DISEÑO DE LA APLICACIÓN LICOREANDO

DESARROLLADO POR EL GRUPO DE AII Nº 8

Romero Espárraga, David
Ortiz Calleja, Jesús
José Manuel López Carnicer

Contenido

1. Introducción

1.1. ¿Qué es *Licoreando*?

Licoreando es una aplicación web que sirve para la búsqueda y comparación de bebidas y licores de distintas páginas webs. En ella, podremos acceder a información de los productos, tales como el título, descripción, precio, graduación, capacidad, URL del producto en la página oficial, si está o no en Stock, etc.

1.2. Páginas webs utilizadas

Las páginas webs que hemos utilizado para obtener la base de datos de nuestros productos son:

Nombre	Url de la página	NºProductos extraídos	Tecnología utilizada
La Casa de los Licores	**http://lacasadeloslicores.es/tienda/**	**1418**	***Beautifulsoup-Webscraping***
Disevil vinos	**https://www.disevil.com/tienda/es/80-licores-y-destilados**	**733**	***Beautifulsoup-Webscraping***
Mariano Madrueño	**https://marianomadrueno.es/tienda/**	**897**	***Beautifulsoup-Webscraping***

1.3. Funcionalidades utilizadas en el proyecto

Funcionalidad	Tecnología utilizada
Registro en la aplicación	*Django-User*
Autenticación en la aplicación	*Django-Authentication*
Web-Scraping de productos	*BeautifulSoup*
Formulario de gustos de licores	*Django Forms*
Búsqueda de licores por título/descripción	*Whoosh-Django*
Búsqueda de licores por precio(Min-Max)	*Whoosh-Django*
Búsqueda de licores por graduación(Min-Max)	*Whoosh-Django*
Búsqueda de licores por categoría	*Whoosh-Django*
Ordenar licores por título	*Whoosh-Django*
Ordenar licores por precio	*Whoosh-Django*
Ordenar licores por graduación	*Whoosh-Django*
Ordenar licores por similitud	*Whoosh-Django*
Modelos de dominio	*Django models*
Índices y esquemas	*Whoosh*
Templates de la aplicación	*HTML-CSS*
Despliegue en Heroku	*Heroku-Django*
Sistemas de recomendación	*Django* *Python-Distance* *Python-Stop words* *Python-Unidecode*

2. Objetivos del proyecto

A continuación, se mostrarán los objetivos tanto didácticos como de la aplicación.

2.1. Objetivos didácticos

- Reforzar el aprendizaje de las herramientas vistas en clase.
- Uso avanzado de Django y Python.

2.2. Objetivos de la aplicación

- Registro de usuario.
- Búsqueda-filtrado de licores/bebidas.
- Comparar precios de distintas páginas de licores.
- Ordenar licores por distintos órdenes.
- Obtener recomendaciones basadas en contenido.

3. Herramientas y tecnologías a utilizar

3.1. BeautifulSoup

La herramienta BeautifulSoup nos ha servido de gran ayuda, puesto que hemos podido acceder a la información de las distintas páginas webs y extraer todos los productos de forma automática.

Para llevar a cabo el *webscraping*, hemos añadido 3 módulos de Python, uno para cada página: *scraping_disevil.py*, *scraping_casalicores.py* y *scraping_marianomadrueno.py.*

Cada uno de ellos se ha implementado de forma personalizada, ya que cada página muestra la información de forma distinta. Antes de realizar el webscraping, pusimos en común una serie de criterios que nos ayudarían más adelante a que los atributos tuvieran una estructura similar, como por ejemplo los atributos que aparecían como nulos, el formato para el precio, la graduación…

BeautifulSoup sólo se aplicará para poblar la base de datos por primera vez, y usaremos ésta misma para el resto de funcionalidades. El motivo es que el tiempo de procesamiento para los 3000 productos es del orden de varias horas.

3.2. Bases de datos

Mientras realizamos el webscraping, almacenamos en una base de datos de Sqlite toda la información extraída. Este proceso se realiza en el fichero *saveInDB.py*, que se encarga de hacer las llamadas a los scrapings y guardar en la base de datos la información.

La información que hemos almacenado en la base de datos es la siguiente:

- **Licor:** Título, descripción, precio, graduación, origen, cantidad/peso, url del producto, url de la imagen del producto, y si está o no en stock.
- **Categoría:** Contiene el nombre de la misma, y está relacionada con el licor.
- **Usuario (auth_user):** Hereda de la librería de Django-User, y contiene atributos como el nombre, apellidos, nombre de usuario, contraseña…
- **Puntuaciones de usuarios:** Contiene las puntuaciones que han añadido los usuarios. Se puede puntuar el tipo de licor, categorías y marcas favoritas.
- **Recomendaciones:** Se trata de una caché que almacena las ids de los licores recomendados.
- **Tablas complementarias:** Además de las tablas anteriores, en nuestra base de datos tenemos varias tablas que se crean al usar las librerías de Django, como las de permisos de usuario, migraciones de Django, grupos…

3.3. Whoosh

La herramienta *Whoosh* nos ha sido de gran utilidad para el proceso de filtrado y ordenación de productos, ya que con el uso de los índices y esquemas, podemos realizar funcionalidades de forma muy eficiente.

Hemos realizado un método genérico para todas las funcionalidades que se muestran a continuación, que recibe todos los parámetros de forma opcional, y según los parámetros que reciba realiza una funcionalidad u otra.

Las funcionalidades que hemos implementado usando Whoosh son las siguientes:

3.3.1. Filtrados de productos

Todos los filtrados que se muestran a continuación, priorizan aquellos productos que se encuentran en stock.

- **Filtrados de productos por título/descripción**: Este filtro de búsqueda permite añadir cadenas de texto, y, aunque las coincidencias no sean exactas, muestra las más parecidas, esto siempre priorizando los productos que más coincidan con la propia búsqueda (uso de FuzzyTerms). Para ello hemos utilizado el criterio de ordenación de Whoosh *ScoreFacet*, dándole distintos pesos a los atributos que hemos visto conveniente.

- **Filtrados de productos por categoría:** La búsqueda de productos por categoría mostrará aquellos productos cuya categoría coincida con alguna de las seleccionadas.

Los siguientes filtros los hemos realizado mediante la metodología de agrupaciones de Whoosh, usando el atributo *groupedby:*

- **Filtrados de productos por rango de precios:** El sistema mostrará los productos cuyos precios se encuentran en el rango indicado. Esta funcionalidad utiliza el mecanismo de facetas, en concreto el *RangeFacet.*
- **Filtrados de productos por graduación:** El sistema mostrará los productos cuyas graduaciones se encuentran en el rango indicado. Esta funcionalidad utiliza el mecanismo de facetas, en concreto el *RangeFacet.*

Si el usuario decide añadir filtros de búsqueda por precio y por graduación a la vez, las dos facetas se incluyen en una multifaceta (*Multifacet)*, y satisfacen la petición realizada.

3.3.2. Ordenación de productos

Todos los productos que se muestran al ordenar por las siguientes funcionalidades, priorizan aquellos productos que se encuentran en stock.

- **Ordenación de productos por similitud:** Este criterio de búsqueda ordena los productos según el grado de similitud con la búsqueda realizada. Si no se introducen datos en la búsqueda, los productos aparecerán en el mismo orden en el que se encuentran en la base de datos.
- **Ordenación de productos por precio:** Esta funcionalidad ordena los productos según el precio que tengan.
- **Ordenación de productos por título:** Esta funcionalidad ordenará alfabéticamente por título los productos que se muestran.
- **Ordenación de productos por graduación:** Este criterio de ordenación mostrará primero los productos con menor graduación.

3.4. Django

El framework de desarrollo web *Django* ha sido una de las herramientas más importantes que hemos utilizado, ya que ha sido la base de la aplicación. Con él hemos podido implementar:

- Modelos de dominio de la base de datos.
- Formularios de registro.
- Formulario de gustos para el sistema de recomendación.
- Estructura de la aplicación web.

3.5. Sistemas de recomendación

El sistema de recomendación que hemos utilizado es el basado en contenido.

Para poder elegir los productos recomendados al usuario, éste ha debido rellenar un formulario con sus preferencias. Una vez completado el formulario, el sistema comparará esos datos con los contenidos de los licores de la base de datos.
Tanto para el criterio de recomendación por precio como por graduación, el sistema calcula la distancia a la que se encuentra el atributo del licor del punto medio del rango de preferencia del usuario indicado en el formulario. Mientras esté dentro del rango, será un ratio positivo, pero, si se encuentra fuera del rango, será un ratio negativo.

El ratio generado para el comentario del usuario en el formulario se basa en compararlo con la descripción de los licores. Para ello, primero se filtran ambos textos, eliminando las palabras carentes de significado. Seguidamente, se pasa todo a minúsculas, se eliminan los acentos, se cuenta el número de veces que se repiten las palabras del comentario en la descripción de cada uno de los licores, teniendo en cuenta la distancia Levenshtein=2.

Aquellos atributos a los que el usuario ha puesto puntuación, primero se les hace pasar por el mismo procesamiento de texto por el que pasa el comentario, y si el ratio devuelto es superior a 1, se devuelve por ratio la puntuación del usuario dividida entre 10. En caso contrario, es 0.

El resultado final consiste en la suma de todos los ratios calculados, dándonos la puntuación de un licor. Los 20 licores con mejor puntuación se mostrarán al usuario como licores recomendados.

3.6. Despliegue en la nube: Heroku

El despliegue de nuestra aplicación en Heroku nos permitirá acceder a la misma por internet. Para llevar a cabo este procedimiento, hemos tenido que realizar varias modificaciones en nuestro proyecto. Hemos añadido un fichero *local_settings_heroku.py*, que nos permite realizar las configuraciones necesarias para el despliegue. Además, hemos añadido en el archivo *settings.py* un apartado para que utilice los settings de Heroku. Por último, hemos creado un fichero *Procfile* que servirá para que Heroku lance los comandos necesarios para ejecutar el proyecto, tales como *python manage.py migrate*, python manage.py makemigrations….etc. Además, se ha añadido un comando para que cargue en su base de datos la nuestra propia, que lo leerá a partir de un fichero que en formato json contiene la información de toda nuestra base de datos.

La URL de nuestra aplicación desplegada en Heroku:

https://licoreando.herokuapp.com/

4. Problemas encontrados

Problema	Descripción	Solución
Webscraping de las páginas	Ha sido muy costoso extraer toda la información de los productos, ya que muchos de ellos no seguían un mismo patrón, incluso perteneciendo a la misma página.	Realizar un análisis exhaustivo de los distintos HTML, hacer procesamientos de texto complejos, y capturar las excepciones que se lanzaban con ciertos parámetros.
Tratamiento de datos	En algunos casos hemos detectado que al extraer los datos pertenecientes a las distintas páginas, había datos incorrectos y no tenían sentido.	Unificación de criterios para todos los datos. Sustituir parámetros incorrectos por unos reales donde ha sido posible.
Ordenación de productos por precio	El sistema de ordenación que utilizábamos con Whoosh no ordenaba de forma correcta los productos que sólo variaban en la parte decimal.	Bifurcación del parámetro precio en dos, de tipo numeric, con la diferencia que uno lleva dos decimales obligatorios.

5. Posibles mejoras

Mejora 1: Hacer un sistema de recomendación basado en conocimiento, en el que los usuarios puedan puntuar a los licores de la aplicación.

Mejora 2: Mejoras en la interfaz de la aplicación.

Mejora 3: Aumento del número de páginas webs de las que se extraen los datos.

Mejora 4: Convertir la aplicación en una red social.

6. Conclusiones

Tras la realización de este proyecto, hemos reforzado nuestro aprendizaje de las herramientas que hemos visto en la asignatura, y además, hemos conseguido aprender muchos conocimientos nuevos.

Hemos conseguido hacer una aplicación que cubre las tecnologías vistas en clase, y también, hemos profundizado en ellas.

Por último, hemos aprendido a desplegar la aplicación en la nube, en Heroku precisamente.

7. Bibliografía

[1] Documentación de BeautifulSoup https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[2] Documentación de Whoosh https://whoosh.readthedocs.io/en/latest/

[3] Tutorial de Django https://docs.djangoproject.com/es/2.1/intro/

[4] Sistemas de Recomendación basados en contenido- Valiente verde
http://www.p.valienteverde.com/sistemas-de-recomendacion-basados-en-el-contenido-content-based/