Facultad de Ingeniería
Ingeniería Civil Informática
Universidad Autónoma de Chile

Proyecto unidad 2:

Elasticsearch

Asignatura:	Big Data
Integrantes:	Ignacio Arancibia
	Eduardo Silva
Profesor:	Edio Mardones

Facultad de Ingeniería
Ingeniería Civil Informática
Universidad Autónoma de Chile

Índice General

1. Introducción 2

2. Desarrollo 3

3. Conclusión 11

4. Bibliografía 12

1. Introducción

Wikipedia es la plataforma de colaboración abierta más activa de internet, este sitio se caracteriza por su gran cantidad de artículos en constante actualización. En pocos segundos miles de usuarios pueden editar un contenido de esta vasta enciclopedia, así generando un historial gigante de modificaciones en sus artículos.

Cada dato que se genera luego de una edición origina una gran cantidad de datos no estructurados que pueden ser usados para llevar a cabo un análisis sobre las conductas de los usuarios influyentes en este sitio web. El problema que se produce al trabajar con esta gran cantidad de datos resulta en el uso de complejos motores de búsqueda, como en este caso, Elasticsearch.

Elasticsearch, como ya fue nombrado anteriormente, es un motor de búsqueda avanzado, destinado a indexar y analizar datos. Para este proyecto el uso de este buscador fue esencial para el análisis del historial de ediciones para la página de Wikipedia. A través de este sistema de búsqueda se implementaron el uso de consultas avanzadas para permitir extraer información que podrá ser visualizada mediante gráficos que facilitan la interpretación de los datos, todo esto usando una biblioteca como matplotlib y utilizando el lenguaje python.

2. Desarrollo

Código utilizado para desarrollar la actividad:

- Conexión a Elasticsearch (más importaciones)

```python
import json
from elasticsearch import Elasticsearch
import matplotlib.pyplot as plt

# Conexión a Elasticsearch
es = Elasticsearch(
    hosts=["https://localhost:9200"],
    basic_auth=("edu", "123456"),
    verify_certs=False,
    ssl_show_warn=False
)

if es.ping():
    print("Conectado a Elasticsearch")
else:
    raise Exception("Error de conexión con Elasticsearch")
```

Primero tenemos las importaciones necesarias para el futuro funcionamiento del código. Lo sigue la conexión a Elasticsearch donde se utiliza el host general para la conexión al cluster, además se añade una función para verificar el estado de la conexión hacia Elasticsearch.

- Creación de un índice "wikipedia_edits" y definición de mapeo:

```python
# Nombre del índice
indexName = "wikipedia_edits"

# Definir el mapeo del índice
editsMapping = {
    "mappings": {
        "properties": {
            'titulo': {'type': 'text'},
            'resumen': {'type': 'text'},
            'nombreEditor': {'type': 'keyword'},
            'fechaEdicion': {'type': 'date'}
        }
    }
}

# Crear el índice si no existe
if not client.indices.exists(index=indexName):
    client.indices.create(index=indexName, body=editsMapping)
    print(f"Índice '{indexName}' creado exitosamente.")
else:
    print(f"Índice '{indexName}' ya existe.")
```

Al inicio se crea el índice junto a su nombre “Wikipedia_edits”, luego se define el mapeo para el índice, esto significa la creación de la estructura que especifica los tipos de datos para cada campo, en este caso se requieren los títulos, resumen, nombres de los editores y la fecha de edición. Finalmente está la creación del índice donde se utiliza una función para crear el índice si este no existe y dar aviso cuando este índice ya exista, además se especifica “cuerpo” utilizado por el índice.

- Ingesta de datos:

```python
def cargar_historial_ediciones(titulo_pagina, limite=7):
    url = f"https://en.wikipedia.org/w/api.php"
    params = {
        "action": "query",
        "titles": titulo_pagina,
        "prop": "revisions",
        "rvlimit": limite,
        "rvprop": "ids|timestamp|user|comment",
        "format": "json"
    }
    response = requests.get(url, params=params)
    data = response.json()
    page_id = next(iter(data['query']['pages']))
    revisions = data['query']['pages'][page_id]['revisions']
    cont = 1
    for revision in revisions:
        doc = {
            'titulo': titulo_pagina,
            'resumen': revision.get('comment', ''),
            'nombreEditor': revision['user'],
            'fechaEdicion': revision['timestamp']
        }
        client.index(index=indexName, document=doc)
        print(f"{cont}. {doc}\n")
        cont += 1

# Llamar a la función para cargar el historial de ediciones
cargar_historial_ediciones("Python (programming language)", limite=7)
```

La primera función que se nos muestra en el código nos permite consultar con la API de Wikipedia para obtener el conjunto de ediciones recientes de una página, luego almacena los datos obtenidos en el índice. Luego está la función que permite convertir la respuesta a un formato JSON. Ya casi terminando tenemos la iteración para las revisiones donde se imprime el documento con enumeración. Finalmente tenemos la última función que permite automatizar la obtención de las ediciones.

- Consultas avanzadas:

```python
# Consultas
def ver_datos(es, indice):
    query = {"size": 5, "query": {"match_all": {}}}
    resultados = es.search(index=indice, body=query)
    for i, hit in enumerate(resultados['hits']['hits'], 1):
        print(f"\nRegistro {i}:")
        print(json.dumps(hit['_source'], indent=4))

def consulta_avanzada(es, indice):
    query = {
        "size": 5,
        "query": {
            "bool": {
                "must": [
                    {"match": {"resumen": "science"}},
                    # {"range": {"timestamp": {"gte": "2021-01-01"}}}
                ]
            }
        }
    }

    resultados = es.search(index=indice, body=query)
    print("\nResultados de la consulta avanzada:")
    if resultados['hits']['hits']:
        for i, hit in enumerate(resultados['hits']['hits'], 1):
            print(f"\nRegistro {i}:")
            print(json.dumps(hit['_source'], indent=4))
    else:
        print("No se encontraron resultados.")
```

Empezamos con función para visualizar los 5 primeros datos del índice, luego tenemos la función que permite realizar una consulta especifica buscando en cada documento que coincida con el requisito, en este caso que “resumen” coincida con “science”

- Consulta ponderada:

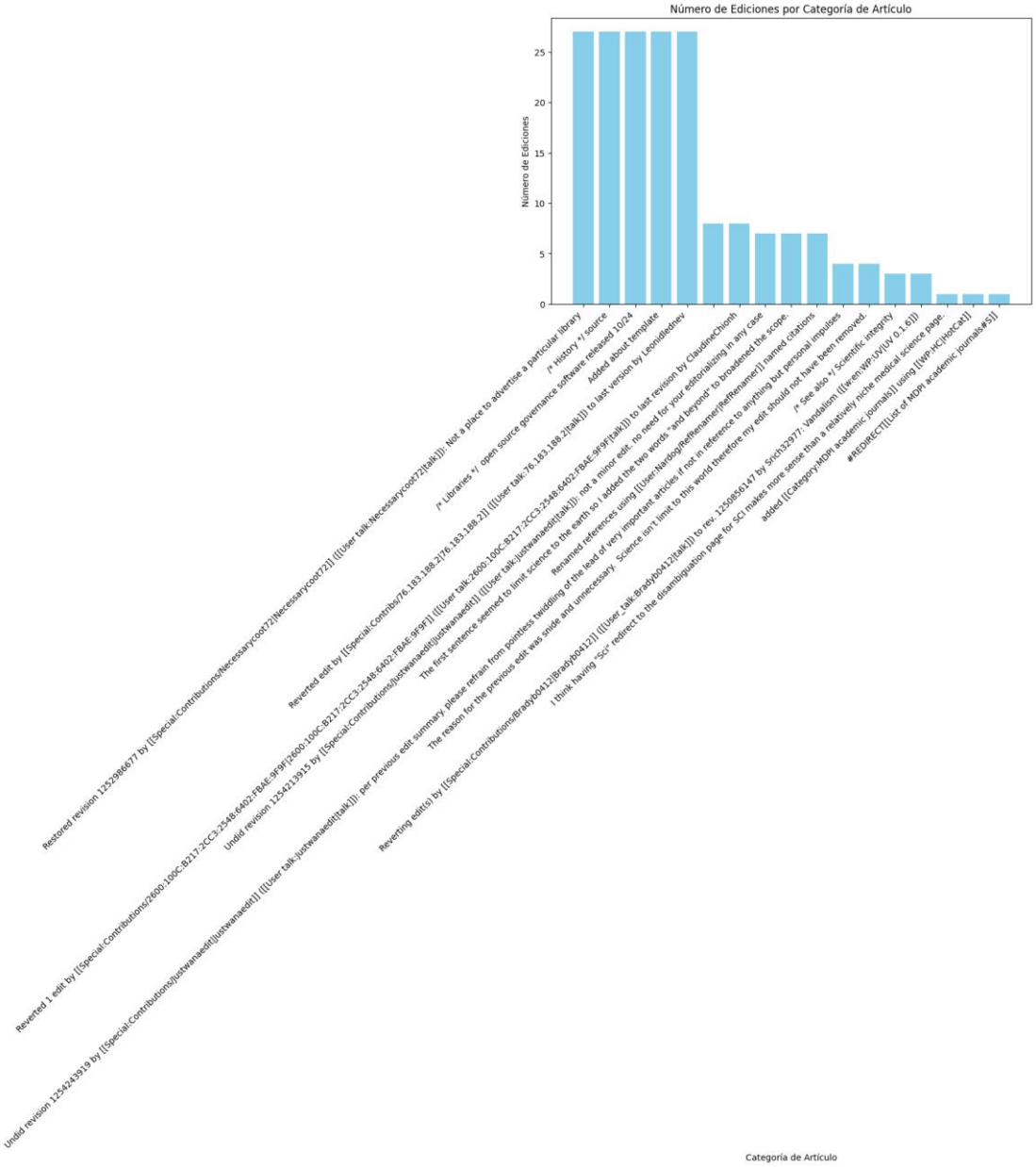
```python
def consulta_ponderada(es, indice, termino):
    query = {
        "size": 5,
        "query": {
            "bool": {
                "should": [
                    {"match": {"titulo": {"query": termino, "boost": 3}}},
                    {"match": {"resumen": {"query": termino, "boost": 1}}}
                ]
            }
        }
    }

    resultados = es.search(index=indice, body=query)
    print("\nResultados de la búsqueda ponderada:")
    if resultados['hits']['hits']:
        for i, hit in enumerate(resultados['hits']['hits'], 1):
            print(f"\nRegistro {i}:")
            print(json.dumps(hit['_source'], indent=4))
    else:
        print("No se encontraron resultados.")
```

La función para la consulta ponderada nos permite visualizar datos para términos que están con un “boost” o con una importancia mayor que los demás términos dentro de la misma búsqueda. Finalmente tenemos la función que nos permita mostrar los resultados, donde, si se encuentran datos los muestra y de lo contrario indica que no se han encontrado resultados.

- Visualización de datos:

```python
def grafico(es, indice):
    query = {
        "size": 1000,
        "query": {
            "match_all": {}
        }
    }

    resultados = es.search(index=indice, body=query)
    categorias = {}

    for hit in resultados['hits']['hits']:
        categoria = hit['_source'].get('resumen', 'Desconocido')
        if categoria in categorias:
            categorias[categoria] += 1
        else:
            categorias[categoria] = 1

    categorias_ordenadas = dict(sorted(categorias.items(), key=lambda item: item[1], reverse=True))

    plt.figure(figsize=(10, 6))
    plt.bar(categorias_ordenadas.keys(), categorias_ordenadas.values(), color='skyblue')
    plt.xlabel('Categoría de Artículo')
    plt.ylabel('Número de Ediciones')
    plt.title('Número de Ediciones por Categoría de Artículo')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
```

Finalmente nos encontramos con la visualización de los datos donde se crea el gráfico que recopila y genera un gráfico de barras que muestra el número de ediciones agrupadas por categoría, dentro de esta función se iteran los resultados obtenidos y se categorizan para finalmente visualizarse en un gráfico mostrado a continuación:


Número de Ediciones por Categoría de Artículo
Número de Ediciones
0
5
10
15
20
25
Restored revision 1252986677 by [[Special:Contributions/Necessarycoot72|Necessarycoot72]] ([[User talk:Necessarycoot72|talk]]): Not a place to advertise a particular library
/* History */ open source governance software released 10/24
Added about template
Reverted edit by [[Special:Contribs/76.183.188.2|76.183.188.2]] ([[User talk:76.183.188.2|talk]]) to last version by LeonidRedney
/* Libraries */ open source governance software released 10/24
Reverted 1 edit by [[Special:Contributions/2600:100C:B217:2CC3:2548:6402:FBAE:9F9F|2600:100C:B217:2CC3:2548:6402:FBAE:9F9F]] ([[User talk:2600:100C:B217:2CC3:2548:6402:FBAE:9F9F|talk]]) to last revision by ClaudineChionh
Undid revision 1254213915 by [[Special:Contributions/Justwanaedit|Justwanaedit]] ([[User talk:Justwanaedit|talk]]): not a minor edit. no need for your editorializing in any case.
The first sentence seemed to limit science to the earth so I added the two words "and beyond" to broadened the scope.
Renamed references using [[User:Nardog/RefRenamer|RefRenamer]] named citations
The reason for the previous edit was snide and unnecessary. Science isn't limit to this world therefore my edit should not have been removed.
Reverting edit(s) by [[Special:Contributions/Bradyb0412|Bradyb0412]] ([[User talk:Justwanaedit|talk]]): per previous edit summary, please refrain from pointless twiddling of the lead of very important articles if not in reference to anything but personal impulses
/* See also */ Scientific integrity
Undid revision 1254243919 by [[Special:Contributions/Justwanaedit|Justwanaedit]] ([[User talk:Justwanaedit|talk]]) to rev. 1250856147 by Srich32977: vandalism ([[en:WP:UV|UV 0.1.6]])
I think having "SCI" redirect to the disambiguation page for SCI makes more sense than a relatively niche medical science page.
added [[category:MDPI academic journals]] using [[WP:HC|HotCat]]
#REDIRECT[[List of MDPI academic journals#S]]
Categoría de Artículo

- Ejecución principal:

```
120  # Ejecución Principal
121  if __name__ == "__main__":
122
123      ver_datos(es, indice)
124      consulta_avanzada(es, indice)
125      consulta_ponderada(es, indice, "science")
126      grafico(es, indice)
```

Como último tenemos la función que nos permite ejecutar y llevar a cabo todo este código inicializando una a una.

3. Conclusión

Pese a las dificultades técnicas, para este proyecto se lograron completar todos los requisitos necesarios para abarcar el manejo de grandes volúmenes de datos dentro de la gran enciclopedia como es Wikipedia. A través de variadas consultas se extrajeron valiosos patrones de datos de colaboración y se comprendió de mejor manera el ajuste de los parámetros enfocados en la relevancia de las búsquedas, lo cual forma una parte esencial en el uso y manejo de resultados con un enfoque en la precisión de estas herramientas.

La visualización de los datos mediante las gráficas enseñadas al final del documento permitió una observación clara de la distribución de las ediciones por artículo, las cuales estuvieron centradas en la reversión de las mismas y en correcciones que fueron transformadas en poco tiempo.

4. Bibliografía

Index modules | Elasticsearch Guide [8.15] | Elastic. (s. f.). Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html