Lab 8 – Elasticsearch en Wikipedia

CC5212-1 – May 7, 2025

Hoy construiremos un motor de búsqueda para Wikipedia.

Primero indexaremos la URL y las palabras clave en el título y el resumen (primer párrafo) de todos los artículos de Wikipedia en español utilizando Elasticsearch. Luego, implementaremos la función de búsqueda por palabras clave.

- Consulta los datos de ejemplo aquí: `http://aidanhogan.com/teaching/data/wiki/es/es-wiki-articles-1k.tsv`. Este archivo contiene las primeras 1,000 líneas de datos, con un artículo codificado en cada línea. Cada línea incluye la URL, el título y el resumen de un artículo. Los datos completos están en el servidor, con 964,838 líneas/artículos en el mismo formato.
- A continuación, presentamos Elasticsearch: un índice invertido distribuido para documentos en formato JSON. Se puede interactuar con Elasticsearch a través de una API RESTful basada en HTTP (que utiliza métodos como `GET`, `POST`, `PUT`, `DELETE`, etc.), enviando y recibiendo solicitudes en formato JSON. El comando `curl` se usa aquí como una herramienta general para enviar solicitudes HTTP y recibir respuestas. Inicia sesión en el clúster como de costumbre. La API REST de Elasticsearch está alojada en `cm:9200`.
  - Para ver el estado del clúster: `curl -X GET "cm:9200/_cat/health?v&pretty"`
  - Para agregar un documento al índice "`customer`" (el índice se creará automáticamente si no existe; todo en un solo comando):

    ```
    curl -X PUT "cm:9200/customer/_doc/1?pretty"
      -H 'Content-Type: application/json' -d '{ "name": "Fin Shepard" }'
    ```

  - Para obtener un documento indexado por su id: `curl -X GET "cm:9200/customer/_doc/1?pretty"`
  - Para mostrar los índices existentes: `curl "cm:9200/_cat/indices?v"`
  - Para eliminar un índice: `curl -X DELETE "cm:9200/customer?pretty"`
  - Para crear un índice llamado `mywiki` con un solo campo `TITLE` de tipo `text`, con `store` como true, y usando el analizador `standard` (lo explicaremos en un momento; observa el apóstrofe de cierre (`'`)):

    ```
    curl -X PUT "cm:9200/mywiki?pretty" -H 'Content-Type: application/json' -d'{
      "mappings" : {
        "_doc" : {
          "properties" : {
            "TITLE" : { "type" : "text", "store" : "true", "analyzer" : "standard" }
          }
        }
      }
    }'
    ```

- Entonces, ¿qué significa ese último comando? En realidad, hay algunos conceptos importantes que aclarar:
  - En Elasticsearch, indexaremos tipos similares de *documentos* en un índice.
  - Un *field* es un atributo de un documento, como title, URL, last-updated, abstract, etc. Más tarde, podemos seleccionar qué fields queremos utilizar para buscar documentos, por lo que les damos nombres a los fields, como `TITLE`, para poder referenciarlos más adelante. El contenido de los fields puede ser de un `"type"`; Por ejemplo, un field `TITLE` podría contener texto, un field `LAST-UPDATED` podría almacenar fechas, etc.
  - A continuación, recordemos que los índices invertidos típicamente analizan un texto (como un título o abstract) dividiéndolo en múltiples palabras. Además, las palabras pueden ser normalizadas (por ejemplo, `Chilean` → `chile`), perdiendo información del texto original. Por lo tanto, solo con el índice invertido, a menudo es imposible recuperar el texto original. Si queremos poder recuperar el contenido original más

tarde, por ejemplo, para mostrar el título o el abstract en los resultados de búsqueda, debemos establecer `"store"` en `"true"`, lo que indica que se debe almacenar el contenido exacto original (con el costo de un índice más grande). Aunque todavía podemos realizar búsquedas en fields con `"store"` establecido en `"false"`, no podremos recuperar el contenido original de esos campos.

  - Finalmente, el `"analyzer"` es el componente que extrae palabras del texto, las normaliza (incluyendo capitalización, acentos, stemming, etc.) y filtrar stopwords. El analizador `"standard"` funciona de manera genérica para todos los idiomas, pero podemos elegir analizadores más específicos, como los diseñados para idiomas específicos, como `"spanish"`. Como ejemplo pequeño, un analizador estándar no considerará una palabra como `"ese"` como un stopword, mientras que el analizador en español sí lo hará. Es importante utilizar el mismo analizador tanto para la indexación como para las consultas; de lo contrario, si se utilizan diferentes métodos de análisis y/o normalización, pueden surgir "desajustes" debido a palabras normalizadas de maneras distintas. Por esta razón, Elasticsearch configura el analizador a nivel de un índice en particular.

- Aunque podríamos usar la API REST para indexar los documentos uno por uno, tiene más sentido construir el índice programáticamente. Para ello, utilizaremos un cliente Java para Elasticsearch (alternativamente, podríamos desarrollar un cliente personalizado para la API REST si lo prefiriéramos). Descarga el proyecto de código desde u-cursos y ábrelo en Eclipse o tu IDE de Java preferido. La primera clase de Java con la que trabajaremos, llamada `BuildWikiIndexBulk.java`, indexará el archivo de datos grande en el índice invertido de Elasticsearch. La segunda clase con la que trabajaremos, llamada `SearchWikiIndex.java`, tomará tus búsquedas por palabras clave y utilizará el índice para encontrar artículos relevantes de Wikipedia.
- En ambas clases, la cadena `TODO` te indicará los lugares donde necesitas completar el código (en total hay tres partes que completar). También ten en cuenta que utilizamos una enumeración para asegurarnos de que mantenemos consistentes los nombres de los fields al consultar e indexar; específicamente, utilizamos:

  - `FieldNames.URL.name()` for the url (which is the string `"URL"`)
  - `FieldNames.TITLE.name()` for the title (which is the string `"TITLE"`)
  - `FieldNames.ABSTRACT.name()` for the abstract (which is the string `"ABSTRACT"`)
  - `FieldNames.MODIFIED.name()` for the time (which is the string `"MODIFIED"`)

  Usar `FieldNames.URL.name()` en lugar del string `"URL"` (por ejemplo) facilita modificar nombres de fields posteriormente, mantener la consistencia en los nombres y evitar errores de ortografía, etc.
- Primero, abre la clase `BuildWikiIndexBulk`. El método main ya está implementado, junto con parte del código de indexación. Sin embargo, en lugar de indexar solo la URL del documento (`tabs[0]`), deberás extender la implementación para incluir el título del documento (`tabs[1]`), el abstract del documento (`tabs[2]` – si está disponible, lo cual puedes verificar con `tabs.length > 2`), y la hora en la que el documento fue indexado (`System.currentTimeMilliseconds()`). Extiende el código para indexar estos campos.
- Ahora compila el archivo JAR con `build.xml`. Copia el JAR a tu directorio personal en el clúster. Ejecuta el JAR con un único INDEX-NAME (es mejor usar **minúsculas** para INDEX-NAME, ya que Elasticsearch puede tener problemas con nombres en mayúsculas):

  - `java -jar mdp-elasticsearch.jar BuildWikiIndexBulk -i` DATA `-igz -o` INDEX-NAME

  Donde los DATA están en `/data/uhadoop/shared/wiki/es-wiki-articles.tsv.gz`. Ignora el error relacionado con `Log4j`. Si el índice no estaba previamente definido, se creará con la configuración predeterminada (incluido el analizador estándar); cualquier campo en los documentos indexados también se configurará con los ajustes predeterminados. Mientras esperas...
- Ahora abre `SearchWikiIndex`. Nuevamente, el método principal ya está escrito para ti, al igual que parte del código de búsqueda. Sin embargo, en lugar de buscar solo en el título del documento, también deberías buscar en el abstract (si está disponible). Además, debes recuperar e imprimir los detalles de cada resultado en la salida estándar: su título (ya recuperado), su URL, su abstract y su puntuación de ranking (`hit.getScore()`). Extiende el código en estas líneas.
- Después de que la indexación haya finalizado (deberían leerse 964,838 líneas/artículos) y el código de búsqueda esté terminado, Recompila el jar, cópialo y realiza una búsqueda en el nuevo índice con:

– `java -jar mdp-elasticsearch.jar SearchWikiIndex -i` INDEX-NAME

Las búsquedas se reciben a través de la consola (mediante la entrada estándar) y los resultados se imprimen en la consola (salida estándar).

- Prueba con búsquedas como "`universidad de chile`", "`obama`", "`trump`". Registra los resultados. ¿Tienen sentido?
- Ejecuta cuatro búsquedas adicionales (que generen resultados no vacíos) y copia los resultados obtenidos.
- ENTREGA: Debes subir a u-cursos – `BuildWikiIndexBulk` y `SearchWikiIndex` –, junto con un archivo de texto que contenga los resultados de busqueda de "`trump`" y otras cuatro búsquedas de tu elección.
- OPCIONAL: Intenta variar los factores de aumento (boost) en `SearchWikiIndex` entre el título y el abstract, y observa cómo afecta los resultados de algunas consultas. ¿Cómo cambian los resultados?