Recuperación de la Información y Web Semántica - MUEI

Integrantes del Grupo

- **Jesús Senra Paz** - jesus.senra@udc.es
- **Adrián García Vázquez** - adrian.gvazquez@udc.es

- Enlace repositorio

Memoria del Proyecto

1. Introducción

El objetivo de este proyecto fue crear un sistema que permita obtener ofertas de juegos de la plataforma Steam mediante scraping web, almacenar dicha información en un índice de Elasticsearch y exponerla a través de un frontend en React. Esto permite explorar, buscar, y visualizar los datos, además de implementar funcionalidades adicionales como recomendaciones y agrupación de resultados.

2. Descripción del Scraper

2.1. Configuración y Estructura

El script utiliza **Scrapy** junto con **Selenium** para realizar scraping dinámico de la página de ofertas de Steam. Esto permite cargar contenido que se genera de forma asíncrona en la página, como las ofertas que aparecen al hacer scroll.

- **Configuración del navegador:**
  - Se configura Selenium para operar en modo headless (sin interfaz gráfica).
  - Se utiliza un `ChromeDriver` para interactuar con el navegador.
- **Carga de contenido:**
  - El método `scroll_to_load` simula el desplazamiento en la página para cargar todas las ofertas disponibles dinámicamente, deteniéndose cuando ya no hay más elementos que cargar o se alcanza un límite.
  - Al principio lo habíamos configurado para que se obtuviera un máximo de 200 elementos, pero más adelante eliminamos ese límite, agregando otro: límite de scrolls.
  - Esto fue necesario pues los datos apareceían en la página mediante un scroll infinito y para efectos de la práctica se ha considerado un límite.
- **Extracción de datos:**
  - Una vez cargada la página, su contenido se procesa con **Scrapy Selector** para obtener información de cada oferta.

2.2. Atributos extraídos

De cada juego se extraen los siguientes datos clave:

- **Título:** Nombre del juego.
- **URL:** Enlace directo al juego en la tienda de Steam.
- **ID de la aplicación:** Identificador único del juego en Steam.
- **Fecha de lanzamiento:** Día, mes y año del lanzamiento del juego.
- **Resumen de reseñas:**
  - Texto descriptivo del resumen.
  - Porcentaje de reseñas positivas.
  - Cantidad de reseñas totales.
- **Precio original:** Precio sin descuento.
- **Descuento aplicado:** Porcentaje de descuento.
- **Precio final:** Precio después del descuento (incluye manejo de juegos gratuitos).
- **URL de la imagen:** Enlace a la imagen promocional del juego.

2.3. Transformaciones realizadas

1. **Normalización de precios:**
   - Se convierten precios a un formato numérico adecuado.
   - Se manejan juegos gratuitos estableciendo su precio final en `0.0`.
2. **Descuento:**
   - El porcentaje de descuento se limpia y convierte a formato numérico.
3. **Reseñas:**
   - Se analiza el tooltip de la página para extraer los porcentajes y conteos de reseñas.
4. **URLs de imágenes:**
   - Se reemplaza la imagen de cápsula pequeña por la imagen de encabezado del juego.
   - Se ha encontrado que remplazando el nombre de la imagen por `header` se obtiene una imagen de mayor calidad.
5. **Fecha de lanzamiento:**
   - Se separan las partes de la fecha en día/mes y año.

2.4. Estructura del Código

El bloque principal del código se organiza en métodos que cumplen roles específicos:

- `start_requests`**:**
  - Configura el driver de Selenium y realiza la navegación inicial.
  - Genera una solicitud Scrapy con la fuente de la página cargada dinámicamente.
- `setup_driver`**:**
  - Configura y retorna una instancia de Selenium WebDriver en modo headless.

- `scroll_to_load`:
    - Simula el desplazamiento infinito de la página y retorna el contenido completo cargado.
- `parse`:
    - Usa Scrapy Selector para procesar el contenido cargado y extraer las ofertas.
- `extract_game_data`:
    - Limpia y normaliza los datos extraídos de cada elemento.
- `save_page_source`:
    - Guarda la fuente de la página como HTML para depuración.
- `parse_review_summary`:
    - Extrae los porcentajes de reseñas positivas y el conteo de reseñas desde el tooltip.
- `replace_img_name`:
    - Cambia la URL de la imagen a una versión de encabezado más relevante.

---

3. Elasticsearch

El archivo `pipeline.py` contiene la clase `ElasticsearchPipeline`, que define la lógica para enviar los datos recopilados por Scrapy a un índice en Elasticsearch. Este pipeline se encarga de gestionar la conexión, crear el índice si no existe y almacenar los documentos correspondientes a los juegos scrapeados.

3.1. Componentes principales

Este método se ejecuta al inicio del scraping y tiene como propósito inicializar la conexión con Elasticsearch. Además, verifica si el índice llamado `steam_deals` ya existe en Elasticsearch. Si no existe, lo crea utilizando el método `create_index`.

Define el esquema (mapping) para el índice `steam_deals` en Elasticsearch. Los campos incluidos son:

- `app_id` (keyword): Identificador único del juego.
- `title` (text): Título del juego.
- `url` (keyword): URL de la página del juego.
- `release_day_month` (text): Día y mes del lanzamiento.
- `release_year` (keyword): Año de lanzamiento.
- `review_summary` (keyword): Resumen de las reseñas.
- `positive_review_pct` (integer): Porcentaje de reseñas positivas.
- `review_count` (integer): Número total de reseñas.
- `original_price` (text): Precio original del juego.
- `discount_pct` (integer): Porcentaje de descuento.
- `final_price` (double): Precio final del juego con descuento.
- `img_url` (text): URL de la imagen asociada al juego.
- `timestamp` (date): Fecha y hora en que se registró el documento.

Este mapping nos sirve más adelante en la parte del front, en concreto a la hora de los filtros `Facet`que nos ayuda a un filtrado más dinámico por parte del usuario.

3.2. Creación del índice en Elasticsearch

- **Índices utilizados:**
    - Cada juego es almacenado como un documento en un índice.
    - Los campos almacenados incluyen todos los atributos procesados por el scraper.
- **Configuración del servidor:**
    - Se habilitó el acceso CORS para permitir consultas desde el frontend.
    - La seguridad del servidor(AUTH) fue desactivada para simplificar el desarrollo.

3.3. Flujo del pipeline

Inicio:

- Se conecta a Elasticsearch.
- Verifica si el índice steam_deals existe.
- Si no existe, lo crea con el esquema definido.

Procesamiento de cada item:

- Extrae los datos del juego.
- Indexa un documento en el índice steam_deals con los datos extraídos.

Finalización:

- Cuando se completa el scraping, el pipeline cierra cualquier recurso abierto (aunque en este caso no se realiza ninguna acción en close_spider).

---

4. Frontend en React

En el frontend, se utilizó la librería **Search UI** de Elastic para implementar una interfaz de búsqueda conectada al índice `steam_deals`. Esta herramienta facilita la integración con Elasticsearch, permitiendo configurar los campos de búsqueda, los resultados que se muestran y los filtros interactivos.

4.1. Casos de uso

1. **Visualización de ofertas:**
    - Los datos se muestran de manera estructurada en tarjetas que incluyen imágenes, precios y descuentos.
2. **Búsqueda simple:**
    - Se permite filtrar juegos por el título.
3. **Búsqueda avanzada:**

- Se permite filtrar juegos por precio, año de lanzamiento, porcentaje de descuento y popularidad.

4. **Clustering de resultados:**
    - Los juegos son agrupados por rangos de precio y porcentaje de descuento. Esto se coonsigue mediante los Facets del frontend.

4.2. Conexión con Elasticsearch

La conexión al servidor de Elasticsearch se establece a través de un conector proporcionado por **Search UI**. Este conector apunta al índice `steam_deals`, asegurando que las consultas se realicen directamente contra los datos almacenados.

4.3. Configuración de la Búsqueda

Se definió un objeto de configuración para personalizar el comportamiento de la interfaz de búsqueda:

- **Campos de búsqueda**: Se configuró el campo `title` como el principal para realizar consultas.
- **Campos de resultados**: Se seleccionaron los campos más relevantes del índice (`title`, `release_year`, `discount_pct`, entre otros) para ser mostrados en los resultados.
- **Facets**: Se añadieron filtros interactivos para que el usuario pueda refinar los resultados por:
    - Año de lanzamiento (`release_year`).
    - Porcentaje de descuento (`discount_pct`).
    - Porcentaje de reviews positivas (`positive_review_pct`).
    - Precio final (`final_price`), configurado como un filtro por rangos.

4.5. Facets y Rangos

El filtro por precios incluye varios rangos definidos manualmente, como `<10€`, `10€ - 20€` o `>100€`.

---

5. Conclusión

Este proyecto muestra lo fácil que es crear un buscador eficiente usando herramientas que se comunican de manera sencilla entre ellas. Al integrar Scrapy para el scraping, Elasticsearch para almacenar y consultar los datos, y React para el frontend, se logró construir una aplicación funcional que permite explorar las ofertas de Steam de forma rápida y dinámica. La configuración de los filtros y la visualización de resultados fueron tareas simples gracias a las herramientas que se usaron.

Además, las herramientas utilizadas en el proyecto ofrecen muchas más posibilidades que las que se han implementado. Elasticsearch, por ejemplo, permite realizar filtros mucho más específicos y complejos, lo que puede mejorar la precisión de las búsquedas. Asimismo, la integración de facets y rangos puede extenderse para incluir más tipos de filtros y agrupaciones, lo que optimiza la experiencia del usuario. Al aprovechar todas las capacidades de estas herramientas, se pueden obtener búsquedas aún más rápidas y eficientes, con resultados mucho más relevantes y personalizados para cada usuario.