Recuperación de Información y Recomendaciones en la Web

Obligatorio 2017

Instituto de Computación

Facultad de Ingeniería - UdelaR

Integrantes:

Santiago Camou

Octavio Perez Kempner

Juan Saavedra

Nicolas Urruty

Índice

1. Introducción

Los estudiantes de la Facultad de Ingeniería suelen entregar objetos perdidos en las recepciones del Edificio Central y del Aulario, así como también reportarlos en distintos grupos de usuarios en redes sociales como Facebook.

Esta realidad plantea algunos desafíos que motivan el problema a abordar en este proyecto, el cual será el de centralizar la información de documentos de identidad encontrados a partir de distintas fuentes.

En las siguientes secciones se presentarán los desafíos, propuestas y avances que se han tomado respecto a dicha problemática finalizando con un balance y propuestas para trabajo a futuro.

2. Problema y Motivación

La búsqueda de documentos de identidad resulta más costosa que la de otro tipos de objetos entre los estudiantes universitarios. En primer lugar, debido a la heterogeneidad de las fuentes (grupos y páginas web) en muchos casos los documentos permanecen extraviados ya que: o bien la persona que lo extravió no lo busca en los mismos sitios en donde se publicó el hallazgo, o bien sus documentos son publicados en sitios con baja actividad de usuarios). Por lo tanto, resulta imperioso poder recabar la información de documentos utilizando distintas técnicas para las distintas fuentes y lograr acceder a a la mayor cantidad posible de resultados desde un único lugar.

En segundo lugar, la búsqueda de documentos de identidad no resulta tan sencilla como la búsqueda de otros objetos dentro del ambiente universitario. A efectos de clarificar este punto listaremos a continuación distintas frases tomadas de publicaciones en grupos de Facebook:

- "*Pidieron la cédula en el a21 en el parcial de economia? Porque a mi solo me pasaron un papel para firmar*"
- "*Che, encontré una cédula en el baile. Camila alejandra rodriguez El lunes la dejo en la pecera*"
- "*se puede ir a un parcial con la boletera en lugar de la cedula?*"
- "*Buenas gente. Apareció esta cédula en el baile ayer, si alguien la conoce avise. El lunes la dejo en la pecera de la fing.*"
- "*Alguien sabe si se puede dar parciales con la denuncia de robo de cédula?*"

- "*Hola, anoche se encontró un paraguas en el B01. Color negro, de mango curvo. Está en la pecera.*".

A partir de estos ejemplos podemos concluir que en lo que refiere a los documentos (a diferencia de lo que ocurre con los paraguas) se tienen distintas publicaciones que si bien hablan sobre estos por cuestiones admnistrativas ligadas al ámbito estudiantil, no refieren a su hallazgo con lo cual resulta importante poder filtrar entre ambos tipos de mensajes.

Finalmente y en tercer lugar, a partir de los ejemplos anteriores podemos ver que los hallazgos cuentan con distinto tipo de información como ser: dónde se realizó el mismo, a quién pertenece, dónde puede encontrarse, etc...
Esto genera que sea de interés poder identificar el tipo de información relevante y asociarla a los datos de manera que la búsqueda del documento resulte lo más satisfactoria posible.

3. Enfoque de la solución

A partir de un relevamiento sobre los distintos tipos de fuentes, se concluyó que las principales son grupos de Facebook y páginas web como `https://www.perdiencontre.com.uy`. Por estos motivos se entendió necesario trabajar tanto a partir de la utilización de API's [1] como con técnicas de scraping y con un lenguaje como Node.js para implementar el código necesario.

Es importante resaltar que las técnicas de scrapping permiten escalar a una gran cantidad de sitios pero no de forma muy sencilla ya que hay que tener en cuenta la heterogeneidad de estructura de los diferentes sitios.
Distinto es el caso de Facebook donde la utilización de la API permite acceder a gran cantidad de datos de forma homogénea. Sin embargo, Facebook requiere de permisos especiales para acceder a grupos privados que llevan a que o bien sea necesario registrar el proyecto o bien se obtengan los permisos (de administrador) para cada grupo en particular. Esto limita las posibilidades de escalar dentro de Facebook para grupo arbitrario aunque sí se puede hacer en grupos públicos.

En cuanto al manejo de datos, la información recuperada se persistirá y manipulará utilizando ElasticSearch.
ElasticSearch es un motor de búsqueda y recuperación de documentos de tipo JSON basado en Lucene, que puede proveer a diferentes aplicaciones de capacidades de búsqueda a texto completo a través de una API web RESTful. ElasticSearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache.[1]

[1] Extraído de `http://www.davinci-ti.es/introduccion-a-elasticsearch-y-como-instalarlo/`

4. Diseño e implementación

El objetivo principal en la etapa de diseño fue obtener una arquitectura simple y escalable que pudiera mantener una alta disponibilidad y un motor de búsqueda performante. Esto motivó la elección de Elasticsearch como motor de búsqueda y base de datos principal.

En primer lugar, Elasticsearch permite una arquitectura distribuída, escalable y de alta disponibilidad. Todo esto con capacidades de búsqueda y análisis en tiempo real y un esquema de indexación flexible y performante.

En segundo lugar, considerando a Elasticsearch como base de datos primaria cabe destacar lo siguiente: puede utilizarse como una base de datos NoSQL aunque existen algunas dificultades en su utilización para dicho fin que llevan a que se recomiende utilizarla como base de datos secundaria y no primaria. El principal problema reportado tiene que ver con la pérdida de datos al realizar inserciones. Para entender esta problemática se recomienda ver lo discutido en [5] y [6].
A pesar de esto, en este proyecto se tomó la decisión de trabajar con Elasticsearch como base de datos primaria por los siguientes motivos:

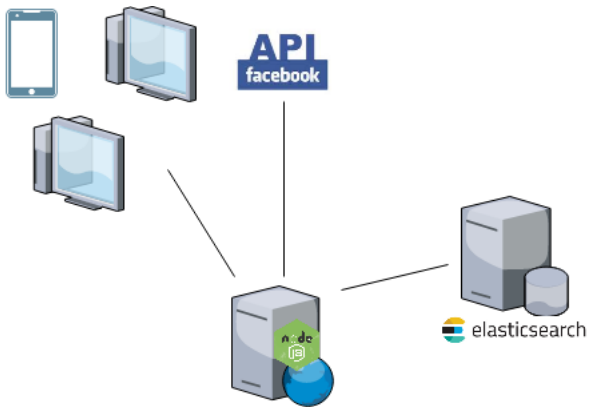
- Los escenarios en donde resulta más probable la pérdida de datos son aquellos en los que se cuenta con un sistema altamente distribuído y con distintos componentes que interactúan a través de distintos casos de uso con ElasticSearch. Este no es el caso del proyecto propuesto dado que se basa en una arquitectura simple (con pocos componentes).

- Si bien potencialmente se podrían manejar fuentes muy heterogéneas, los tipos de documentos no lo serían tanto por lo que resulta sencillo manejar un esquema de backup para mitigar posibles pérdidas. Esto es algo que no podría hacerse en otras condiciones pero que es factible para este proyecto.

Finalmente en cuanto a la implementación, se decidió trabajar con Node.js debido a la ventaja que se tenía al trabajar con objetos JSON. Estos aspectos serán presentados en más detalle en la sección siguiente.

4.1. Arquitectura del sistema

A continuación se muestra la arquitectura propuesta en la que se muestra la interacción del programa con los distintos componentes.



Básicamente dicha interacción consiste en:

- Extraer las publicaciones de las fuentes de internet (para este proyecto Facebook).
- Persistir los datos en la base de Elasticsearch interactuando directamente con la API que este provee.
- Desde la UI se realizan las consultas por palabras clave y se ejecuta un nuevo post a la base de datos para obtener los resultados y mostrarlos.

En cuanto a Facebook, este provee de una API simple para acceder a las publicaciones mientras que los resultados de las consultas resultan fáciles de interpretar dado que en todo momento se opera con objetos JSON.

Con respecto a recabar información de otras fuentes, si bien en este proyecto únicamente se implementó Facebook es importante presentar a continuación una discusión sobre el impacto que tendría incorporar otras fuentes en la arquitectura propuesta.

En particular lo que más interesa al problema son los sitios web dado que son una fuente de datos que enriquece la base de datos y por ende incrementa el rendimiento del sistema de búsqueda. Muchos sitios no cuentan con una API pública para acceder a sus publicaciones, por lo tanto es necesario aplicar técnicas como web scrapping. Para aplicar scrapping con Node.js se puede utilizar la libreria Cheeiro, que simplifica esta tarea creando una estructura de objetos (JSON) a partir de un HTML.

Cada caso de web scrapping resulta distinto según el sitio web por lo que ese camino no es sencillamente escalable como lo sería el caso de Facebook de no ser por los problemas de obtención de permisos para grupos privados. De todas formas, identificando sitios de interés y obteniendo los datos en objetos JSON la integración con el sistema (que opera con este tipo de objetos) sería directa por lo que la arquitectura propuesta resulta lo suficientemente flexible para este fin.

5. Funcionalidades y utilización

El prototipo realizado consta de una aplicación realizada en Node.js la cual al ser iniciada realiza las siguientes acciones:

- Comienza a recibir solicitudes en los puertos 8091 y 8000.
- Al recibir un GET en el puerto 8000 realiza una consulta a la API de Facebook la cual trae todos los posts de la pagina "Que joraca.[en] función de parámeros configurables como la fecha de comienzo.
- Inserta los post extraídos de la web en la base de Elasticsearch.
- Finalmente realizando una solicitud GET al 8091 se puede consultar la UI (realizada en Angular.js). Esta UI permite realizar consultas utilizando los siguientes campos:
  - Campo de palabras clave. Se ingresan con un espacio en blanco entre cada una palabras que usuario quiere que se utilicen en la búsqueda.
  - Un selector de tres opciones, Cédula, Boletera, Tarjeta. Esta opción también es incluida en la búsqueda.
  - Un selector de tres valores de distancia. Esta refiere a la distancia de Levenshtein, distancia de edición o distancia entre palabras que es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra.

    Por ejemplo, **encontré** y **encontró** tendrían una distancia de 1.

El usuario entonces puede ingresar las palabras clave que desee, seleccionar el tipo de documento y la distancia que prefiera.

Al presionar **buscar** se realiza una solicitud POST al servidor web Node el cual a su vez realiza un **search** en la base de Elasticsearch. Esta búsqueda tiene dos características importantes:

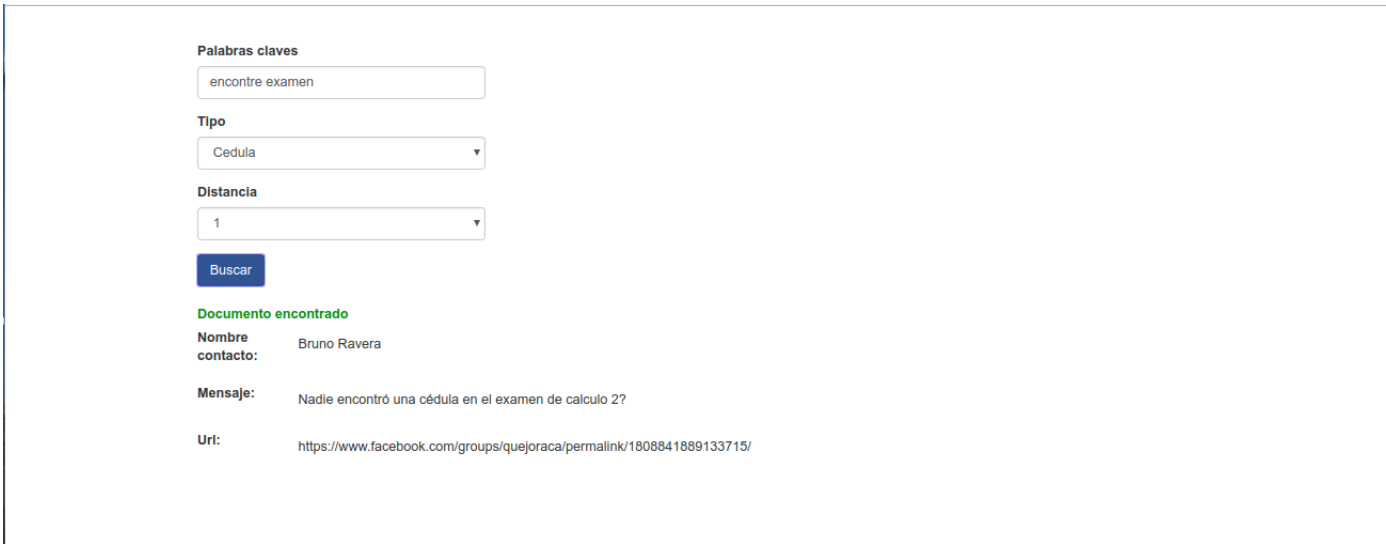
La primera es que obliga a que todas las palabras claves como el tipo elegido estén en los post recuperados. Por otra parte, el valor de distancia permitirá que igual se recuperen instancias del mismo termino en formas distintas.

Finalmente cuando se obtiene la respuesta de Elasticsearch, se despliegan los resultados en la UI. En la misma se incluyen el nombre del autor del post, el texto del mensaje y la url de Facebook donde poder ir a visualizar la publicación.

6. Evaluación y resultados

En secciones anteriores se comentó la posibilidad de encontrar publicaciones que contengan palabras relativas a documentos pero que no refirieran precisamente a la pérdida o hallazgo del mismo. A continuación se muestra uno de los resultados de búsqueda obtenidos que evidencia dicha problemática:

Si el usuario utiliza más filtros en su búsqueda como se muestra a continuación, las búsquedas resultan más satisfactorias. Notar que en el siguiente ejemplo al buscarse con distancia 1 se retorna una publicación en donde alguien **encontró** una cédula cuando el usuario había utilizado la palabra clave **encontre**.



Los ejemplos anteriores ponen de manifiesto la importancia de agregar palabras claves como **encontrar** o **dejar** a la hora de realizar la búsqueda. A continuación se muestra un resultado exitoso de búsqueda aplicando lo dicho anteriormente.

Una conclusión en base a los casos presentados podría ser que por defecto el sistema utilice una serie de palabras claves como las antes mencionadas para así evitar búsquedas muy abiertas por parte del usuario. De todas formas se concluye que los resultados son satisfactorios en términos generales dado que con pocas palabras claves y manejando la distancia se puede obtener la información relevante al interés del usuario.

7. Conclusiones

En base a lo expresado en la sección anterior y considerando los resultados obtenidos por este prototipo de la herramienta, se observa claramente el potencial de la misma. Este prototipo usa una indización simple de palabras presentes en publicaciones (Elasticsearch se encarga de hacerlo) para que luego, usando algunas palabras claves y aplicando cierta tolerancia en distancia de Levenshtein entre palabras como forma de normalización sencilla, los resultados sean buenos. Esto hace pensar que la precisión de las búsquedas si se aplicaran técnicas más avanzadas de normalización de palabras resultaría más satisfactoria aún.

También se vió que incorporando sitios y grupos de facebook a la misma se podría obtener un buen repositorio centralizado de publicaciones de documentos perdidos, siendo de gran ayuda para que el usuario pueda encontrar lo que busca.

Por otra parte al trabajar en esta herramienta se pudo dimensionar el potencial que tiene el área de recuperación de datos en la web y tener un primer acercamiento a tecnologías (principalmente Elasticsearch) que de otro modo no hubiera ocurrido.

Finalmente, un problema que se puede observar es que muchas publicaciones contienen solamente una foto del documento encontrado y tienen poco o nada de texto agregado que clarifique las características del hallazgo. Este tipo de publicaciones no serían recuperables utilizando la herramienta ya que la búsqueda se hace principalmente por palabras claves. Esto si bien es un problema para la herramienta en el estado actual, también marca un camino a seguir que será desarrollo en la siguiente sección.

8. Trabajo Futuro

Considerando que muchos estudiantes reportan fotos de los documentos, la extensión más notoria a este proyecto sería la incorporación de un módulo de escaneo de imágenes en la aplicación. El mismo podría utilizar Google Vision, una API proporcionada por Google para reconocimiento de imágenes. Con la misma, la aplicación podría permitir subir una foto de la persona y en base a la misma y a las imágenes recuperadas de las fuentes, tratar de encontrar alguna publicación donde la imagen coincida.

Por otra parte, en 2015 uno de los grupos [7] realizó un proyecto de búsqueda por texto en videos e imágenes que podría integrarse para que a partir de la foto de una cédula se pueda obtener el nombre de la persona y de esta forma incluir más información sobre el hallazgo al momento de almacenar la información del mismo.

9. Referencias

Referencias

[1] Facebook's Graph API.

`https://developers.facebook.com/docs/graph-api/overview`

[2] Ejemplo de uso: librería cheerio (Node.js).

`https://www.youtube.com/watch?v=HP-HP0jPzMQ`

[3] Rao, Dalip. Notas de curso: IR for Social Media. Johns Hopkins University.

`http://www.cs.jhu.edu/~delip/teaching/social-media-IR.pptx`

[4] Vasudevan, Sudharsan. Zhang, Liangliang. 2014 fall CS224W project, Stanford University. Facebook User Networks Based on Information Retrieval from Media Fan Pages: Network Analysis and Recommendation System.

`http://snap.stanford.edu/class/cs224w-2014/projects2014/cs224w-33-final.pdf`

[5] Elasticsearch Resiliency Status. Documentos oficiales del equipo de Elasticsearch.

`https://www.elastic.co/guide/en/elasticsearch/resiliency/current/index.html`

[6] Elasticsearch as a primary database. Discusión oficial sobre Elasticsearch como base de datos primaria.

`https://discuss.elastic.co/t/elasticsearch-as-a-primary-database/85733/10`

[7] H. Esteves, N.Tomeo, M. Rodal. Búsqueda por texto en imágenes. Proyecto de curso: Recuperación de Información y Rendaciones en la Web.

`https://eva.fing.edu.uy/pluginfile.php/99370/mod_resource/content/1/grupo-20-informe-final-webir.pdf`

10. Anexo: Instructivo de Instalación

- Instalar node.js desde la pagina oficial `https://nodejs.org/es/download/`
- Instalar elasticSearch desde pagina oficial `https://www.elastic.co/downloads/elasticsearch`
- Ejecutar el comando npm install
- Ejecutar el comando node main.js
- Mediante Chrome o Postman realizar un GET al puerto 8000. (Esto descarga los post desde facebook)
- Desde un navegador ir a `http://localhost:8091` para utilizar la UI