Motor de Búsqueda Simple Foro U-campus

Yerko Sepúlveda Rojas

December 27, 2020

1 Objetivo

El objetivo principal del proyecto es generar un motor de búsqueda para el foro de U-campus el cual da acceso a los mensajes alojados en el foro, siendo como principales datos el autor con el mensaje principal y los comentarios que obtuvo dicho mensaje con sus autores respectivos, con ello se podría analizar los tópicos que generan mas interés e interacción entre la comunidad universitaria, quienes son los usuarios mas activos y cuales son sus aportes, etc.

2 Datos

La idea de este proyecto nace de la necesidad para encontrar mensajes en el foro de u-campus (se uso el foro general para el proyecto debido a su mayor volumen de datos en comparación al foro de asignaturas), dado que es una funcionalidad que puede llegar a resultar útil y no se encuentra disponible, para el proyecto se usaron datos del mismo foro de U-campus los cuales se extrajeron mediante Web scraping, los datos se componen de:

- autor: representa el autor del mensaje principal o de los comentarios
- mensaje: representa el mensaje principal o de los comentarios
- respuestas: representa el numero de respuesta que tiene el mensaje principal

El formato en el que se encontraba era string de html y se proceso a json para su posterior uso, todos los datos tienen un peso total de 4.45 MB con un total de 2601 mensajes principales y 9280 mensajes leídos (esto incluye mensajes principales y comentarios), sin embargo, estos aumentan con el tiempo.

3 Métodos

Los métodos utilizados para concretar el proyecto fueron principalmente el web scraping y selección de los datos a usar, normalización de los datos, estructurar los datos y almacenar los datos.

- Web Scraping:
  Para extraer los datos del foro se utilizo Selenium y beautifulsoup4 (bs4), ambos como librerías de Python, Selenium controla las acciones de la pagina (login, extraer cuerpo de la web y navegación), mientras que bs4 se utiliza para extracción de los datos entregados por Selenium. En principio se utilizaría solo Selenium o bs4, sin embargo, el uso del primero era obligatorio dado que, bs4 no tiene funciones para login o navegación web, por otro lado bs4 se tuvo q usar debido a que los tags de la web no son generales en lo que mensajes corresponde, lo que generaba errores y complejidad a la hora de intentar usar solo Selenium para la extracción.

- Selección de Datos:
  En un inicio se planeaba almacenar el nombre de quien creo la entrada principal, la fecha, cantidad de respuestas y el mensaje, anidando los comentarios con el mismo formato, sin embargo se encontró el problema de que los datos no estaban en un formato general, los nombres tenían variantes, por lo que podría alguien tener su nombre completo o una variante tanto en el contenido y orden del mismo, por lo tanto se trabajaron los nombres como sub listas del mismo, los demás datos como la cantidad de respuestas y el mensaje era simple de obtener, sin embargo la fecha necesitaba un proceso especial, dado que tenia variantes: segundos, horas, días, por meses y fecha con formato, lo que aumentaba la complejidad del problema, debido a esto se simplifico la recolección de datos y se elimino la fecha de los datos.

- Normalización de Datos: los datos obtenidos de la parte anterior debían separarse y limpiarse, dado que contenían residuos de código html, por lo que, se crearon funciones para esto. ()

- Estructurar Datos: Los datos se almacenaron en diccionarios con formato json, debido a su flexibilidad a la hora de almacenar datos y que es totalmente compatible con Mongodb.

- Almacenar Datos: Inicialmente se almacenaron en csv dado que aun no se tenia claro el donde se almacenarían los datos, el formato generaba problemas con la anidación de los comentarios por lo que se decidió cambiar a el formato json, este formato es mas flexible con el almacenamiento de datos y totalmente compatible con Mongodb.

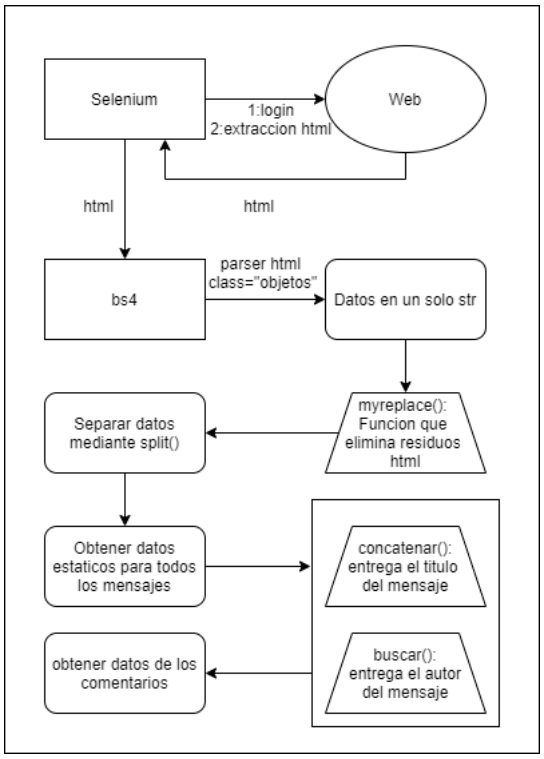

Figure 1: Modelo simple de extracción y normalización de los datos

4 Resultados

Como resultado se obtuvieron los datos del foro de u-campus con un formato simple de trabajar:

```
{
    "id": int, #representa los mensajes principales
    "id_mens": int, #representa mensajes principales y comentarios
    "autor": [], #autor del comentario en una lista
    "titulo": "string", #titulo del hilo
    "respuestas": int , #numero de respuestas del hilo
    "mensaje": string,#contenido del mensaje
    "comentarios": [{"id_mens": int,
                     "autor": [],
                     "mensaje": string}]
 }
```

Figure 2: Formato de json

El proceso para obtener los datos tomo el siguiente tiempo (todos los tiempos son en segundos):
extracción de datos y normalización El tiempo de ejecución fue: 158.61955952644348
estructuración de datos El tiempo de ejecución fue: 0.014013528823852539
almacenamiento de datos El tiempo de ejecución fue: 0.2574739456176758
Total de hilos leídos : 2601
Total de mensajes leídos : 9285
Peso total : 4.45 MB
Con todo esto almacenado en mongodb podemos buscar por ejemplo cuales son los 3 hilos con mas respuestas.

Figure 3: Top 3 hilos con mas mensajes

5 Conclusión

Durante la realización de este proyecto lo mas complejo fue la extracción y limpieza de los datos, a pesar de estar familiarizado con los métodos utilizados, finalmente las mejoras pendientes serian la agregación de la fecha en los mensajes, algo importante cuando se van a analizar datos de este tipo, debido a que podríamos analizar el impacto de un mensaje en cierto periodo de tiempo en la comunidad, agregar una interfaz de búsqueda que no sea directamente desde Compass o del shell de Mongodb, entre otras cosas, algo que se podría mejorar en eficiencia serian los métodos de limpieza o extracción del código y tal vez probar con distintas estructuras del json para almacenar los datos.