Recuperación de Información y recomendaciones en la Web

Integrantes:

Juan Medina 4.921.887-8

Germán Costabel 4.778.044-9

Índice

Introducción

En este proyecto buscamos generar un buscador de mayor utilidad para la página de noticias de presidencia (https://www.gub.uy/presidencia/comunicacion/noticias). El buscador presente en la página no da opciones para realizar búsquedas detalladas, en el ámbito periodístico (y también ciudadano) la información de presidencia es un recurso importante, sin embargo, encontrar información en ella puede presentar dificultades. En este proyecto intentamos generar un prototipo de buscador que pueda solventar estos problemas

Problema

El centro del problema es la gran cantidad de artículos presentes en la web de presidencia sin una forma razonable de poder filtrarlos o buscar algo particular entre ellos. Si no se cuenta con el título exacto de un artículo dado es casi imposible poder llegar a él a través de la página o utilizando herramientas externas como google.

El buscador proporcionado por la web no cuenta con herramientas básicas que ayuden a buscar algo tan simple como conseguir todos los artículos que mencionan covid durante un periodo de tiempo, esto es casi imposible con las herramientas actuales.

Actualmente según la página de presidencia se cuentan con 30.844 artículos, y cada día se agregan más. Contar con una herramienta que permita buscar en un corpus de este tamaño permite un mayor acceso real a la información provista por el estado. Esto beneficia fuertemente tanto a periodistas que busquen información que ha provisto presidencia previamente como al ciudadano común que busque acceder a esta información.

Enfoque de la solución

La solución se enfoca en intentar implementar la mayor cantidad de casos de uso y herramientas a un buscador comprometiendo lo menos posible la escalabilidad del proyecto a futuro. Se busca generar un buscador que pueda lidiar con las necesidades de buscar dentro de los artículos de presidencia. Desde poder filtrar por texto a poder seleccionar en qué rango de fechas se encuentran los artículos, o buscar artículos por exclusión para poder refinar más la búsqueda.

La solución se enfocará en los problemas que le interesa solucionar a un periodista que utilice el buscador ya que inicialmente son los problemas más complejos y una vez que se tiene eso solucionado poder reconvertir la herramienta en algo que funcione para el uso del público general conlleva relativamente poco trabajo.

Diseño

Para el diseño de la aplicación se intentó minimizar la cantidad de overhead de los componentes y minimizar la complejidad para poder dedicar la mayor parte del esfuerzo al desarrollo de las funcionalidades.

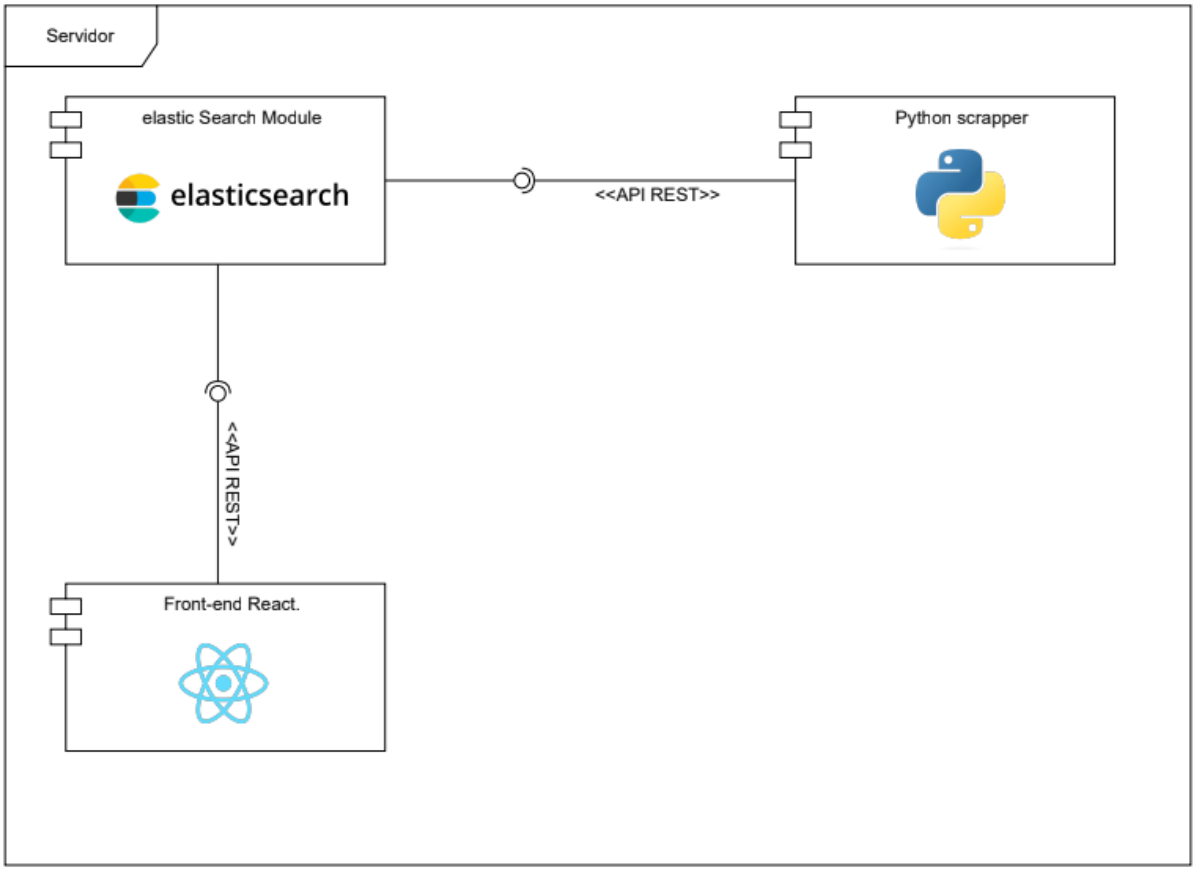


Diagrama de componentes

Python Scrapper

Un scrapper hecho en python con Beautiful Soup que recorre la web de presidencia y almacena en formato CSV los artículos, separado por las columnas de URL (la cual se utiliza como una suerte de clave), fecha de publicación del artículo, título del artículo, y cuerpo del artículo. Se eligió descartar el contenido multimedia de los artículos, usualmente imágenes o videos, para minimizar el coste de almacenamiento, por que no hay interés en utilizarlos para búsqueda y por que son fácilmente recuperables teniendo la URL al artículo si fuese necesario.

Se utilizó CSV para almacenarlos inicialmente porque la librería por defecto para manejo de CSVs de python permite escribir línea a línea lo que permite más fácilmente retomar el scrapeo en caso de fallo de conexión o algún otro problema inesperado.

Una vez se tienen los artículos en formato CSV se convierten al formato json que utiliza Elasticsearch y se cargan a través de la API de Elasticsearch.

Elasticsearch Module

Un contenedor de docker con una instancia de Elasticsearch corriendo que permite a la aplicación cliente realizar las búsquedas. Se alimenta del scraper y maneja tanto el almacenamiento e indizado del corpus como las búsquedas en el mismo. Se eligió por ser el método que requería menor overhead por parte del equipo, el servicio de Elasticsearch provee de todo lo necesario mientras que otras herramientas requieren mucho mayor trabajo de implementación y/o implementar un servidor o wrapper para poder comunicarse con ellas.

Front-End React

Una aplicación cliente construida con react.js que consume directamente las funcionalidades del componente con Elasticsearch a través de API REST. Se eligió react por su facilidad en cuanto a su velocidad de desarrollo además de proveer una forma simple de comunicarse con la API de Elasticsearch.

Funcionalidades y uso

Acceso a corpus de 1000 artículos

Se scrapeo e importó a Elasticsearch mil artículos con los cuales se realizaron todas las pruebas y no se presentó ningún problema en cuanto funcionamiento por tamaño del corpus.

Búsqueda por texto en título y cuerpo

Se utiliza la query “query_string” de Elasticsearch, que consiste en especificar los campos en los que se quiere buscar un determinado texto, además, permite realizar búsquedas booleanas en las que simplemente se escribe una operación booleana y se agregan los parámetros que se desean buscar.

Preprocesamiento

Previo a realizar las consultas a Elasticsearch, se realiza un preprocesamiento de la entrada. Se cuenta con una bolsa de stop words obtenida del sitio countswordsfree (https://countwordsfree.com/stopwords/spanish).

Cada palabra ingresada se anida como un OR a la palabra anterior. Por ejemplo si se ingresa “obras en hospitales” la búsqueda será “obras OR hospitales”. Se le asignará mejor puntuación al resultado de la búsqueda si cuenta con ambos términos.

Filtrado por rango de fechas

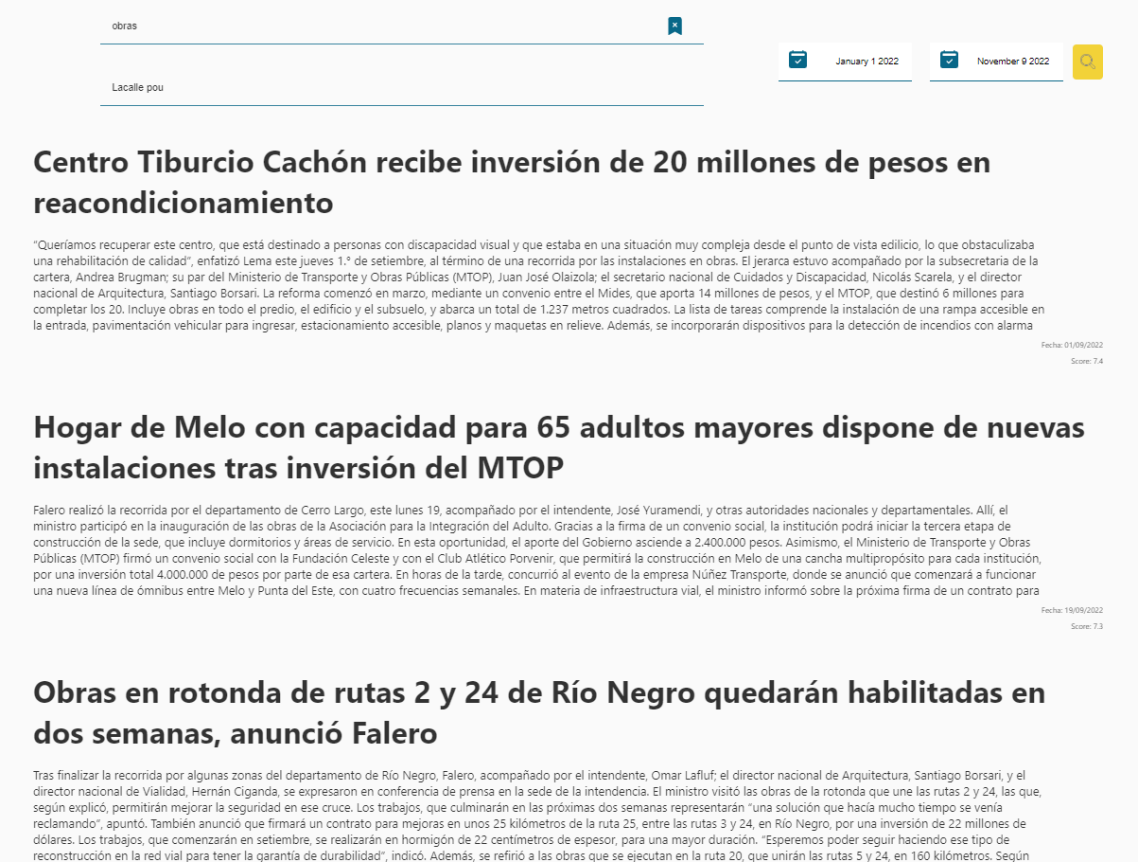
Una de las características de Elasticsearch es la manipulación de fechas, estas se filtran mediante operadores como lt (menor que) o gt (mayor que) permitiendo un filtrado sencillo de implementar. Se toman de dos campos del sitio las fechas para conformar el rango en el que se desea buscar las noticias.

Búsqueda por exclusión

Se implementó con la funcionalidad de búsqueda booleana de Elasticsearch, con el parámetro “must_not” que descarta los artículos que cumplan con una condición dada.

Por ejemplo:

Y luego quitando los que contengan el texto "Lacalle Pou" se observa el resultado:



Orden del resultado

El orden por defecto que se asigna al set resultado está dado por el score que brinda Elasticsearch para cada artículo encontrado, con la salvedad que se le da el doble de peso a la función de ranking utilizada si uno de los términos buscados está en el título del artículo.

Por defecto el score dado por Elasticsearch está basado en el algoritmo B25, sin entrar en mayor detalle ya que no es un algoritmo particularmente simple lo que se hace con el mismo es darle prioridad a los siguientes puntos:

- Frecuencia de los términos (TF): A más veces aparece un término buscado en el documento mayor importancia tiene el documento.
- Frecuencia inversa de los documentos (IDF): A más documentos contienen el término buscado menos importante es el término.
- Largo del campo: Si un documento contiene el término buscado y su cuerpo es pequeño es más relevante que si tiene el término y su cuerpo es grande.

Conclusiones

Se considera que el resultado del proyecto fue más que satisfactorio. Solo uno de los casos de uso planteados inicialmente no se implementó, el agregado automático de artículos al corpus, mayormente por que se determinó que no agrega demasiado al prototipo y sería algo que agregaría valor una vez que el proyecto esté funcionando, pero no en su estado inicial.

Se encontró en Elasticsearch una herramienta poderosa para implementar búsquedas que además combinándolo con React resulta en una infraestructura sólida y simple para llevar a cabo este proyecto y que puede ser útil para otros proyectos similares.

Trabajo Futuro

Se logró el desarrollo de muchas de las características deseadas pero quedan funcionalidades que serían interesantes de agregar, tanto algunas que se encontraban previamente como otras que surgieron durante el desarrollo.

Scrapping automático del corpus

Hay dos funcionalidades en cuanto a la adquisición del corpus que serían útiles de agregar. En este momento el corpus es estático y sin intervención manual no crece de ninguna manera, mientras que los artículos de presidencia si crecen, además que los artículos ya importados están lejos de ser todos los presentes en la web.

Primero, la importación automática de artículos nuevos al corpus. Desarrollar un proceso que chequee los nuevos artículos presentes en la web de presidencia y los agregue automáticamente al corpus, por el volumen de artículos esto sería suficiente correrlo una vez al día, o incluso una vez a la semana. La forma más simple sería agregar otro componente a la arquitectura que corra este proceso en los tiempos indicados, existe gran variedad de servicios que permiten realizar algo así por un costo bajo por lo que probablemente sería la forma más eficiente de solucionarlo.

Segundo, incorporar el resto de los artículos ya presentes en la página de presidencia al corpus. Desde el punto de vista de desarrolla es un costo similar al punto anterior con la salvedad de que desde el punto de vista de hardware o contratación de servicios esta parte si puede volverse costosa ya que aumentan rápidamente el tamaño del corpus demandando más recursos de Elasticsearch además que por el volumen de artículos también tiene un costo no ignorable el procesamiento inicial.

Categorización

Generar una categorización de los artículos en el corpus agrega valor al buscador. Poder filtrar entre artículos de promulgación de leyes, informes, artículos sobre un departamento específico o por cualquier otra de las categorías que puedan ser útiles sería una función útil en el buscador.

En cuanto a la parte técnica de esto lo primero sería determinar las categorías en las que interesa categorizar los artículos. Aunque hay algunas obvias puede ser de interés entrevistar usuarios (o potenciales usuarios) para tener un idea más concreta de qué categorías serían las útiles. Una vez se cuente con las categorías se puede determinar el mejor método para realizar el filtrado, algunas como las que traten sobre promulgación de leyes probablemente se puedan determinar de forma facil a traves de aplicar un regex mientras que otras pueden tomar mayor trabajo necesitando recurrir a algún tipo de aprendizaje automático para clasificarlas, este último método puede beneficiarse también de

tener el buscador ya funcionando con usuarios para contar con mayor información que pueda alimentar un algoritmo de ese tipo.

Estadísticas

Una vez se tiene todo el corpus incorporado en Elasticsearch se generan estadísticas del mismo que pueden ser útiles o de interés. Teniendo acceso a las búsquedas que realizan los usuarios se puede conseguir saber qué artículos son de más interés para el público, o que temas están llamando más la atención. ¿Qué tipos de leyes generan más interés en el público? ¿Qué tipo de noticias generan más tráfico? ¿Hay más interés en las noticias sobre la apertura de nuevos hospitales? ¿O sobre cambios en la reglamentación impositiva?

Que parte de las estadísticas serían más importantes o útiles no es obvio inicialmente, pero tener acceso a más información puede ser un valor agregado interesante del proyecto.