EII
Escuela de Ingeniería Informática de Oviedo

C. de Valdes Salas, 11
Tel. 98 510 27 96
eii@uniovi.es
33007 Oviedo

- ELASTICSEARCH -

Guillermo Facundo Colunga, Pablo Menéndez Suárez y Jorge Vila Suárez.

PROFESORADO:
Daniel Gayo Avello - dani@uniovi.es

Resumen. En este documento se presentan la propuesta, realización y conclusiones de la tercera práctica de la asignatura de Repositorios de la Información 2017 de la Universidad de Oviedo. Usando el Sistema de Recuperación de Información elasticsearch se explorará cómo se indexan documentos y a partir de ahí como implementar un sistema que permita, dada una o varias palabras, encontrar aquellos términos más relacionados dentro de nuestra colección de documentos indexados. Para ello exploraremos el algoritmo GND (Google Normalized Distance) que nos permite cuantificar la relación entre términos de una colección.

Palabras clave: elasticsearch, recuperación, información, GND, búsqueda.

Introducción

Esta práctica se enmarca en encontrar términos relacionados en una colección de tweets. Para ello, utilizaremos elasticsearch como sistema de recuperación de información y el algoritmo de Google Normalized Distance para cuantificar la relación entre los términos de la colección. A continuación realizaremos una aplicación en java que sea capaz de indexar los tweets y encontrar los términos más relacionados a un texto introducido (una o más palabras).

Elasticsearch

Elasticsearch es un sistema de recuperación de información que nos permite almacenar y procesar grandes volúmenes de datos en un tiempo muy reducido. Puede consultar más información en www.elastic.co.

Google Normalized Distance

Este algoritmo es us sistema de medición de similaridad semántica derivado de la búsqueda de términos en un sistema de recuperación de información como elasticsearch. A continuación se detalla el algoritmo.

$$NGD_{(x,y)} = \frac{max(\log f(x), \log f(y)) - \log f(x,y)}{\log N - min(\log f(x), \log f(y))}$$

Donde *f(x)* representa el número de hits para x, *f(y)* los hits the y, *f(x, y)* los hits de las dos palabras juntas y *N* representa el número de documentos en los que se ha buscado.

Metodología de trabajo

Para la realización del proyecto y dadas las dificultades que teníamos para reunirnos y trabajar sobre una única máquina adoptamos una metodología distribuida. Usamos Git como sistema de control de versiones, en su version de GitHub. Y Gradle como gestor de dependencias.

El trabajo y tareas las repartimos de forma que fueran equitativas de tal manera que más o menos todos realizamos el mismo trabajo. Eso sí, para mejorar nuestro rendimiento cada uno se centró en una parte del proyecto, interfaz de usuario, conexión con elasticsearch e indexación y búsquedas. Así mismo la evolución del proyecto puede diferenciarse en tres fases:

Primera fase

Empezamos el proyecto investigando qué lenguaje de programación tendría la curva de aprendizaje más rápida para implementar la API de elasticsearch. Nosotros, como estudiantes de la EII (Escuela de Ingeniería Informática) de Oviedo, somos nativos en java. Así que este fue el lenguaje escogido. A continuación seguimos nuestra investigación con el algoritmo GND y así implementarlo en java.

Finalmente teníamos una aplicación funcional que era capaz de buscar términos relacionados a un tercero a través del algoritmo GND implementado. Para ello:

1. Cacheábamos todas las palabras de la colección en un mapa junto con su número de apariciones.
2. Buscábamos en la colección el número de apariciones de la palabra x con todas las posibilidades de la palabra y (todas las palabras de la colección).
3. Aplicábamos GND a cada relación de la palabras y guardábamos en un mapa las palabras y su puntuación.
4. Recuperábamos del mapa las palabras más relacionadas.

Esta implementación fue descartada porque era extremadamente lenta ya que tenía que comprobar el número de apariciones de la palabra buscada con todas las posibilidades de palabras que había en la colección. Sin embargo en el proyecto final se incluye el paquete google con la implementación del algoritmo GND.

Segunda fase

Necesitábamos una implementación más rápida así que lo primero que hicimos fue reestructurar nuestro proyecto, al principio no usábamos ningún gestor de dependencias y cada uno importábamos una versión de las APIs distinta así que convertimos el proyecto en un proyecto de Maven y a partir de aquí comenzamos a investigar cómo mejorar el rendimiento.

La única forma era reducir el número de consultas realizadas a elasticsearch. Cada consulta consume un tiempo muy valioso. De esta forma investigamos y dentro de la API de búsqueda de elastic encontramos la funcionalidad de agregados, que a su vez incluye consultas de agregación de texto significativo. Estas consultas son muy potentes ya que realizan todo el proceso que habíamos realizado anteriormente en elastic. Y automáticamente, devuelve las palabras más relacionadas semánticamente. Así fue como como terminamos realizando una única consulta con elastic. (Se detalla a continuación).

C.1
Objetivo Consulta:
Sacar los términos semánticamente más relacionados con el [textoABuscar].
Código Consulta:

```
{
   "query":{
      "match":{
         "text":{
            “query”:[textoABuscar],
            "operator":"and"
         }
      }
   },
   "aggs":{
      "sample":{
         "sampler":{
            "shard_size":10000
         },
         "aggs":{
            "keywords":{
               "significant_text":{
                  "field":"text",
                  "filter_duplicate_text":false
               }
            }
         }
      }
   }
}
```

Tercera fase

La tercera fase se centraba en realizar un refinamiento del código, añadiendo un menú para el usuario y permitiendo también pasar algunos parámetros a través de la linea de comandos. Fue en esta fase cuando para insertar la librería willyOS, que permite agilizar tareas en java, decidimos pasar el control de dependencias a Gradle. Además de que esto nos permitió aprender un nuevo sistema de control de dependencias.

Es el código correspondiente a la finalización de esta fase el que se adjunta junto a este documento.

Instrucciones para la ejecución

Para una correcta ejecución por favor siga las instrucciones detalladas a continuación.

Elasticsearch

Inicie elasticsearch de forma normal y asegurase de que está disponible en el puerto 9300 para conexiones de API.

Proyecto de java

1. Descomprima el proyecto
2. Abra Eclipse (Oxigen) para que tenga reinstalado gradle.
3. En el explorador de proyector haga click en el botón secundario > Importar
4. Seleccione proyecto de Gradle.
5. En el asistente continue hasta el punto en el que se le pide que escoja su proyecto.
6. Escoja la carpeta donde está alojado el proyecto.
7. Finalizar.

El proyecto debería de estar importado correctamente y sin errores.

Ejecución

1. Ejecute la clase Main.java
2. Se le mostrará un menú con las opciones posibles.

Si selecciona Indexar la colección se le pedirá que inserte el número de tweets que desea indexar. Max 39940.

Si selecciona Buscar se le pedirá que inserte una palabra o palabras. Pulsando enter se realizará la búsqueda.

Si selecciona Salir se terminará la aplicación.

Bibliografía

Normalized Google Distance (Septiembre, 2017). Wikipedia. Recuperado el 1 de diciembre de 2017. https://en.wikipedia.org/wiki/Normalized_Google_distance

Elasticsearch API (6.0). Elasticsearch. Recuperado el 1 de diciembre de 2017. https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html

Gradle Documentation. Gradle. Recuperado el 1 de diciembre de 2017. https://gradle.org