Recuperación de información y web semántica
Master Universitario en Enxeñería Informática

Implementación de una aplicación web para la búsqueda en un subreddit

Explain Like I'm Five

Guillermo Alfonso Varela Chouciño
Luis Fernando Cruz Ramos

Universidad de A Coruña

22 de Noviembre de 2019

Índice general

Índice de figuras

1. Introducción

Esta es una práctica comprende la parte de Recuperación de Información de la asignatura, en la que se ha seleccionado como dominio el subreddit ***Explain Like I'm Five***.

Esta página de Reddit es un foro comunitario en el que los usuarios realizan preguntas que esperan ser respondidas por la misma comunidad de una forma simplificada y fácil de interpretar. En función de las valoraciones de los usuarios, tanto las preguntas como las respuestas podrán ser puntuadas positiva o negativamente. Además, se pueden filtrar los resultados por diferentes categorías como *Cultura*, *Matemáticas*, entre otras.

Para el desarrollo de esta página nos centraremos en diferentes elementos de este subreddit:

- Páginas de listado de publicaciones
- Páginas de preguntas y comentarios
- Puntuación de las preguntas y respuestas
- Categorías de las preguntas

En los siguientes apartados se describirá el proceso seguido desde la obtención de la información de estas páginas, hasta la consulta de contenido de estas páginas a través de una aplicación web.

Se describirá la estructura del motor de búsqueda y las herramientas seguidas.

1.1 Enlaces

El entorno generado estará disponible en el siguiente enlace:

`explainlikeimfive.live` `ec2-3-219-181-239.compute-1.amazonaws.com`

El subreddit original está disponible en los siguiente enlaces (versión nueva y crawleada):

`https://reddit.com/r/explainlikeimfive/top/?t=all&count=25`

`https://old.reddit.com/r/explainlikeimfive/top/?t=all&count=25`

2. Tecnologías utilizadas

2.1 ElasticSearch

Elasticsearch es un motor de búsqueda distribuido open source para cualquier tipo de datos, incluyendo textual, numérico, geoespacial, estructurado y no estructurado. Elasticserach está construido sobre Apache Lucene (ver glos. A) [1]. Es popular por la simpleza de sus APIs REST, su naturaleza distribuida, velocidad y escalabilidad.

Fue seleccionada para este proyecto además de las ventajas mencionadas anteriormente, por la integración que Spring ofrece mediante Spring Data (ver subsec. 2.2.2).

Además, el proyecto Spring Boot (ver subsec. 2.2.1) ofrece una autoconfiguración con Elasticsearch, facilitando la integración de estas tecnologías.

2.2 Tecnologías Java

2.2.1 Spring Boot

Spring Boot es un framework Java que facilita la creación de aplicaciones Spring, simplificando la configuración y la integración con diferentes librerías internas y externas.

El arquetipo sobre el que se contruyó la solución implementada usando Spring Initializer, una herramienta que proporciona Spring Boot para crear facilmente las bases de un proyecto.

Esta tecnología fue escogida ya que permite una rápida puesta en marcha de una aplicación web, permitiendo así centrarnos en el desarrollo del motor de búsqueda.

Además, Spring Boot ofrece diversos dependencias llamadas "Starters", que simplifican la gestión de dependencias agrupándolas en unidades más fáciles de importar. Este es el caso de `spring-boot-starter-data-elasticsearch`.

2.2.2 Spring Data Elasticsearch

Spring Data, en general, proporciona una abstracción sobre la capa de acceso a datos mediante el uso de los llamados repositorios. Reduciendo así la repetición innecesaria de código.

En concreto, Spring Data Elasticsearch proporciona esta funcionalidad para la comunicación con Elasticsearch.

2.2.3 Crawler4j

Crawler4j es un web crawler open source para Java que proporciona una interfaz simple que permite poner en marchar un crawler multi thread en pocos minutos.

Esta tecnología fue seleccionada debido a su popularidad en GitHub, mantenimiento frecuente y su facilidad de uso.

A diferencia de algunos crawlers, este proyecto no permite el parsing de páginas obtenidas. Esta es la razón de usar una tecnología de parsing como lo es Jsoup.

2.2.4 Jsoup

Jsoup es una librería Java dedicada al parsing de código HTML. Proporciona una API conveniente para extraer y manipular datos, similar a los metodos proporcionados por el DOM, CSS y jQuery.

En este caso, Jsoup fue utilizado para el parsing de los documentos HTML obtenidos del crawler. Esto permite la extración de la información relevante de dichas páginas.

2.3 Vue.js

Vue.js es un framework JavaScript para la creación de aplicaciones de una sola página (SPAs, ver A). La aplicación web implmentada para este proyecto utiliza Vue.js para la creación de un interfaz para el usuario.

3. Arquitectura del motor de búsqueda y diseño

En este capítulo se describirán los bloques principales que componen el motor de búsqueda implementado para este proyecto. En la figura 3.1 aparece representado este sistema en su totalidad, junto con las tecnologías relacionadas en cada bloque.

3.1 Text acquisition

3.1.1 Crawling

En las arquitecturas de un motor de búsqueda, el ***crawler*** tiene la principal tarea de recorrer el dominio y adquirir el contenido para su posterior uso en el motor de búsqueda.

En este caso, se ha implementado un *web crawler* que siguiera la estrategia de *site search* para el recorrido del subreddit, ya que solamente se recogen las páginas que listan las publicaciones del dominio y las páginas de las publicaciones y sus comentarios.

Por lo tanto, situamos la *seed* del crawler en la página que contiene los resultados populares de todos los tiempos. Desde este punto se visitan las páginas de las publicaciones y se avanza en la paginación del listado de publicaciones. Para visitar únicamente las páginas deseadas, se analiza la forma de la URL.

Solamente cuando se recorren las páginas las obtenemos en formato HTML y pasan a ser transformadas.

3.2 Text transformation

3.2.1 Parser

Posterior al proceso de crawling, llega el momento de transformar los datos recibidos por el crawler en formato HTML. Esta conversión se realiza mediante un parser que divide el documento HTML en componentes y que preparará objetos del tipo *Question* y *Answer* para su procesado o indexación.

3.2.2 Stopping y Stemming

El proceso de ***stopping*** tiene la tarea de eliminar palabras comunes de la lista de términos que pasarán a formar parte del índice. Estas palabras comunes son definidas en una lista denominada *stop-word list*.

Stemming es el proceso de agrupar palabras que pertenezcan a la misma raíz (i.e. *fish* y *fishing*).

Las tareas de *stopping* y *stemming* quedan delegadas al analizador proporcionado por Elasticsearch que especificamos en el momento de indexar y buscar.

3.2.3 Information extraction

En nuestro proyecto obtenemos la siguiente información de la página:

- Identificador de la pregunta
- Identificadores de las respuestas
- Texto de la pregunta
- Textos de las respuestas
- Puntuación de la pregunta (*karma*)
- Puntuaciones de las respuestas
- URL de la pregunta (un *permalink* que ofrece reddit)
- URLs de las respuestas

La obtención de la categoría de la pregunta, para permitir la búsqueda por ***facets***. La búsqueda por facetas permite explorar una colección de resultados aplicando diversos filtros sobre los resultados. En el caso de este proyecto, permitimos al usuario hacer una búsqueda facetada si lo desea, indicando la categoría de la pregunta.

3.3 Index creation

Las tareas de creación de índices son realizadas internamente por Elasticsearch. Simplemente explicitamos qué campos deseamos que se indexen y el analizador que se utilizará; en nuestro caso usamos uno de los analizadores que dispone Elasticsearch.

3.4 User interaction

El usuario proporciona una cadena de texto desde el cliente web. Para la selección de documentos y su *scoring*, que es la puntuación del nivel de relevancia de un documento, se toma esa cadena y se construye una consulta en un formato proporcionado por Elasticsearch.

Para la selección y cálculo de puntuación de los documentos tenemos en cuenta:

- El texto de la respuestas o el texto de la pregunta contienen algún término de los introducidos.
- La pregunta de la respuesta pertenece a la categoría especificada, si es el caso.
- Para esto se utiliza la query de Elasticsearch *match* sobre estos campos, lo que calcula la puntuación basándose en diferentes factores. [2] [3]
- Se hace una búsqueda por facetas, excluyendo las respuestas cuyas preguntas no pertenezcan a la categoría especificada, si es el caso. La categoría no influye en ningún caso en el cálculo de la puntuación del resultado. Para esto se utiliza en Elasticserach una *query* en *filter context*. [4]
- Se hace un *boost* o una suma extra a la puntuación de un resultado en función del *karma* de la respuesta. El *karma* es una puntuación en Reddit que los usuarios dan a una respuesta que consideran relevante. [5]
- Se tiene en cuenta la *fuzziness* o borrosidad de los términos introducidos; esto es la coincidencia con términos similares a los de entrada basándose en la distancia de Levenshtein [6]. Esto está implementado por Elasticsearch y hemos dejado la configuración por defecto. [7]

Una vez obtenidas las respuestas, se devuelven al usuario de manera ordenada y paginada.

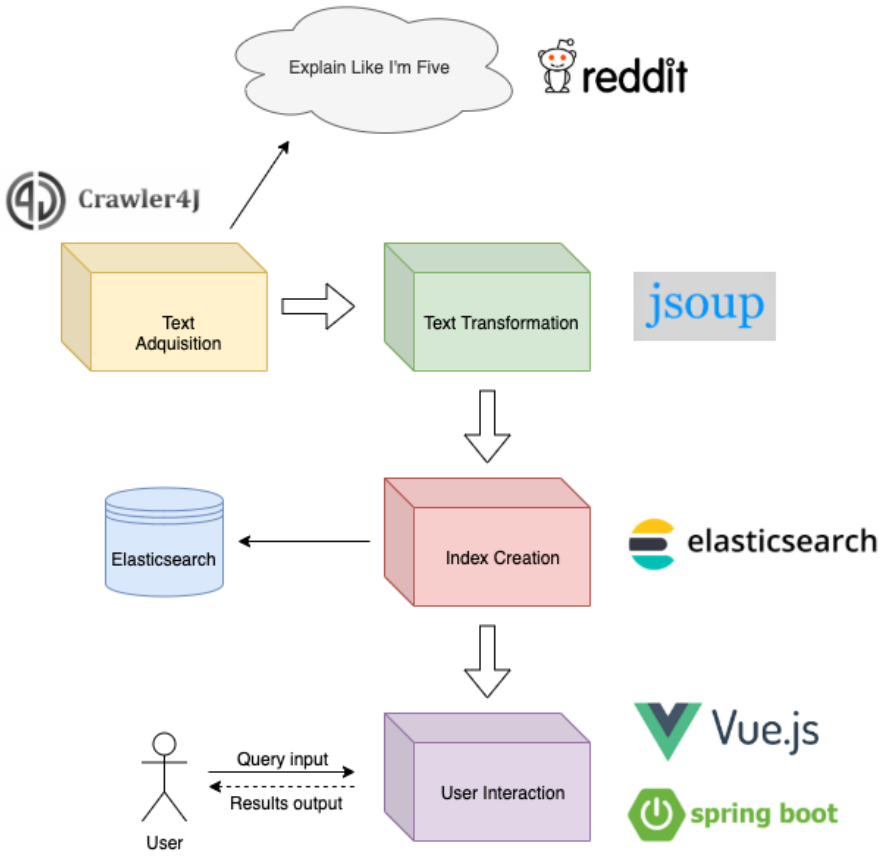

Figura 3.1: Arquitectura del motor de búsqueda del proyecto

4. Desarrollo del proyecto

En este capítulo expondremos detalles sobre la implementación de lo descrito en el capítulo anterior.

4.1 Crawling

Se hace crawling en un servidor Java mediante la librería Crawler4j. Esta librería permite que configuremos diversas opciones como las siguientes:

- *User agent*; fue necesario ser modificado para que el host no nos bloquease.
- Número de crawlers; en nuestro caso establecemos 4 crawlers.
- Número máximo de páginas a recorrer; limitamos a 15000 el número de páginas recorridas en total.
- Profundidad del crawling; 15, ya que queremos que profundice en las páginas preguntas.
- *Seed*; semilla del crawling, establecido en `https://old.reddit.com/r/explainlikeimfive/top/?sort=top&t=all`. Elegimos el subdominio `old` ya que la versión nueva de la web tiene paginación infinita por JavaScript lo que dificultaría el recorrido.

4.2 Parsing

Se parsea el HTML descargado por Crawler4j utilizando Jsoup. Se obtiene la información deseada principalmente utilizando la API que ofrece para la selección mediante selectores CSS.

Por ejemplo: `Elements questionResponses = doc.select(".commentarea >.sitetable > .comment:not(.stickied) >.entry")` selecciona los elementos de texto de todos los comentarios de la página de una pregunta

4.3 Indexing y Querying

El indexado y las consultas son realizadas utilizando los métodos de los repositorios proporcionados por Spring Data Elasticsearch.

Spring Data permite mapear los documentos indexados como objetos Java. Esto nos facilita indicar qué campos queremos que se indexen y su analizador:

```
@Field(type = FieldType.Text, analyzer = "english")
```

Para las consultas ha sido necesario implementar métodos personalizados en los repositorios utilizando el Query DSL [8] de Elasticsearch.

Esta es la consulta que se realiza para obtener las respuestas:

```
{
   "query": {
      "function_score": {
         "query": {
            "bool": {
               "must": {
                  "multi_match": {
                     "query": "?0",
                     "fields": [
                        "text^2",
                        "question.text"
                     ],
                     "type": "most_fields"
                  }
               },
               "filter": [
                  {
                     "term": {
                        "question.category": "?1"
                     }
                  }
               ]
            }
         },
         "functions": [
            {
               "field_value_factor": {
```

```
                    "field": "karma",
                    "factor": 0.1,
                    "modifier": "sqrt",
                    "missing": 1
                }
            }
        ],
        "boost_mode": "sum"
    }
  }
}
```

Elementos a destacar sobre la query anterior:

- Se realiza un `multi_match` [3] en Elasticsearch sobre el texto de la respuesta y de la pregunta utilizando la entrada del usuario.
- Se hace una query en contexto de `filter` [4] para solo tener en cuenta las respuestas a preguntas con esa categoría.
- Se combinan las dos anteriores mediante una `Bool Query` [9]
- Mediante una función `field_value_factor` [10] se incrementa la puntuación generada sobre el documento en base al karma de la respuesta. Hay que tener en cuenta de que no se ha analizado en profundidad el nivel de influencia que sería deseable, por lo que hemos optado por ser prudentes debido a que sobrevalorar el karma podría disparar la puntuación de manera indeseada.

4.4 User interaction

En el proyecto, la interacción del usuario con el sistema se ha realizado mediante la implementación de una aplicación web. Esta web se conectará con Elasticsearch para realizar las consultas solicitadas por el cliente y devolverá resultados.

La aplicación se divide en un módulo frontend y un módulo backend. En el módulo frontend se ha implementado la interfaz de usuario utiizando Vue.js. Por otro lado, el módulo backend fue implementado un API REST utilizando Spring Boot.

Figura 4.1: Página principal de la aplicación

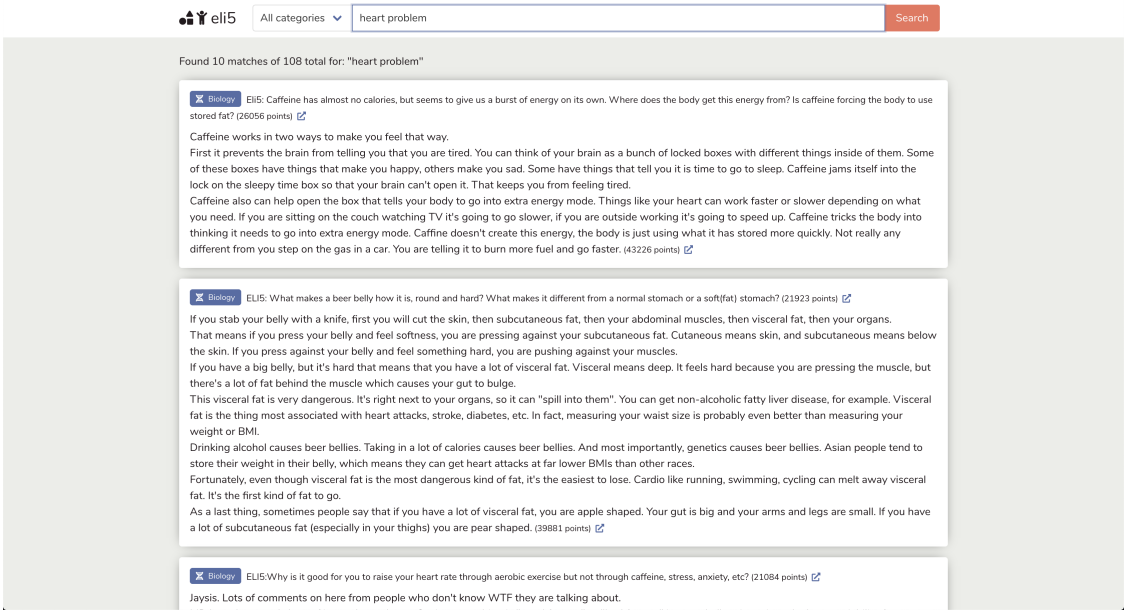

Figura 4.2: Página de resultados

5. Manual

5.1 Instalación y desarrollo

Requisitos:

- Elasticsearch 6.8.4
- Java 1.8 y Maven
- Node 12.3.0 y NPM para ejecutar webpack-dev-server

Para compilar el proyecto, desde la raíz se ejecuta el comando:

```
\$ mvn clean install -DskipTests
```

Antes de lanzar la app, se iniciará Elasticsearch mediante el comando:

```
\$ elasticsearch
```

Para ejecutar la aplicación se utilizará el comando:

```
\$ mvn --projects backend spring-boot:run
```

Además, se puede lanzar el servidor de desarrollo de Vue

```
cd frontend
npm run serve
```

Una vez lanzada la aplicación Spring, podremos acceder a ella desde el puerto `8088`

```
http://localhost:8088/
```

El servidor de desarrollo de Vue es accedido desde el puerto `8080`

```
http://localhost:8080/
```

5.2 Uso

El crawling no es realizado automáticamente, existe un endpoint en la aplicación que inicia una secuencia de crawling, populando las respuestas sobre las que se harán las búsquedas.
Para iniciar el crawling, basta con visitar la URL

```
http://localhost:8088/api/index
```

Una vez realizado, se puede comenzar a realizar búsquedas a través de la aplicación web.

A. Glosario de términos

Lucene es una API en código abierto para recuperación de información, originalmente implementada en Java por Doug Cutting. Es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo.

SPA - Simple Page Application

Bibliografía

[1] The Apache Software Foundation. Apache Lucene. [Online]. Available: https://lucene.apache.org/

[2] Elastic. Match query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-match-query.html

[3] ——. Multi Match Query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-multi-match-query.html

[4] ——. Query and filter context | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-filter-context.html

[5] ——. Function Score Query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-function-score-query.html

[6] D. M. . T. S. Bruce Croft, *Search Engines: Information Retrieval in Practice*. Pearson Education, Inc., 2015.

[7] Elastic. Fuzzy Query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-fuzzy-query.html

[8] ——. Query DSL | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl.html

[9] ——. Bool Query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-bool-query.html

[10] ——. Function Score Query | Elsaticsearch Reference [6.8]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-function-score-query.html#function-field-value-factor