Sistema de Recuperación

de Información

Arnel Sánchez Rodríguez

Samuel Efraín Pupo Wong

Grupo C312

Universidad de la Habana

1. Introducción

La idea del uso de computadoras para la búsqueda de información se popularizó a raíz de un artículo *As We May Think* de [Vannevar Bush](https://es.wikipedia.org/wiki/Vannevar_Bush) en el año 1945. Los primeros sistemas automatizados de recuperación de la información fueron presentados durante la década de 1950 a 1960. En esta etapa se describe una máquina denominada UNIVAC capaz de buscar referencias de texto asociadas a un código temático. También se comenzó el uso de las listas de términos para indexar los objetos, el empleo de ranking en los documentos y la retroalimentación para la relevancia. Ya en este momento estaban consolidadas las computadoras como las herramientas definitivas para la búsqueda.

Durante 1970 se realizaron pruebas en un grupo de textos como la colección [*Cranfield*](https://es.wikipedia.org/wiki/Cranfield) para un gran número de distintas técnicas cuyo rendimiento fue bueno. Comenzaron a utilizarse los sistemas de recuperación a larga escala, como el Sistema de Diálogo *Lockheed*. Asimismo, se introdujo la idea de la frecuencia inversa en documentos

En la década de 1980, una preocupación de la comunidad académica era el tamaño de las colecciones documentos que se estaban empleando para las pruebas, pues se presumía eran muy pequeños respecto a los reales. Se creó una organización y anualmente se reunían para, cooperativamente, construir colecciones de prueba más grandes. El trabajo con estos nuevos conjuntos de datos mostró que las funciones existentes de ponderación y ranking no eran ideales. Además, se confirmó la idea de que diferentes colecciones requerirían diferentes acercamientos.

En 1992, el Departamento de Defensa de los Estados Unidos, en conjunto con el Instituto Nacional de Estándares y Tecnología (NIST por su sigla en inglés), patrocinaron la Conferencia de Recuperación de Texto (TREC) como parte del programa TIPSTER. Esto proveyó ayuda desde la comunidad de recuperación de la información al suministrar la infraestructura necesaria para la evaluación de metodologías de recuperación de texto en una colección a larga escala.

Desde los inicios de la *World Wide Web*, a finales de 1990, se hizo necesaria la adopción de motores de búsqueda que respondieran de manera eficiente a los requerimientos de los usuarios. Los desarrolladores rápidamente se dieron cuenta de que podían emplear los enlaces a los sitios web para construir robots que recorrieran y recopilaran la mayoría de las páginas en la Internet, automatizando la adquisición de contenidos.

Las aplicaciones de búsqueda y el campo de la recuperación de información continúan cambiando mientras que la computación evoluciona. Ejemplo de esto es el rápido crecimiento de los dispositivos móviles y las redes sociales. Nuevas investigaciones en una variedad de áreas como el etiquetado de usuarios, la recuperación de conversaciones, el filtrado y la recomendación comienzan a proporcionar herramientas efectivas para el manejo de la información personal y social.

Los [buscadores](https://es.wikipedia.org/wiki/Buscador), tales como [**Google**](https://es.wikipedia.org/wiki/Google), [**Yahoo**](https://protecciondatos-lopd.com/empresas/buscadores-de-internet/#Yahoo) **Search**, [**Bing**](https://protecciondatos-lopd.com/empresas/buscadores-de-internet/#Bing), [**Baidu**](https://protecciondatos-lopd.com/empresas/buscadores-de-internet/#Baidu) y [**DuckDuckGo**](https://protecciondatos-lopd.com/empresas/buscadores-de-internet/#DuckDuckGo), son algunas de las aplicaciones más populares para la recuperación de información.

1. Diseño del Sistema

Un Sistema de Recuperación de Información (IRS, por sus siglas en inglés) se compone por el cuádruplo , donde:

* es un conjunto de representaciones lógicas de los documentos que conforman un *corpus*.
* es un conjunto compuesto por representaciones lógicas de las consultas que el usuario realiza al sistema.
* es un *framework* para modelar las representaciones de los documentos, consultas y sus relaciones.
* es una función de *ranking* que asocia un número real con una consulta que pertenece a y una representación de documento que pertenece a . La evaluación de esta función establece un cierto orden de relevancia entre los documentos de acuerdo con la consulta.

El modelo clásico de búsqueda comprende, en primer lugar, al usuario con un requerimiento informacional, que se transforma en una consulta. Luego, esta es representada de una manera lógica y procesada en un motor de búsqueda que examina una colección de documentos, denominada *Corpus*. De aquí se seleccionan los documentos relevantes, pudiendo darles una ordenación, según su importancia, y se presentan al usuario. En algunos sistemas se realiza luego un proceso de retroalimentación en el que se trata de refinar la respuesta con el objetivo de que sea más relevante respecto a la necesidad del usuario.

Analizaremos a continuación nuestro sistema en su totalidad: las consultas, la composición del motor de búsqueda y la clasificación de los documentos según su relevancia. En nuestro caso, el sistema va enfocado a la búsqueda de documentos, los cuales tienen varias características, siendo las dos más útiles: el título y el cuerpo.

* 1. Motor de Búsqueda

El motor de búsqueda implementa un modelo de recuperación de información, en este se realiza el preprocesado de los documentos para su representación interna. También es el encargado de construir el ranking de relevancia de los documentos del *corpus*, acorde a la consulta de entrada.

**Preprocesamiento:** Para representar los documentos, es necesario realizar un preprocesado. En nuestro sistema empleamos recursos del procesamiento de lenguaje natural para modificar los documentos y luego poder vectorizar. Las siguientes transformaciones son realizadas para cada documento del *corpus*:

1. *Tokenización* del documento: Primeramente, se crea la representación básica de un documento: la división en tokens; de tal forma que podamos realizar procesamientos más avanzados sobre el texto. Desde esta etapa del preprocesamiento podemos llevar a minúscula todos los caracteres de las palabras, eliminar símbolos y signos de puntuación.
2. Eliminación de *stopwords*: Las *stopwords* son palabras vacías de significado, las cuales pueden servir de nexo entre entidades o funcionan como modificadores. Si incluimos estas palabras en la representación del documento se puede afectar precisión del modelo, debido a la alta frecuencia que poseen estas palabras en los textos.
3. *Stemming*: Este método busca relacionar palabras con igual significado pero que difieren en cuanto a la escritura debido a que presentan prefijos o sufijos. En este caso no usaremos *lemmatizing*.

Estas técnicas son las empleadas por el sistema para obtener un conjunto de textos formados por entidades con un significado y peso correcto para empezar a construir el modelo, pero antes de definir el modelo es necesario definir cómo serán estas entidades y qué representación interna tendrán. Empleando los tokens como entidad del sistema, se construye una colección con el siguiente formato:

corpus: {

[id: n,

'title': Tokenize(doc\_title),

'body': Tokenize(doc\_body),

],

...

}

**Vectorización:** Una vez que tenemos los tokens es necesario realizar una vectorización, obteniendo así una entrada correcta para el modelo, emplearemos para ello TF-IDF. Para calcular los valores de frecuencia se utilizarán el título y el cuerpo del documento juntos, sin realizar distinción, por lo que el título podrá ser una oración más del texto.

*Term Frequency–Inverse Document Frequency* (TF-IDF):Es una técnica para cuantificar términos en documentos, generalmente calculamos un peso para cada término, cuyo significado es la importancia de este en el documento y en el *corpus*. Su idea se basa en que la ocurrencia de un término en una colección de documentos es inversamente proporcional a su importancia en la recuperación. Dicho método es ampliamente utilizado en recuperación de información y minería de textos. Se representa por , donde es un término del documento .

La definición de TF-IDF se compone de dos conceptos fundamentales en el procesamiento de los documentos:

TF (Frecuencia de Término): Mide la frecuencia de una palabra en un documento. Esto depende en gran medida de la longitud de dicho documento y de la generalidad de esta palabra. Debido la variación de la longitud, contar simplemente las palabras podría dar prioridad a documentos de mayor tamaño, por esta razón se divide la cantidad de ocurrencias por el número total de palabras. TF es individual para cada documento y palabra, por lo tanto, podemos formularlo de la siguiente manera:

Donde representar la cantidad de ocurrencias del término en el documento. Cada componente tendrá el valor de frecuencia asociado para esa palabra, el cual estará en el intervalo . Sin embargo, esta ponderación beneficia a palabras comunes que se repiten en los documentos.

DF (Frecuencia de Documento): Mide la importancia del término en todo el conjunto de *corpus*. Es el recuento de apariciones de este en el conjunto de documentos , es decir, el número de documentos en los que está presente la palabra. Se considera una ocurrencia si el término consta en el documento al menos una vez, ignorando el número de veces que está presente.

IDF (Frecuencia Inversa de Documento): Es la inversa de DF, que mide la informatividad del término . Este será muy bajo para las palabras más frecuentes y mayor para las menos comunes. Como es posible que la DF sea , entonces es necesario realizar algunas modificaciones en la división de la inversa. Además, si el *corpus* es grande se podría obtener un valor alto, que no es de utilidad, por lo que se empleará una función de crecimiento más lento para suavizar la IDF.

Combinando ambas ecuaciones (TF-IDF):

Anteriormente se expuso que no existe distinción entre el título y el cuerpo del documento, puesto que el primero se analiza como una oración más del texto. Por tanto, no hay diferencias en el cálculo de la ponderación de los términos de ambos.

Para vectorizar el documento emplearemos el siguiente concepto:

*Bag of Words*: Consiste en tomar todas las palabras del *corpus* que constituyen el vocabulario y formar una colección donde cada palabra tiene asociado un índice, de esta forma se obtiene un vector de dimensión asociado al documento , donde en cada componente se encuentra el valor de TF-IDF ligado a la palabra.

*El Bag of Words* puede considerarse como un *mapping* donde podemos comprobar en que posición de una lista formada por todas las palabras se encuentra la que estamos buscando, empleando esta idea es posible vectorizar cada documento.

* 1. Consultas

Se conoce que en el sistema existe un conjunto , compuesto por las consultas realizadas por el usuario. Una consulta no es más que una oración o pequeño párrafo empleado para realizar una búsqueda. Para poder determinar cuáles documentos son relevantes para esa consulta, es necesario establecer un orden de estos, acorde a la similitud que exista entre ellos. Por lo tanto, debemos convertir la consulta a un vector empleando el mismo proceso descrito en el motor de búsqueda.

Se usa el mismo vocabulario utilizado para la representación del *corpus*, y se procede a crear el vector *,* en caso de que existan palabras en que no se encuentren en el vocabulario, serán ignoradas. El valor que tomarán las componentes no tiene necesariamente que ser calculado igual a como se realiza en el motor de búsqueda para representar el *corpus*, depende mayormente de la función de ranking que empleamos. En nuestro caso tomamos dos representaciones vectoriales acorde a como se calculaba el ranking.

Para crear el vector de consulta usando TF-IDF se agrega un parámetro de suavizado , el cual permite amortiguar la frecuencia, evitando que aparezcan grandes saltos en esta o se repita el mismo valor de forma consecutiva. El cálculo de los pesos para los términos de la consulta sería del modo:

* 1. Función de *Ranking*

Ahora es necesario relacionar la consulta con el *corpus*, para determinar el valor de relevancia que tienen los documentos con respecto a esta. Para ello debemos definir la función de la cual expondremos a continuación dos enfoques:

**Puntuación Coincidente:** Una forma sencilla de calcular la similitud es representar el vector de consulta de forma *booleana*, asignando para los términos que aparecen en la consulta y en caso contrario. Luego, para cada documento sumamos los valores de TF-IDF que tienen en el vector consulta, obteniéndose de esta forma el valor de similitud para realizar la ordenación. Esta operación solamente realiza un producto escalar de los vectores.

De esta forma, obtenemos mayores valores de relevancia para documentos con valores de frecuencia altos para los términos que componen la consulta. Al representar el vector consulta solamente con 0 y 1, no se aprovecha la relación de cercanía espacial que podría existir entre los vectores de consulta y documento.

**Similitud del Coseno:** Se utilizan el valor de TF-IDF asociado al término en la consulta y el coseno del ángulo comprendido entre los vectores consulta y documento para determinar la similitud:

Podemos apreciar cómo se busca establecer una relación entre los valores de frecuencia de la consulta y el documento. De esta forma, mientras más cerca estén los vectores de consulta y documento, más relevante es ese documento para el usuario que realiza la petición.

***Output* del Sistema:** Una vez se tiene la función de *ranking*, se calculan los valores correspondientes para la consulta, ordenando de forma descendente aquellos que son distintos de 0. Se devuelve el título del documento y el valor obtenido.

* 1. Retroalimentación de Relevancia

La retroalimentación de relevancia es una característica de algunos sistemas de recuperación de información. La idea consiste tomar los resultados que se devuelven inicialmente de una consulta determinada, recopilar la retroalimentación de los usuarios y utilizar información sobre

si esos resultados son relevantes o no para realizar una nueva consulta. Podemos

distinguir de manera \_util entre tres tipos de retroalimentaci\_on: retroalimentaci\_on

expl\_\_cita, retroalimentaci\_on impl\_\_cita y retroalimentaci\_on ciega o "pseudo".

Debido a que emplear el TF-IDF no siempre resulta en una representaci\_on

con\_able ya que no se toma en cuenta la posici\_on de los t\_erminos ni las relaciones

que se establecen entre estos, por ello decidimos incluir un algoritmo de

retroalimentaci\_on expl\_\_cito, con el objetivo de suministrar al usuario una manera

de mejorar el vector consulta y ayudar a mejorar el recobrado del sistema,

as\_\_ podemos resolver problemas de ambiguedad en las consulta. Seleccionamos

el algoritmo de Rocchio el cual es abordado en [2], este es simple y f\_acil de

implementar.

Algoritmo de Rocchio: Es un algoritmo muy conocido para la retroalimentaci\_on

en los sistemas de recuperaci\_on de informaci\_on. Era empleado en los IRS que

surgieron del Sistema de Recuperaci\_on de Informaci\_on SMART, desarrollado en

1960-1964. El enfoque de retroalimentaci\_on de Rocchio se desarroll\_o utilizando

el Modelo de Espacio Vectorial que empleaba SMART. El algoritmo se basa en

la suposici\_on de que la mayor\_\_a de los usuarios tienen una concepci\_on general de

qu\_e documentos deben indicarse como relevantes o no relevantes para la consulta

que realizan. Por lo tanto, la consulta de b\_usqueda del usuario se revisa para

incluir un porcentaje arbitrario de documentos relevantes y no relevantes como

un medio para aumentar la precisi\_on del motor de b\_usqueda.

El algoritmo de Rocchio busca realizar una mejora a la consulta que deseamos,

podemos de\_nir que una consulta 􀀀!q

i realiza una bipartici\_on del conjunto

D en dos subconjuntos, El subconjunto DR donde se encuentran aquellos

documentos que fueron considerados relevantes por el sistema y D 􀀀 DR ser\_an

aquellos que no fueron marcados como relevantes. Ahora debemos encontrar una

consulta \_optima a partir de 􀀀!q

i para la cual se maximice la similitud con el conjunto

de documentos relevantes DR implicando un valor m\_\_nimo de similitud

para el conjunto de documentos no relevantes:

1. Detalles de la Implementación

La implementaci\_on del sistema se realiz\_o en Python, este cuenta con muchas

herramientas para el procesamiento de lenguaje natural, permite representar

Sistema de Recuperaci\_on de Informaci\_on 9

y manejar vectores de forma m\_as simple a otros lenguajes y al ser tan empleado

para realizar trabajos de manejo de datos, recuperaci\_on de informaci\_on

e inteligencia arti\_cial contamos con una extensa documentaci\_on y ayuda para

diversos problemas.

El objeto que representa el IRS se de\_ne por IrSystem, el cual contiene las

implementaciones de los temas abordados en la secci\_on anterior. Para crear una

instancia se emplea el valor de \_ para de\_nir el peso que se le da al t\_\_tulo de

los documentos, y la direcci\_on del dataset que ser\_a empleado, el cual contiene

el corpus. Si nuestro corpus no contiene textos con t\_\_tulo solamente tendremos

que hacer \_ = 0, para que no se tome en cuenta el peso del t\_\_tulo.

1. Evaluación del Sistema
2. Resultados Obtenidos
3. Recomendaciones
4. Referencias

Interfaz visual

Para hacer más cómoda la realización de consultas, se implemento una interfaz visual utilizando la librería React de JavaScript, tomando una plantilla de Google como base.

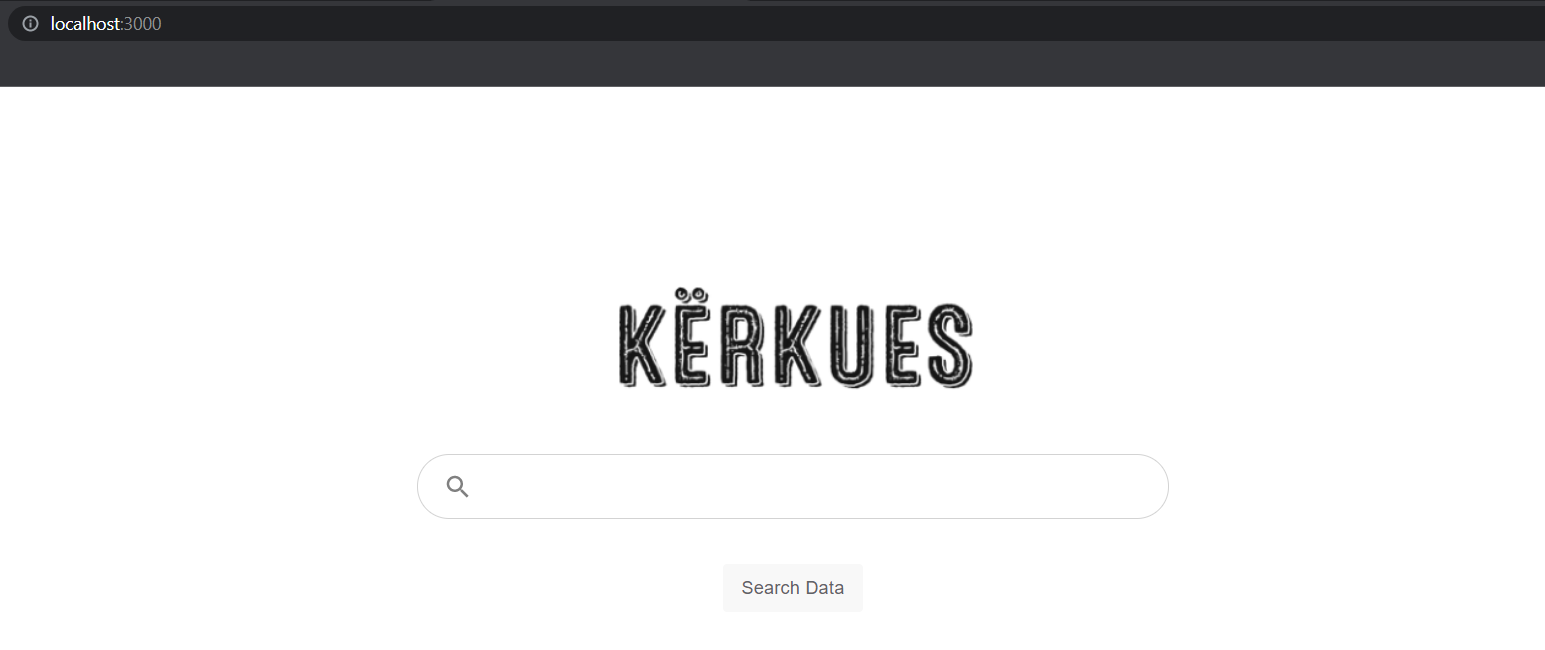


Imagen 1. Pantalla principal

La interfaz es muy intuitiva, puesto que cuenta con una pantalla principal con un campo de texto donde se introduce la búsqueda deseada y un botón para iniciarla.

Luego, los resultados de la búsqueda son mostrados en una nueva pantalla donde estos se listan, de manera tal que se pueden visualizar los títulos de los documentos obtenidos y un link que permite abrirlos en caso de desearse.

En esta pantalla también se cuenta con un campo de texto que permite ejecutar una nueva búsqueda y un botón que retorna a la pantalla principal.

Al ejecutar cada búsqueda, se observa el total de resultados obtenidos y el tiempo en que se obtuvo respuesta. Del mismo modo, se muestra una lista de posibles sugerencias afines con los términos empleados.

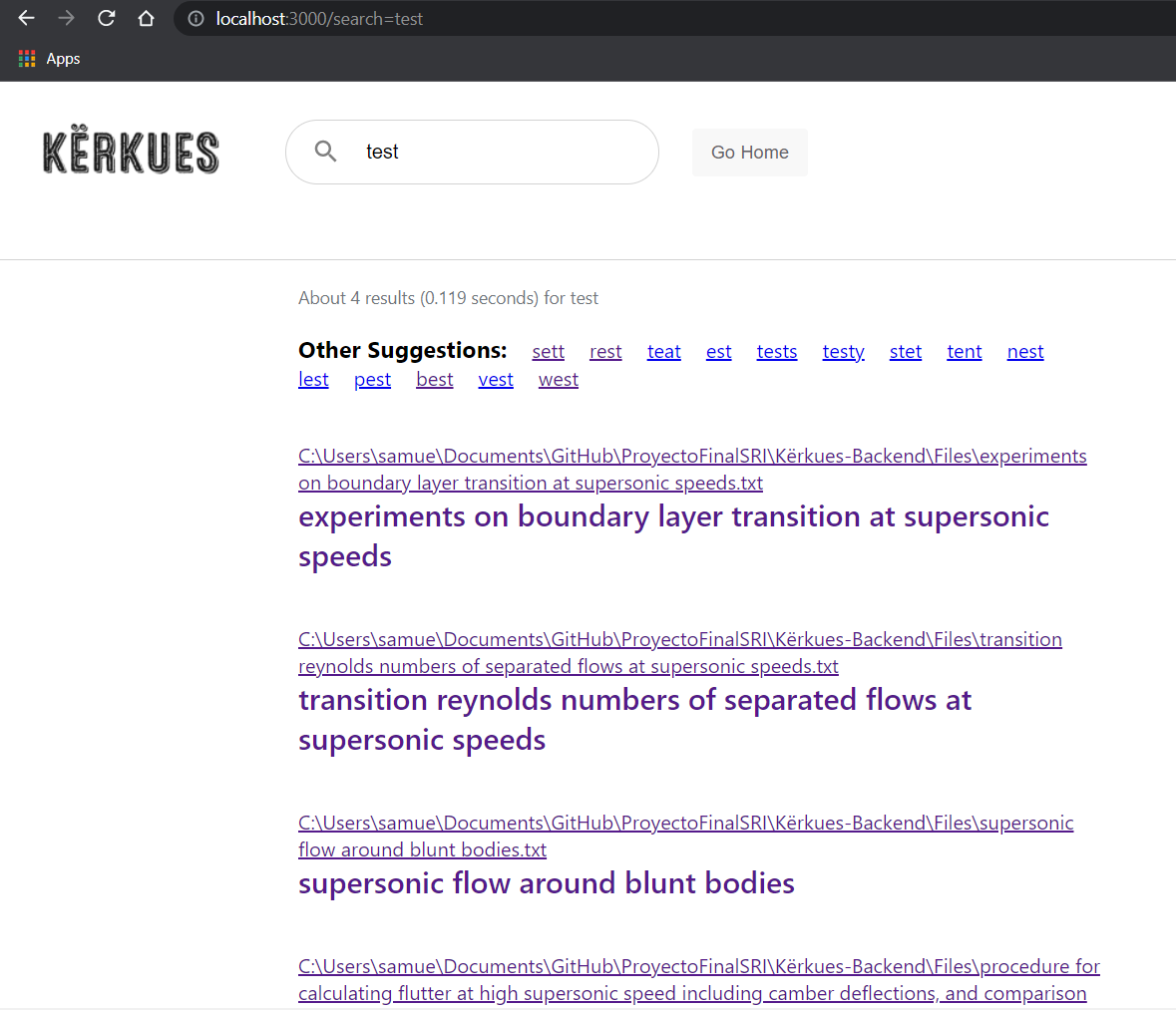


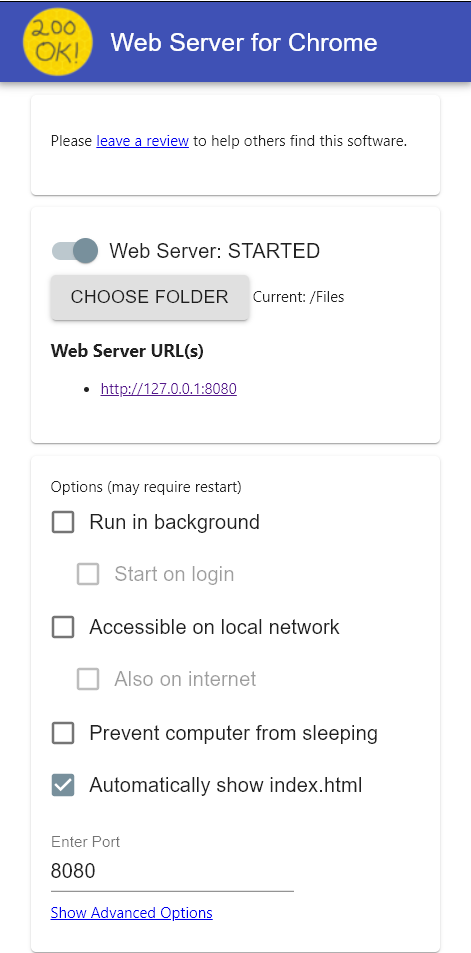
Imagen 2. Pantalla de búsqueda

La dirección inicial del navegador (pantalla principal) es: [*http://localhost:3000*](http://localhost:3000)

Las direcciones de búsqueda son: [*http://localhost:3000/search=termino*](http://localhost:3000/search=termino)

Para el correcto funcionamiento de la interfaz visual, el **Backend** del proyecto debe estarse ejecutando.

Dado que los navegadores no permiten abrir archivos locales de manera automática (por cuestiones de seguridad), es necesario crear un servidor local donde estos sean accesibles. Se propone utilizar el **Add** de **Google Chrome**: **Web Server for Chrome**.



En el botón **Choose Folder** debe seleccionarse la carpeta en la cual están los archivos utilizados en el **Backend**, y el puerto a utilizar debe ser el **8080**.

La URL del servidor debe ser: [*http://127.0.0.1:8080*](http://127.0.0.1:8080)