Recuperación de Información y Recomendaciones en la Web

Curso 2019

Farmasearch

Facultad de Ingeniería, UDELAR

Autores:
Gastón Abella - 4.769.218-1
Victoria Cruces - 4.669.389-9
Noelia Lencina - 5.150.227-5
Victoria Madrid - 4.716.972-6

Docente:
Libertad Tansini

Versión 1.0

Noviembre, 2019

Índice

1. Introducción

Las compras web han crecido notoriamente en Uruguay en los últimos años. Cada vez hay más comercios que cuentan con una página web donde comprar sus productos. Además, es de conocimiento general que el precio de un producto puede variar significativamente según quién lo ofrezca, ya sea por la existencia de promociones particulares o por el simple hecho de que cada vendedor fija su precio de forma de competir con los demás en su rubro.

El caso de las farmacias es uno de los más notables. Es común ver grandes variaciones en los precios de medicamentos, por ejemplo. En algunos casos puede deberse a que se ofrecen marcas diferentes de productos similares o que un mismo producto de una misma marca se encuentre a otro precio por decisión del comercio. En ambos escenarios, el usuario quisiera tener toda la información posible que le ayude a decidir en qué lugar realizar su compra.

En este trabajo presentamos una posible herramienta con la que el usuario podrá comparar precios y tomar una decisión informada cuando necesite comprar algún producto de farmacia.

2. Problema

El problema que se busca resolver es la dificultad de elegir dónde conviene comprar algún artículo en particular. Se busca tener una forma de poder comparar los precios y artículos de forma fácil, teniendo toda la información en un único lugar.

En el contexto de este proyecto, particularmente se trabajó con farmacias, pero es un problema que se podría extender a cualquier otra área para la cual se cuente con páginas web de donde obtener datos para comparar.

En este caso, se utilizaron dos farmacias conocidas de Uruguay (Farmashop y San Roque), ya que tienen sus artículos disponibles en la web y además son dos comercios particularmente interesantes para comparar.

3. Enfoque de la solución

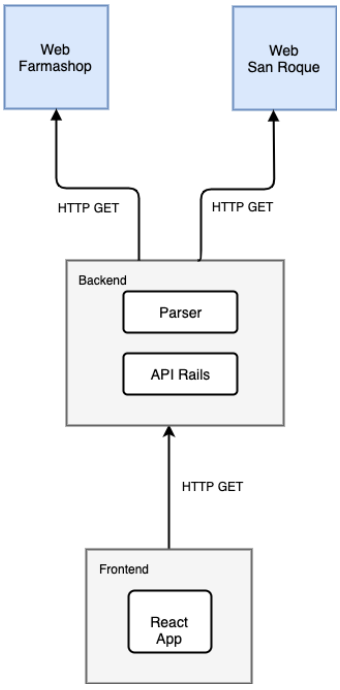
Mediante la solución planteada, el usuario puede buscar el artículo deseado, así como seleccionar las distintas fuentes (es decir, las distintas farmacias) en las que desea realizar la búsqueda. Para el alcance de este proyecto, como se mencionó anteriormente, decidimos utilizar específicamente a Farmashop y San Roque, por lo que nos pareció que no agregaría valor poder seleccionar menos de dos fuentes.

4. Diseño

En lo que respecta a la arquitectura del sistema, se dividió la misma en dos aplicaciones: una para el *backend* y otra para el *frontend*.

Para el *backend* se implementó una API en `Ruby on Rails`, encargada de proveer un endpoint que retorna, dado un string de búsqueda, los productos encontrados en ambas farmacias emparejados según similitud. Para hacer esto, la aplicación realiza pedidos a las url de las farmacias, parsea los datos mediante *Scrapping* y los agrupa de a pares. Se explicará más en detalle la implementación en la sección siguiente.

Para el *frontend* se construyó una aplicación en `ReactJS`, la cual utiliza el endpoint definido en la `Rails App` y muestra los resultados de forma amigable para el usuario y de manera que la comparación de los productos sea intuitiva.



5. Implementación

5.1. Backend

El sistema utiliza la gema `nokogiri` en el backend para obtener el HTML producto de hacer un pedido GET a las páginas de las farmacias, con los parámetros adecuados, y parsear la respuesta para obtener los datos necesarios para la comparación. Los datos que se extraen son:

- `item_name`: es el nombre del producto según lo presenta cada farmacia
- `image_url`: es la url a la imagen del producto
- `url`: es la url del producto (útil para cuando el usuario quiere realizar la compra)
- `price`: precio del producto en la farmacia
- `source`: farmacia fuente de los datos

Esta gema utiliza selectores en el HTML como lo son las clases aplicadas a los elementos para navegar los nodos y obtener sus valores. Es importante notar que cambios en las páginas web fuente, pueden requerir cambios en el código actual ya que al ser dos sistemas independientes, el sistema puede dejar de funcionar correctamente ante una nueva versión de la página web.

Los datos mencionados se exponen a través de una API json. Una respuesta ejemplo se ve de la siguiente manera:


```
{
    "similar_pairs": [
        {
            "similar_pair": [
                {
                    "item_name": "ENJUAGUE BUCAL LISTERINE ZERO MENTA SUAVE 250 ML",
                    "image_url": "https://tienda.farmashop.com.uy/media/catalog/...",
                    "url": "https://tienda.farmashop.com.uy/listerine-zero-...",
                    "price": "$ 226",
                    "source": "Farmashop"
                },
                {
```

```
                    "item_name": "LISTERINE ENJUAGUE BUCAL ZERO MENTA SUAVE 250ML",
                    "image_url": "http://www.sanroque.com.uy/media/catalog/...",
                    "url": "http://www.sanroque.com.uy/listerine-enjuague-bucal-...",
                    "price": "$U 265,00",
                    "source": "San Roque"
                }
            ],
            "similarity": 0.9978014647330867
        },
        {
            "similar_pair": [...],
            "similarity": ...
        },
        ...
    ],
    "farmashop": [
        {
            "item_name": "DESODORANTE BUCAL DEOXAN MENTA 9 ML.",
            "image_url": "https://tienda.farmashop.com.uy/media/catalog/...",
            "url": "https://tienda.farmashop.com.uy/deoxan-deo...",
            "price": "$ 71",
            "source": "Farmashop"
        },
        ...
    ],
    "san_roque": [...]
}
```


similar_pairs contiene pares de productos, uno por cada farmacia, que son considerados similares. Esta similitud se determina utilizando la distancia coseno con un umbral de 0.70. Con esto, dos productos cuya distancia coseno sea mayor o igual a 0.70 se consideran similares por lo cual se retornan agrupados. Además, aquellos productos que siempre estén por debajo de este umbral al compararlos con los demás, son devueltos en listas sin agrup ar, una para cada farmacia. Esto último es con el objetivo de presentarle información completa al usuario, aún cuando el sistema no pueda encontrar pares que sean parecidos.

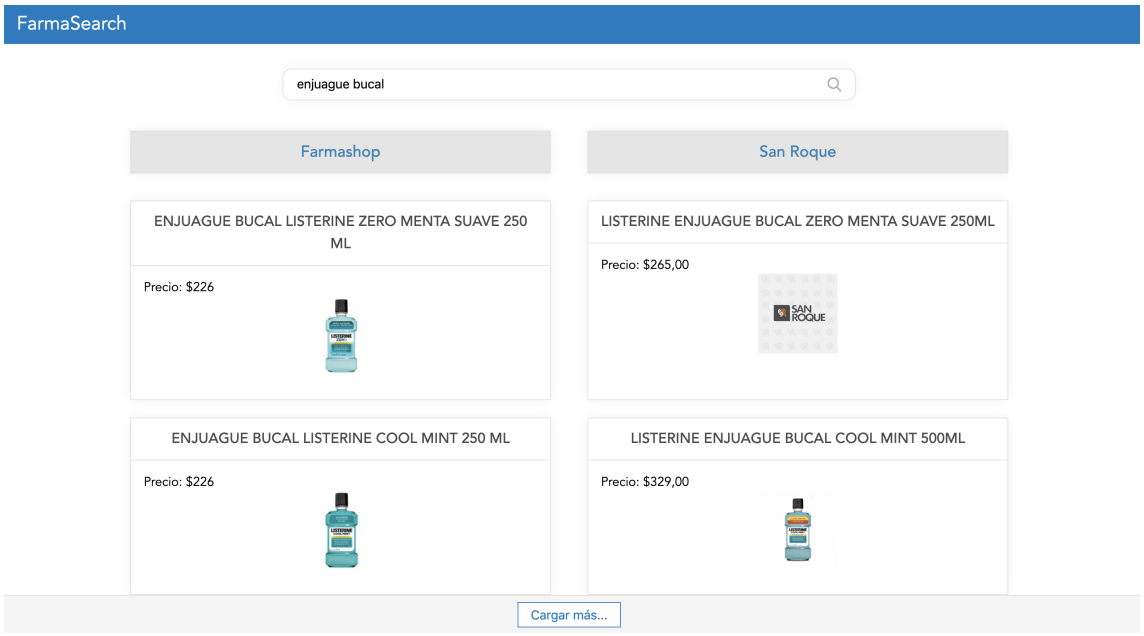
Finalmente, el backend implementa paginación de los resultados. Esto lo hace apoyándose en la paginación ya implementada en las páginas de las farmacias, con el detalle de que

para lograr esto, los pedidos a las páginas de las farmacias se hacen utilizando siempre el mismo tamaño de página para que de esta forma ambas fuentes retornen como máximo el mismo número de productos por pedido.

5.2. Frontend

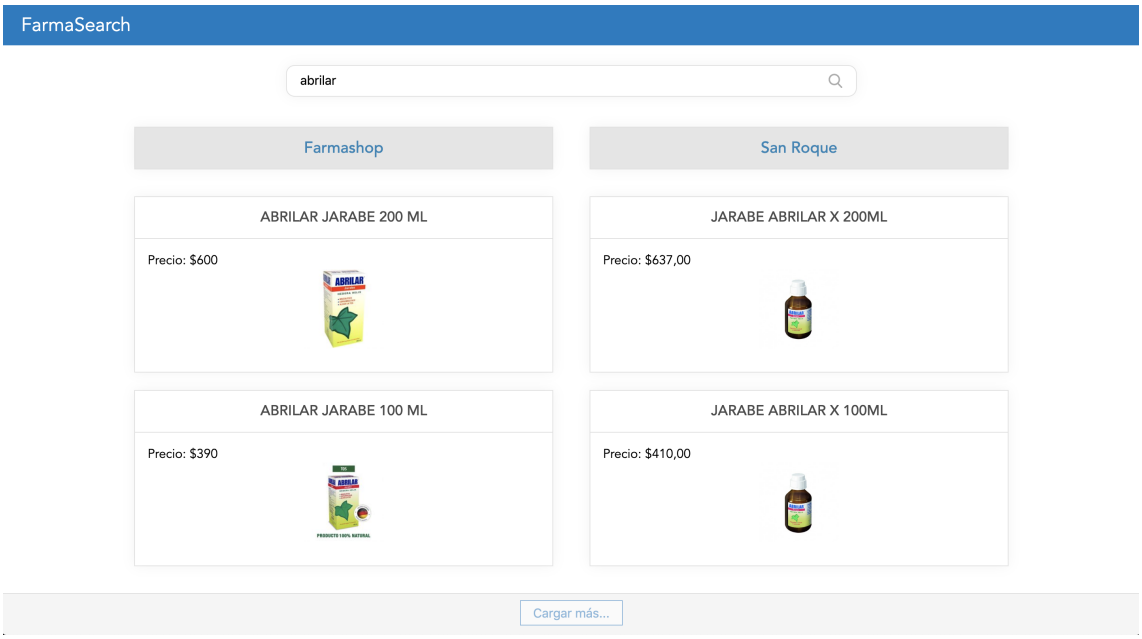
La aplicación de frontend (implementada en ReactJS) es independiente del backend. Se comunica a través de pedidos mediante el protocolo HTTP a la API que este segundo expone, y utiliza las respuestas que recibe en formato json para renderizar y mostrar los artículos de forma adecuada.

En la siguiente imagen se puede ver cómo se presentan los productos lado a lado de forma de poder compararlos fácilmente.



En la segunda imagen, se puede observar cómo se presentan los productos que el algoritmo no fue capaz de emparejar con otro.

En la última imagen queda a la vista la utilidad del sistema. Se muestran exactamente los mismos productos y fácilmente se puede ver en qué farmacia convendría más comprarlo.



6. Funcionalidades y uso

La funcionalidad principal es buscar productos a comparar. Para esto, se provee una barra de búsqueda para que el usuario pueda buscar lo que desee. Finalmente se listan los resultados, de forma tal que sean fácilmente comparables.

Los artículos se muestran con la foto original, siendo esta un link al artículo en su página web fuente, de forma tal que sea fácil para el usuario poder seleccionar qué artículo comprar y hacerlo en la web de la farmacia correspondiente.

Otra funcionalidad es poder seleccionar las fuentes a las que se irán a buscar los artículos, aunque como las únicas fuentes disponibles son Farmashop y San Roque, actualmente no le agrega tanto valor al usuario.

Por último, también se realiza el paginado de los artículos, pudiendo navegar por las distintas páginas de resultados.

7. Evaluación y resultados

Para evaluar el resultado obtenido, se hicieron varias pruebas con diferentes artículos, pensando casos donde el nombre del artículo sea variado (con espacios sin espacios, con mayúsculas o no, etc) y también artículos de distintos rubros, de forma que pueda haber distinta cantidad de resultados en las distintas respuestas.

Como se mencionó anteriormente, se utiliza la distancia coseno para comparar las descripciones de los artículos y emparejarlos según cuán similares sean.

Si se busca por ejemplo el medicamento “Abrilar”, se obtienen los siguientes pares de artículos:

Primer par:

- ABRILAR JARABE 200 ML (de Farmashop)
- JARABE ABRILAR X 200ML (de San Roque)

Segundo par:

- ABRILAR JARABE 100 ML (de Farmashop)

- JARABE ABRILAR X 100ML (de San Roque)

Como se puede ver, los resultados obtenidos al usar la distancia coseno son realmente buenos.

Si bien este modo de resolver el problema se ajusta bien y es aceptable para el contexto de nuestra solución, en algunos casos, se emparejan artículos que idealmente no serían comparables. Esto sucede sobre todo cuando se utilizan las unidades para describir el artículo.

Uno ejemplo de esto se ve al buscar "Enjuague Bucal". En este caso, se obtienen pares de artículos realmente similares como "ENJUAGUE BUCAL LISTERINE ZERO MENTA SUAVE 500 ML" (de Farmashop) y "LISTERINE ENJUAGUE BUCAL ZERO MENTA SUAVE 500M" (de San Roque), pero también se ven casos como "ENJUAGUE BUCAL LISTERINE CUIDADO TOTAL ZERO 500ML" y "LISTERINE ENJUAGUE BUCAL ZERO", que en realidad son artículos diferentes.

8. Conclusiones

En primer lugar, podemos afirmar que se alcanzaron los objetivos propuestos inicialmente en el proyecto, construyéndose una aplicación que permite la comparación de productos de distintas farmacias, así como una amigable visualización de los resultados.

Además, en general se obtuvieron resultados satisfactorios en la comparación de productos. Sin embargo, se observó que para ciertos productos el agrupamiento en pares similares no logró el mejor resultado, siendo en muchos casos las unidades de medida la principal causa de este problema. Intuitivamente, se observó que los mejores resultados fueron arrojados por búsquedas más específicas, mientras que las búsquedas por términos más generales llevaron a resultados no tan buenos.

En cuanto al desempeño en performance de la aplicación, se alcanzaron tiempos de respuesta aceptables considerando que *Scrapping* es una técnica costosa si se trabaja con volúmenes considerables de datos. Además, se identificaron mejoras para este punto a desarrollar en la sección siguiente.

Por último, la implementación realizada permite que sea fácilmente extensible a más farmacias y otros tipos de comercios, lo cual puede ser deseable bajo otros contextos de aplicación.

9. Trabajo futuro

Si bien las funcionalidades alcanzadas cumplen con los objetivos propuestos al principio del proyecto, consideramos que existen un sinfín de mejoras que se le pueden realizar a la aplicación. Se listan algunas a continuación:

- **Cache**. Con la implementación actual cada vez que un usuario hace una búsqueda de un producto particular, la aplicación realiza un request a cada farmacia sin guardar este resultado en ningún lugar. Si en cambio guardáramos las respuestas en memoria cache, tendríamos mejoras en los tiempos de respuestas para búsquedas idénticas en períodos cortos de tiempo. Otra solución sería guardar la información en una base de datos, lo que mejoraría aún más los tiempos pero implicaría tener que chequear por datos desactualizados.

- **Mejora en apareamiento de productos**. Los productos son comparados utilizando el nombre, con la función coseno como valor de similitud. Esto en la práctica trae algunos problemas, especialmente en productos que expresan sus unidades en el nombre y las representan de manera distinta. Una solución a este problema podría ser hacer un preprocesamiento de los nombres antes de compararlos, de manera de llevarlos a una forma canónica. Otra mejora podría ser considerar otros elementos del producto para la comparación. Por ejemplo se podría utilizar el precio y no permitir que los productos difieran más que cierto porcentaje. También se podrían comparar las imágenes de los productos utilizando alguna librería especializada.

- **Usar APIs de farmacias**. Si el día de mañana alguna de las páginas de las farmacias modifica sus url o ciertos tags HTML, la aplicación se rompe. Sería ideal que ésta fuera independiente de dichos atributos logrando así un mayor estado de estabilidad. Una solución sería (de ser posible) utilizar las APIs reales de cada farmacia, de manera de reducir lo más posible la cantidad de elementos causantes de error. Esto también implicaría evitarnos los costos de parseo, lo cual conlleva mejoras en performance.

- **Más farmacias**. El sistema actualmente soporta solamente comparación de productos entre *Farmashop* y *San Roque*, pero sería ideal poder incluir otras farmacias y así ampliar el rango de búsqueda de los usuarios.

- **Otros comercios**. Así como nos interesa incluir otras farmacias, también resultaría interesante agregar comercios distintos, tales como supermercados o tiendas especializadas en productos específicos (pinturerías, ferreterías, etc).

- **Otro Scrapper**. Podría experimentarse con el uso de otras gemas para Scrapping o con una tecnología distinta.

10. Bibliografía

Referencias

[1] Diapositivas del curso

[2] `https://nokogiri.org/`

[3] `https://reactjs.org/`

[4] `https://guides.rubyonrails.org/api_app.html`