Less Expensive

Informe para el obligatorio de Recuperación de Información y Recomendaciones en la Web edición 2017

Facultad de Ingeniería, UDELAR

Tutora:

Libertad Tansini

Estudiantes:

Camila Serena 5.258.250-9

Juan Serra 4.758.444-1

Diego Irigaray 4.680.516-7

Ignacio Prandi 4.473.654-2

Andres Veiro 4.755.272-3

Índice

1. Introducción

Este trabajo se realiza en el marco del proyecto final del curso de Recuperación de Información y Recomendaciones en la Web edición 2017 [1] de la Facultad de Ingeniería de UDELAR.

El mismo trata de vincular los conocimientos aprendidos a lo largo del curso en un problema real, combinando distintas fuentes de información e incorporarlas en una presentación más sencilla para el usuario. Para esto se utilizan distintas técnicas y algoritmos de recuperación de datos como por ejemplo scrapping de páginas web y consulta a distintos servicios expuestos como APIs.

2. Problema

El problema a resolver es investigar e implementar el prototipo de una aplicación cuya funcionalida principal es buscar los precios de productos comerciales en distintas tiendas en base a su código de barras.

En mucho casos al momento de comprar un producto en una tienda nos preguntamos "*¿Es este es el mejor precio?*" Si se conoce el nombre del producto, sería posible buscarlo en distintas tiendas para comparar y finalmente evaluar si el precio fijado es razonable realmente.

Con el fin de resolver el dilema de una manera automatizada se propone usar la cámara del celular para lograr escanear el código de barra y luego que una aplicación busque en un conjunto de tiendas el precio de dicho producto. Generando una lista de precios de dicho producto para poder tomar la decisión.

Previo al prototipo se evaluó la posibilidad de realizar la operativa mencionada, para esto investigamos algunos conceptos y fuentes de información para ver si a partir del código de barra que está en el producto o envoltorio podemos llegar al precio en alguna tienda.

3. Conceptos previos

3.1 Códigos de Barra

Son códigos basados en la representación de un conjunto de líneas paralelas de distinto grosor y espaciado que en su conjunto contienen una determinada información. Lo codificado es una cadena de caracteres, la cual se asocia a un artículo en particular y permite reconocerlo de forma única, global y no ambigua en algún punto de una cadena logística(por ejemplo para los productos de un fabricante puntual). Las aplicaciones del código de barras cubren una gran variedad de actividades, tanto en

industria, comercio, instituciones educativas, gobierno, etc. Esto se da porque es un método sencillo, rápido y eficiente de captura de información.

Nos encontramos con distintos tipos de códigos de barras, en un principio podemos diferenciar entre códigos de barra de una y dos dimensiones. Siendo los primeros los más utilizados para la identificación de artículos en el comercio. A su vez también nos encontramos con diversas codificaciones utilizadas, nos centraremos en la codificación U.P.C y la E.A.N que son los estándares más utilizados para la identificación de productos comerciales.

U.P.C (Universal Product Code) [12]

E.A.N (European Article Numbering) [13]

3.2 Algoritmos: Web Scraping

Web scraping es una técnica que consiste en extraer información de páginas web. Esta tarea normalmente es realizada de forma automática mediante el uso de programas de computadora que se encargan de extraer la información deseada del código de un sitio web para su posterior uso. En primer lugar el programa debe descargar la página web, de forma similar a como haría un navegador para finalmente extraer de esta la información necesaria.

4. Funcionalidades

En esta sección explicaremos más a fondo las funcionalidades que se implementan.

Primero se requiere una aplicación móvil, preferentemente en Android. La principal razón de esto es que el uso de la aplicación se dará de forma móvil, ya que será utilizada una vez encontrado un producto que se quiera comparar su precio y para esto lo más intuitivo es incluirla en un dispositivo móvil, como ser una tablet o un celular. Por otro lado tenemos que existen aplicaciones que funcionan como lectores de código de barras y traducen las imágenes tomadas por la cámara a su valor real. De

esta forma con solo un celular el usuario será capaz de escanear el código de barras y realizar la búsqueda del producto que quiera comparar.

Una vez escaneado un código de barras, se desea traducirlo al producto que este pertenece. Dado que no todos los códigos de barras son únicos para un producto, es decir, un producto puede cambiar el código de barras que tiene asociado en distintos puntos de la cadena logística, se deberá buscar en distintas fuentes y se podrá obtener distintos productos. En este caso se listará al usuario los distintos productos con el fin de que este identifique cual es el que realmente se desea buscar. De esta selección se obtiene el nombre del producto.

Luego, el usuario tendrá la posibilidad de buscar dicho artículo en distintas tiendas online con el fin de devolver al usuario los posibles lugares donde el artículo buscado puede ser adquirido. Dado la gran cantidad de tiendas, se decidió reducir la búsqueda a un conjunto acotado de tiendas, teniendo en cuenta algunas del medio local así como también alguna extranjera.

Para las tiendas locales, se tomó como referente MercadoLibre, esto es porque podemos encontrar una gran variedad de artículos y al ser una tienda en la cual los precios son fijados por distintos vendedores, nos encontramos con un buen indicador de precio de los artículos a nivel local. Además también se utilizara los supermercados Tata y Tienda Inglesa estos últimos dos se lo eligió por que publican los precios de los productos en la web de cada uno respectivamente.

En el caso de las tiendas extranjeras optamos por Amazon, esta se encuentra ubicada en E.E.U.U y es una empresa referente en el comercio online internacional. Una de las razones por la cual se optó por incluir una tienda extranjera es para que el usuario pudiera comparar la diferencia de precios con las tiendas locales y así poder definir por sí mismo si le es más conveniente comprar el artículo buscado localmente o importarlo pagando los respectivos costos de importación.

5. Arquitectura y Diseño

El estilo arquitectónico utilizado es cliente-servidor dado que un trabajo a futuro sería poder hacer un cache de los resultados para alivianar el procesamiento de las consultas de lado del servidor.

El cliente se realizó fino, con la lógica de presentar la información y consumir servicios expuestos por el servidor.

El servidor en cambio utiliza una arquitectura en dos capas.

1. Capa de servicios: Implementa los servicios que utiliza la capa de presentación
2. Capa lógica: Realiza la búsqueda del articulo por el código y luego la búsqueda en las distintas tiendas.

En la Figura 1 se ilustra con un diagrama de componentes de alto nivel cómo se comunican y la relación con sistemas externos.

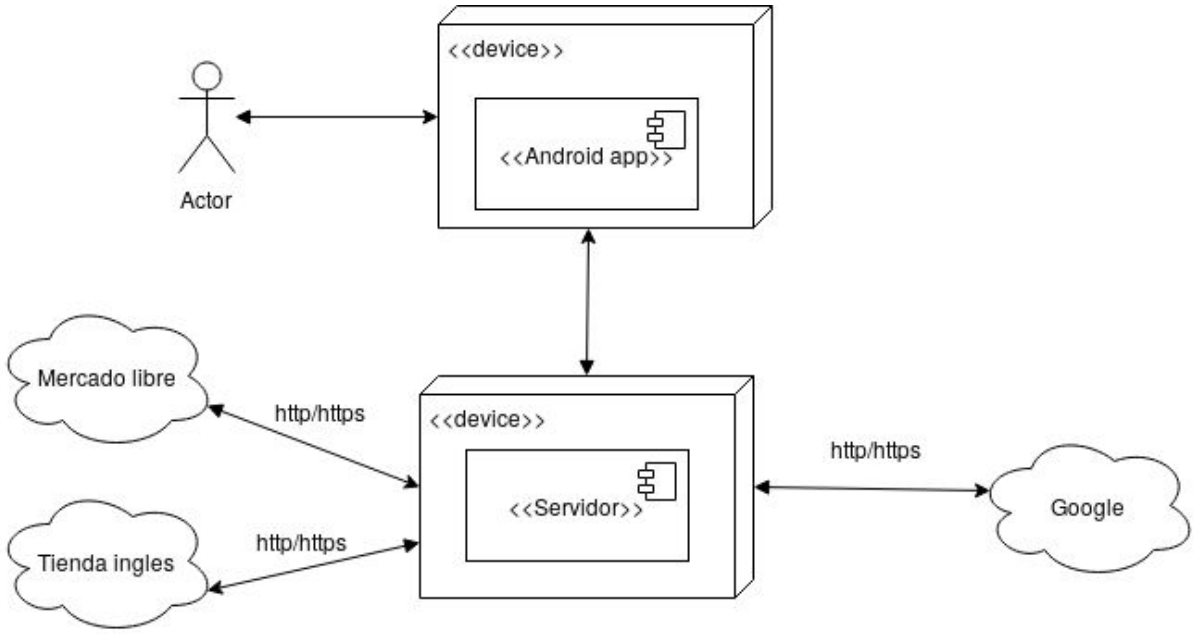

Figura 1: Diagrama de componentes y sistemas externos.

6. Implementación

La implementación se hizo en 4 etapas:

1. Implementación del Frontend
2. Implementación del Backend
3. Traducción del código de barras
4. Métodos de búsqueda de información.

6.1 Tecnologías utilizadas

Al tener libertad de elegir en qué tecnologías desarrollar, se buscó elegir las que más facilidades nos pudieran brindar para el proyecto en sí. Cualquier tecnología actual cumple con los requisitos pero se decidió usar algunas en las que el equipo no tenía experiencia para para poder evaluarlas.

6.2 Implementación del Frontend

Para la implementación del Frontend se decidió utilizar una tecnología capaz de ejecutar en Android, iOS y otros. A estas tecnologías se las llaman híbridas. Con el fin de poder acercarnos a la mayor cantidad de usuarios con el mínimo re trabajo por plataforma.

Para esto se utilizó IONIC 2 [2], dicha plataforma permite realizar aplicaciones móviles para Android, IOs y Windows Phone utilizando Angular [3] para realizar las

vistas y Apache Cordova [4] para hacer uso de las funcionalidades propias del dispositivo.

Además permite hacer pruebas en dispositivos a través de una aplicación provista por la plataforma lo que facilita la etapa de verificacion y depuracion.

Las Figuras 2, 3 y 4 son capturas de esta capa de presentación. En la Figura 2 se observa la página principal de búsqueda. En la Figura 3 muestra los resultados para un código de barras ingresado. Por último en la Figura 4 se observa los precios en algunas tiendas. Los datos que se muestran son ejemplos realizados en la etapa de verificación.

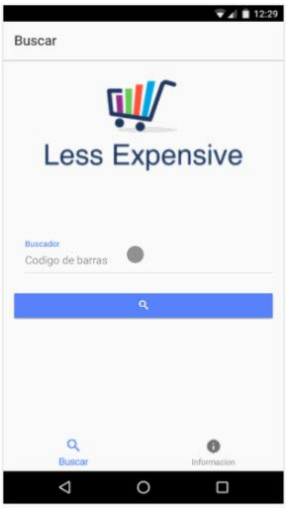

Figura 1: Vista inicial.

Figura 2: Vista de búsqueda.

Figura 3: Vista en tiendas

6.3 Implementación del Backend

6.3.1 Python/Django:

La elección de Python como lenguaje de programación se debe a que cuenta con múltiples librerías sencillas de utilizar para realizar el web scraping y también resolver el llamado a las distintas APIs que exponen las tiendas.

Entonces, para desarrollar el *Backend* se utilizó Django, un framework para el desarrollo web en alto nivel sobre Python, que nos permitió concentrarnos en la lógica de negocio pudiendo configurar y correr el servidor web de forma sencilla.

6.3.2 Heroku:

Dado que el desarrollo de *Frontend* y *Backend* fue dividido en dos equipos, resolvimos utilizar una herramienta llamada Heroku. Esto nos permitió dividir mejor y facilitar el trabajo, dándole la posibilidad al equipo de *Frontend* de acceder en todo momento a la última versión del *Backend* a través de la nube.

Heroku es lo que se denomina PaaS o *platform as a service*, el cual funciona como un servicio en la nube, permitiendo crear, ejecutar y mantener una aplicación sin la necesidad de concentrarnos en la infraestructura requerida por esta. Simplemente, se crea un proyecto en alguna de los lenguajes soportados (ej. Ruby, Java, Node.js entre otros) y se sube a un repositorio en la nube de Heroku, el cual comienza un proceso

(pipeline) que termina en la aplicación en funcionamiento con una IP pública para utilizarlo.

En nuestro caso, utilizamos las funcionalidades básicas para lograr que todo el equipo pueda utilizar una versión estándar del *Backend* para realizar pruebas de ser necesario, pero también nos permite en caso de utilizarla en producción, escalar en recursos (crear réplicas de la aplicación al tener más demanda) y monitorear el servicio brindado.

6.4 Traducción del código de barras

Una vez recibido el código de un producto desde la aplicación *Frontend*, el primer paso es obtener un nombre para el producto al que corresponde dicho código.

Para esto se realiza una consulta a los servidores de Google, enviando el código a buscar como parámetro. La respuesta es obtenida en formato html, donde se listan sitios que se corresponden con la búsqueda realizada. Este html es interpretado para obtener las palabras que aparecen con mayor frecuencia en los titulares de todos los sitios obtenidos. A partir de estas palabras se forma el nombre del producto.

Mencionamos también que Google ofrece una Api para desarrolladores, con múltiples servicios, entre ellos el Google Search Engine, que permite de forma gratuita realizar 100 consultas diarias y se puede comenzar a utilizar luego de configurar credenciales en el servicio de Google. Decidimos no continuar utilizando este método, ya que el formato de las respuestas a las consultas, eran links redirigiendo a otras páginas, no directamente como necesitamos.

Otra opción estudiada fue de utilizar servicios externos que permiten la traducción de códigos de barra, las cuales poseen api para la consulta. La desventaja es que solo se habilitan en forma de prueba para un tiempo limitado y la calidad de las respuestas para productos locales no era buena.

6.5 Búsquedas de artículos en tiendas

Para la implementación de la búsqueda de los artículos traducidos (el nombre del artículo desde el código de barra) en las tiendas elegidas se utilizaron dos técnicas, en primer lugar realizar scrapping, esto se aplicó a la búsqueda en las tiendas locales Ta-Ta y Tienda Inglesa. En segundo lugar, para la búsqueda en MercadoLibre y Amazon se optó por utilizar las APIs que estos proporcionan [5][9].

6.6 Mediante Web Scraping

Para realizar web scraping se utilizaron las siguientes 2 librerías: requests [14], que permite realizar consultas HTTP de forma rápida y sencilla, y BeautifulSoup [15], la cual permite extraer información tanto de archivos HTML como de archivos XML.

En primer lugar se debió formar la string de consulta, utilizando la url y el formato particular de cada sitio web para incluir el nombre del producto que se desea consultar. Luego de generada esta url se utilizo la libreria *requests* para descargar la página web correspondiente y finalmente se utilizó *BeautifulSoup* para extraer la información deseada.

BeautifulSoup permite seleccionar elementos HTML utilizando, entre otras cosas sus tags y nombres de sus clases, para luego extraer la información contenida en los mismos, además de sus atributos como “src” en el caso de las imágenes, “href” para los hipervínculos, etc.

En conjunto con estas librerías, se utilizó otra librería de python llamada difflib [16] que permite calcular la similitud entre 2 strings con el objetivo de seleccionar de entre la información obtenida mediante scraping aquellos resultados que más se asemejan al producto buscado.

6.7 Mediante API

En el caso de MercadoLibre se utilizó un servicio público que permite la búsqueda de productos, los resultados son idénticos a que si se realizara una búsqueda a través de su la página web. Este servicio devuelve los resultados de la búsqueda en formato json, el cual luego es parseado para obtener los datos que nos interesan (precio del producto y la URL correspondiente).

Amazon por su parte no expone su API de forma pública, por lo que fue necesario crearse una cuenta de “asociado”, esto permite realizar búsqueda de productos así como también otras funcionalidades como realizar publicaciones o publicitar publicaciones existentes. Una vez creado este tipo de cuenta, se tiene acceso a la API mediante un usuario y una clave. En el caso de Amazon la respuesta se encuentra en formato XML, el cual también debió ser parseado para obtener los datos relevantes, esto se realizó utilizando *AmazonSimpleProductAPI* la cual es una librería para python que simplifica la comunicación con la API [10].

Al realizar las búsquedas se pueden incluir varios filtros, en nuestro caso solamente consideramos los productos que se encuentren publicados en la modalidad “comprar ahora” y descartamos los que son subastados. También se tomó en cuenta que los productos sean nuevos. Por defecto las APIs devuelven un listado ordenado descendentemente por relevancia, siendo el primero el producto considerado más relevante. Al momento de devolver los productos para dicha tienda se consideran los primeros N productos del listado, siendo N parametrizable.

Para realizar las múltiples consultas de forma simultánea, tanto de scraping como aquellas que usan una API, se utilizo la libreria *multiprocessing* [17] para crear varios subprocesos y así paralelizar el procesamiento.

7. Evaluación del resultado

En cuanto al proyecto pudimos lograrlo exitosamente dado que se completo las funcionalidades especificadas en el marco de tiempo del proyecto de la materia.

Entre los puntos débiles que se encontraron, uno de ellos es a la traducción de código de barra a nombre de producto no es en el 100% de todos los casos correcta. Por que al tomar las palabras que se repiten con mayor frecuencia en los encabezados puede llevar a resultados inválidos. Cabe mencionar que existen aplicaciones que ofrecen APIs para realizar la traducción desde código de barra a producto utilizando bases de datos propias, pero estos servicios suelen ser pagos y no contienen mucha información para códigos de barra de productos comercializados en Uruguay.

Al no usar base de datos y almacenar las consultas, en casos de mucho tráfico de consultas se puede apreciar a simple vista una reducción de la eficiencia.

Los puntos positivos es que se hizo la interpretación de datos de distintas fuentes y generar una interfaz que permita utilizarlos por los usuarios finales de forma simple y amigable.

De momento todas las tecnologías que utilizamos son de uso gratuito por lo tanto la aplicación no tiene costos salvo el hospedaje en la nube que se debería pasar a un plan pago si se desea pasar a producción.

8. Conclusiones

Como objetivo principal queríamos comprobar que se podía realizar la traducción de un código de barras al costo del artículo específico, que fue la parte de mayor complejidad del problema. Como se menciona en el informe se realizó una traducción intermedia de código de barra a nombre de artículo para luego poder buscar el precio en las tiendas.

En cuanto a la búsqueda de productos en distintas tiendas, nos encontramos que es una funcionalidad relativamente sencilla de implementar si las tiendas disponen de una API con una documentación adecuada. También notamos que distintas APIs pueden utilizar distintos formatos de respuesta, lo cual implica un desarrollo más orientado a unificar las respuestas obtenidas.

En términos del curso de Recuperación de Datos y Recomendaciones de Web encontramos un montón de algoritmos y técnicas que resuelven la búsqueda de

información en la web. Además de todas las optimizaciones que se deben hacer para poder lograr un producto exitoso.

La libertad de poder elegir el tema a presentar y las tecnologías a utilizar es una experiencia que cómo estudiantes es muy útil porque en el mercado laboral en muchos casos se debe tomar este tipo de decisiones intentando prever que funcione para el problema a abordar y no se pierda tiempo innecesario en aspectos que no son de vital importancia para el problema.

9. Trabajo a Futuro

Como primer objetivo creemos que sería un buen comienzo agregar una base de datos de caché, la cual guarde todas las consultas realizadas que se lograron resultados exitosos, esta ayuda, a retornar una respuestas mas rápida al usuario y también permite devolver un resultado con mayor certeza de lo que se está buscando.

En un futuro es posible incluir más tiendas como fuente de búsqueda, teniendo en cuenta que para cada tienda en particular se debe implementar la forma en que el producto es buscado ya sea haciendo scrapping o consumido algún servicio y parseando la respuesta correspondiente.

También mejorar el algoritmo de búsqueda del nombre del producto a partir del código de barra, ya sea con ayuda de la base de datos que se habló al inicio de esta sección, incluyendo resultados conocidos de forma manual. O bien utilizar otra técnica que mejore los resultados.

10. Referencias

[1] Webir - Curso de recuperación de información y recomendaciones de la web https://eva.fing.edu.uy/course/view.php?id=821 - [consultado noviembre 2017].

[2] IONIC 2- Framework para desarrollar aplicaciones hibridas https://ionicframework.com [consultado noviembre 2017]

[3] Angular 2 - Framework de desarrollo web https://angular.io/ - [consultado noviembre 2017]

[4] Apache Cordova - Framework para creacion de aplicaciones hibridas https://cordova.apache.org/ - [consultado noviembre 2017]

[5] API MercadoLibre http://developers.mercadolibre.com/es/api-docs-es/ - [consultado noviembre 2017]

[6] Heroku https://www.heroku.com/developers - [consultado noviembre 2017]

[7] Django https://www.djangoproject.com/ - [consultado noviembre 2017]

[9] API Amazon https://docs.aws.amazon.com/AWSECommerceService/latest/DG [consultado noviembre 2017] - [consultado noviembre 2017]

[10] Librería python para Amazon https://pypi.python.org/pypi/python-amazon-simple-product-api - [consultado noviembre 2017]

[11] Web scraping https://en.wikipedia.org/wiki/Web_scraping - [consultado noviembre 2017]

[12] UPC código de barra http://ack123.com/?page_id=194 [consultado noviembre 2017]

[13] EAN-13 Código de barra http://www.barcodeisland.com/ean13.phtml [consultado noviembre 2017]

[14] Librería para python requests http://docs.python-requests.org/en/master/

[15] Librería para python BeautifulSoup https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[16] Librería para python difflib https://docs.python.org/2/library/difflib.htmll

[17] Librería para python multiprocessing https://docs.python.org/2/library/multiprocessing.html