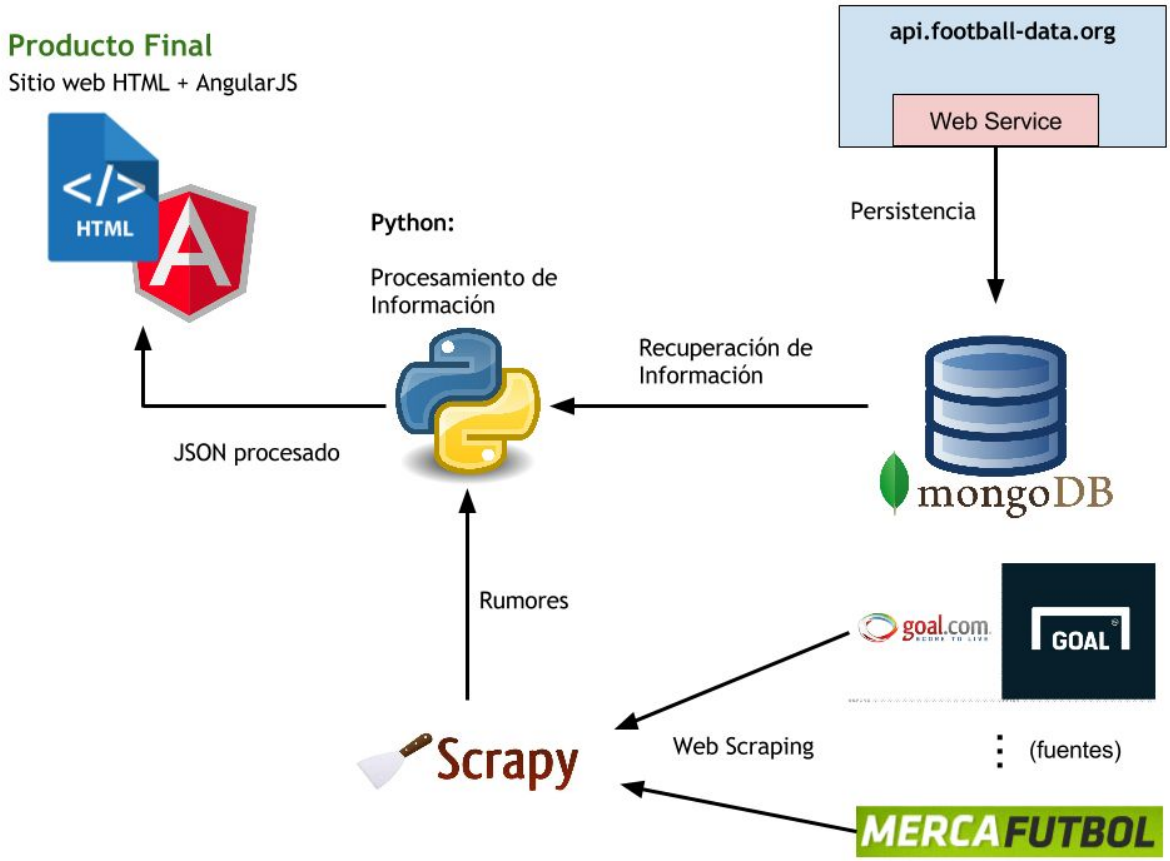
Los Items son los contenedores de la información resultante de la extracción. Es un DSL mediante el cual se define la estructura de los datos a extraer.

Los Spiders son clases, piezas de código y módulos escritos con el fin de descargar el código fuente de la URL objetivo, y recorrer la información del sitio web descargado, para luego depositarlo en los Ítems.

Decidimos persistir los datos mencionados en una base de datos NoSQL, ya que persistimos en formato JSON los rumores, y las operaciones involucradas son : get de un rumor (se requiere de la información completa del objeto) y get de todos los objetos rumor. Si bien no conocíamos las bases de datos NoSQL observamos que éstas son muy útiles para este tipo de casos en donde se requiere persistir en formato json, y se trata de evitar los joins entre objetos dada las características de las operaciones a la base de datos.

Luego, la web accede a los JSON obtenidos como resultado del procesamiento de las noticias obtenidas, para luego presentar la información en la tabla web y hacerla visible al usuario.

Arquitectura



Tecnologías utilizadas

Se utilizaron para el desarrollo de la lógica del programa el lenguaje python y la herramienta de Scrapy, mientras que para la persistencia se utilizó mongoDB, particularmente se utilizó el IDE recomendado por mongo "Robomongo". En cuanto a Frontend se utilizaron las tecnologías HTML 5, CSS3, AngularJS y JQuery.

Pruebas

Para testear nuestro prototipo elegimos 2 páginas relacionadas a las transferencias. Ellas son goal y mercafutbol. Para ello creamos 2 scripts python (uno para cada página), donde cada uno depende del código HTML de la página.

Por ejemplo, para el caso de Goal, el título de la noticia se encuentra dentro de un , que a su vez se encuentra dentro de un<div class=”rumour-content column”>, y este está en un , y a su vez todos los rumores están dentro de un ; pero no es igual para el caso de las otras páginas, por lo que los scripts varían según el código HTML de estas.

Al ejecutar los scripts sobre las respectivas páginas, obtuvimos un .json que contiene las noticias, donde cada noticia está formada por un título, la url a la que pertenece la noticia, el nombre del jugador involucrado, el club en el que éste juega y los destinos que se rumorean.

Como pruebas realizadas sobre nuestro sistema, decidimos acceder a las 2 páginas anteriormente mencionadas y recorrer manualmente algunos artículos con el fin de encontrar algunos que involucren clubes de las ligas importantes sobre las cuales trabaja nuestro prototipo, posteriormente ejecutamos nuestro scripts tanto de scraping como de procesamiento y verificamos que en la tabla de la web se encuentre las noticias que habíamos previamente detectado manualmente y además que estén procesadas correctamente.

Luego, realizamos una nueva prueba, la cual consistió en setear el parámetro para el script de python que define el número de páginas sobre las webs seleccionadas, a scrapear, se fue a cierta pagina (supongamos la página 3) y se procedió realizando una prueba de características similares a la anterior.

Dificultades

- Es muy común que en las noticias se utilice abreviaciones y jergas en el lenguaje del fútbol, el cual muchas veces se repiten, y en otras no, por lo que tuvimos que tener en cuenta muchas abreviaciones y jergas del lenguaje, para procesar la noticia. Esto, por otra parte representó que en algunas noticias tanto nombre de jugadores como nombre de clubes no sean detectados por abreviaciones que se emplean en lo mismo como puede ser un apodo para un jugador o un diminutivo en el nombre del club
- Se requiere de un gran trabajo de procesamiento de lenguaje natural para el desarrollo de la aplicación.
- Acceso a información actualizada respecto a las plantillas de los clubes y de alguna api que proporcione un servicio gratuito para el consumo de esta información, encargándose así, un tercero de mantener actualizados los planteles y liberándonos de esa tarea. Esto implicó incluso, el contacto vía email con el desarrollador de una api alemana (la cual utilizamos), la cual nos permitió acceder a la información deseada así como a funcionalidades que actualmente no están publicadas en la documentación.
- Fue necesario investigar sobre BD no relacionales para guardar las plantillas de los distintos cuadros, ya que se adecúan más al tipo de datos a guardar (JSON), y ninguno tenía conocimiento previo sobre estas.

Conclusiones

Luego de realizar las pruebas y medir los tiempos con los que se obtienen las respuestas, podemos concluir que Scrapy es una herramienta muy poderosa cuando se trata de navegar sobre estructuras HTML de sitios web y obtener información de ellas.

Otra conclusión que podemos sacar es que las bases de datos no relacionales son muy útiles a la hora de persistir datos en formato JSON, ya que de manera muy fácil y rápida se pueden guardar datos como también acceder a ellos.

Uno de los problemas que ocurre en este tipo de aplicación es que, en algunos casos, para hacer referencia a un mismo club existen varias maneras (abreviaciones, sobrenombres, con o sin tilde, etc.). Por ejemplo, si en la noticia aparece el nombre “Man Utd”, se debe considerar que se está haciendo alusión al Manchester United. Esto ocurre en el caso tanto de los clubes como de los jugadores en algunos casos, y hay que tenerlo presente a la hora de buscarlos en la noticia.

Otro problema es que cada página tiene su propio código HTML, diferente de los demás, por lo que hay que generar un script python diferente por cada una de ellas, lo que conlleva un costo asociado.

Dado que se obtuvo un prototipo satisfactorio, cumpliendo con nuestros objetivos, podemos concluir que se puede desarrollar una aplicación de gran porte, tomando como base dicho prototipo.

Trabajo a futuro

Se evaluó la posibilidad de ponderar la probabilidad de las transferencias en base a los términos utilizados en el artículo ya que existe cierto conjunto de términos muy común en dicho ámbito (inminente, probable, set to en inglés, etc), los cuales se pueden utilizar para otorgar una mayor o menor probabilidad de ocurrencia al traspaso; sin embargo no pudimos llegar a implementar esto en el prototipo por motivos de longitud de proyecto, pero queda documentado como una posibilidad a futuro para extender la funcionalidad y utilidad del sistema propuesto.

Se podría extender los sitios fuentes de noticias, así como también los datos de ligas y jugadores de fútbol.

Uno de los puntos importantes de trabajo a futuro es mejorar el algoritmo de procesamiento, de forma de optimizarlo y poder reducir el tiempo de obtención de datos, y una mejora en la cantidad y calidad de las noticias.

Se puede ampliar las cantidad de ligas, no solo para las principales ligas de europa sino para todas las ligas del mundo.

También la posibilidad de que la web se actualice cada cierto periodo de tiempo, realizando nuevamente el scraping de la web y actualizando la tabla de rumores de forma automática, y no manual como se encuentra actualmente.

Manual de usuario

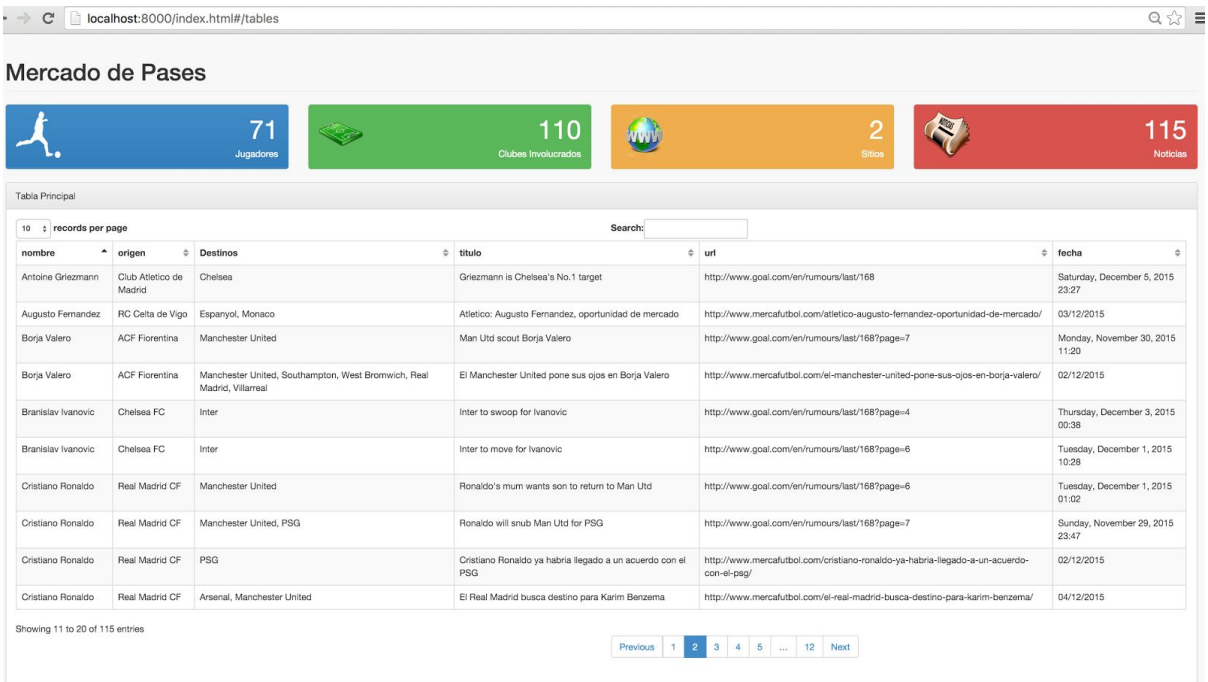
Cuando el usuario ingresa a la página, se le muestra una tabla con todos los últimos rumores obtenidos de las páginas ya mencionadas, y este tiene la opción de ordenarlo alfabéticamente por cualquiera de los parámetros de la tabla, así como también puede filtrar los rumores obtenidos.

Para ordenar alfabéticamente (tanto descendente como ascendentemente), se debe clickear en la columna correspondiente a la información por la que se desea ordenar.

Por otra parte, para realizar filtrado, tanto por clubes, como por jugadores o página de la noticia, se debe escribir el texto deseado para realizar el filtro en el campo search que se encuentra justo por encima de la tabla.

También es posible seleccionar la cantidad de entradas en la tabla a mostrar y debajo se encuentra el número de páginas y la posibilidad de avanzar en estas.

Arriba de la tabla se puede apreciar diferentes estadísticas cuantitativas, correspondientes a la cantidad de jugadores involucrados en los rumores, cantidad de clubes, sitios de los que se obtuvo noticias y número de noticias.



Referencias

[1] Scrapy 1.0 framework, scrapy.org, consultado el 12 de Noviembre de 2015.
[2] Base de datos No SQL, Mongo DB, www.mongodb.org, consultado el 22 de Noviembre de 2015.
[3] Angular JS, angularjs.org. consultado el 8 de Noviembre de 2015
[4] Jquery, jquery.com, Consultado el 8 de noviembre de 2015
[5] football-data API, http://api.football-data.org/ , Consultado el 8 de noviembre de 2015
[6] Tutorial proyecto en Scrapy, http://doc.scrapy.org/en/latest/intro/tutorial.html, Consultado el 12 de noviembre de 2015
[7] Items, http://doc.scrapy.org/en/latest/topics/items.html, Consultado el 12 de noviembre de 2015
[8] Spiders, http://doc.scrapy.org/en/latest/topics/spiders.html, Consultado el 12 de noviembre de 2015