Findtastic

https://github.com/MaferGlez03/SRI-Project-2024

Daniel Polanco Pérez
@Daniman30
regulardani1@gmail.com

María Fernanda Suárez González
@MaferGlez03
mariafernandasuarezglez@gmail.com

Adrián Navarro Foya
@adrianfoya
adriannf0108@gmail.com

Introducción

En la era digital actual, los sistemas de búsqueda juegan un papel crucial al ofrecer a los usuarios contenido y recomendaciones adaptadas a sus preferencias individuales. Sin embargo, a pesar de su precisión, estos sistemas carecen de transparencia, lo que genera desconfianza y falta de control por parte de los usuarios. Este informe aborda la necesidad de desarrollar un motor de búsqueda que permita a los usuarios consultas personalizadas y una trazabilidad detallada para entender cómo se toman las decisiones; y selecciones personalizadas para incidir en la necesidad de transparencia en este tipo de algoritmos.

¿Qué es la transparencia de los algoritmos?

La transparencia de los algoritmos se refiere a la capacidad de comprender y analizar los mecanismos de toma de decisiones de los sistemas algorítmicos. Esto implica a menudo poder examinar el código fuente, los datos operativos y los criterios de decisión de los algoritmos.

Hay varias razones por las que los algoritmos deben ser transparentes. En primer lugar, hay una búsqueda de justicia e imparcialidad: las decisiones basadas en datos erróneos o sesgados pueden perjudicar a personas o grupos concretos.

Eli Pariser (2017) ha denominado la “burbuja de filtros” al efecto por el que un algoritmo selecciona las informaciones que prefiere el usuario, en función de las búsquedas anteriores, la geolocalización, o las preferencias en cualquier tipo de selecciones. La burbuja presenta una web empequeñecida con los mismos contenidos que coinciden con los propios puntos de vista, reforzando una ideología previa y al margen de un pensamiento crítico.

En el procesamiento de grandes datos se dan estereotipos muy significativos en los que basa su aprendizaje la IA. Aún lo es más si se tiene en cuenta que la ideología de las personas que deciden el problema a resolver con el algoritmo, sus objetivos y requisitos, también genera sesgos en el propio proceso de formulación.

Epstein y Robertson (2015) defienden que Google tiene posibilidad de cambiar el resultado del 25 % de las elecciones nacionales de diferentes países del mundo sin que nadie advierta si se produce una manipulación en los resultados.

Así llegaron a la definición del Efecto Manipulador del Motor de Búsqueda o Search Engine Manipulation Effect (SEME) (Epstein y Robertson, 2015) sobre el que aseguran que es uno de los mayores efectos sobre el comportamiento humano que se haya sistematizado.

Descripción del tema

Se hace necesario un motor de búsqueda que brinde la transparencia necesaria para evitar caer en el sesgo generado por los algoritmos actuales y por la infraestructura de la red. Por ello se presenta la opción de “Findtastic” que utiliza herramientas similares a las de los algoritmos predominantes, pero con la característica de permitir al usuario la personalización total y completa de estas herramientas en cada una de sus búsquedas.

Antecedentes de la temática seleccionada

Otras iniciativas tratan de atajar el problema de los sesgos. El proyecto europeo FA*IR, con participación de la Universidad Pompeu Fabra y Eurecat, ha investigado la discriminación según género, procedencia o apariencia que lleva a invisibilizar a determinados colectivos en la búsqueda de empleo. El equipo de investigación ya ha desarrollado un algoritmo de búsqueda que invalida el sesgo étnico, de género o de edad, sin afectar al ranking obtenido.

Otras alternativas se han centrado en indicadores que cuantifican y definen los sesgos, directos o indirectos, cuando se insertan en el software de un buscador o en otras utilidades basadas en el Big Data, con un modelo en la práctica (Bolukbasi et al., 2016) que se puede utilizar sin modificar las propiedades útiles del sistema.

Explicación de la solución implementada

La implementación de "Findtastic" involucra varios componentes clave que trabajan juntos para proporcionar una experiencia de búsqueda personalizada y transparente. En el lado del servidor, se utiliza Python para procesar la solicitud y realizar la búsqueda. La función search en el backend recibe todos los parámetros necesarios, incluyendo los estados booleanos de las opciones de configuración.

El código presentado implementa una función de búsqueda que utiliza el modelo de similitud de coseno y la técnica de TF-IDF (Term Frequency-Inverse Document Frequency) para encontrar y ordenar documentos relevantes en función de una consulta dada. A continuación, se describe el funcionamiento del código paso a paso.

La función search toma los siguientes parámetros:

- query: La consulta de búsqueda.
- verbose: Un parámetro booleano para controlar la verbosidad de la salida.
- values: Valores de relevancia asignados a cada token por el usuario.
- query_expand_bool, stop_words_bool, stemmer_bool: Parámetros booleanos para controlar el procesamiento de la consulta.

processed_query = process_query(query, query_expand_bool, stop_words_bool, stemmer_bool)

El método process_query se encarga de procesar una consulta de búsqueda aplicando varias técnicas de procesamiento de lenguaje natural (NLP) como la tokenización, normalización, eliminación de palabras vacías, expansión de consulta y stemming. A continuación, se describe cada paso del método en detalle:

Para la tokenización se divide la consulta en palabras individuales (tokens), en la normalización se convierten todos los tokens a minúsculas para asegurar la consistencia. Luego se Define un conjunto de palabras comunes en español que no aportan mucho significado (como “el”, “la”, “de”), las cuales son conocidas como stopwords.

En la siguiente parte se definen la expansión de consulta, que consiste en buscar sinónimos a cada palabra de la consulta. Luego se utiliza un stemmer (Porter Stemmer) para reducir las palabras restantes a su raíz y por ultimo se utilizan las stopwords para eliminar las palabras que no aportan mucho a la consulta. Cada una de estas funciones se proceden en función de la personalización de la consulta que haya hecho el usuario mediante los booleanos.

Para cada una de estas funciones se aprovecha la biblioteca The Natural Language Toolkit (NLTK) la cual es una biblioteca de código abierto que implementa todas estas funciones.

documents = read_documents_from_folder()

El método read_documents_from_folder se encarga de leer todos los archivos de texto (.txt) desde una carpeta específica y devolver su contenido en un diccionario. Esto para formar el conjunto de datos definido con el que se trabaja.

matrix, final_values = coseno_similitud(texts, query, verbose, values)

El método coseno_similitud es el que finalmente calcula la similitud de coseno entre los textos y la consulta definida, utilizando la representación TF-IDF, como ya se mencionó anteriormente.

Se crea una instancia de TfidfVectorizer con unos parámetros definidos que convierten todas las palabras a minúsculas, no eliminan palabras vacías e incluyen todas las palabras independientemente de su frecuencia.

Se ajusta el vectorizador a los textos y se transforman en una matriz de vectores TF-IDF. La matriz se convierte en un array de NumPy. Y se obtiene el vocabulario del vectorizador, que es un diccionario que mapea cada palabra a su índice correspondiente en los vectores TF-IDF.

En este punto, si el usuario activo la opción de personalización de los valores de la consulta se le asignan, en caso contrario se mantienen con los valores calculados por el TfidfVectorizer. Posteriormente se calcula la similitud del coseno utilizando la función cosine_similarity. Esta función y el TfidfVectorizer provienen de la biblioteca sklearn.

Finalmente se ordenan y limitan los resultados obtenidos para crear el ranking final y este es devuelto para ser impreso en el frontend.

En el lado del cliente, se utiliza JavaScript para gestionar los eventos del formulario de búsqueda y la interacción con los elementos DOM. El código JavaScript implementa eventos de cambio para los checkboxes de configuración, permitiendo que los usuarios modifiquen dinámicamente los criterios de búsqueda en tiempo real.

El formulario de búsqueda se envía utilizando la API Fetch, lo que permite una comunicación asíncrona con el servidor sin bloquear la interfaz del usuario. El método POST se utiliza para enviar los datos de búsqueda, incluyendo los estados de los checkboxes y los valores de los sliders.

Finalmente, para el trabajo del frontend se utilizan HTML y CSS para representar la barra de búsqueda, el menú de personalización y los resultados de cada consulta.

La implementación de “Findtastic” destaca por su enfoque sólido en la transparencia y personalización dentro de los motores de búsqueda. Proporciona una interfaz simple pero intuitiva, permitiendo a los usuarios interactuar directamente con los criterios de decisión del algoritmo. Este avance es significativo en términos de transparencia, ya que brinda a los usuarios una mayor comprensión y control sobre cómo se generan los resultados de búsqueda.

Además, la flexibilidad en la configuración de procesos, como la expansión de consultas y la aplicación de stemming, demuestra una comprensión profunda de cómo estos ajustes pueden influir en el proceso de búsqueda, permitiendo a los usuarios afinar los resultados según sus necesidades específicas. Integrando además tecnologías avanzadas de procesamiento de lenguaje natural (NLP).

Sin embargo, hay áreas que podrían beneficiarse de mejoras sustanciales. Sería clave proporcionar más opciones de personalización de la consulta, la cual estaría relacionada con metadatos tanto del usuario como de los documentos provistos.

Es claro que la base de datos establecida para el trabajo con el algoritmo es pobre y su crecimiento propondría cuestiones que no fueron abordadas en este trabajo como son el rendimiento del algoritmo. Relacionada con esto es digno mencionar la posibilidad de incluir una cache para la agilización de las búsquedas, estableciendo en ella todos los resultados TF-IDF de los documentos y solo siendo necesario el cálculo para la consulta.

En resumen, aunque “Findtastic” ya muestra un enfoque robusto y avanzado, la incorporación de estas mejoras podría fortalecer aún más su posición como una herramienta de búsqueda transparente y personalizada.

Conclusión

La transparencia de los algoritmos de consultas y búsqueda sigue siendo un reto contemporáneo crucial, el control de las grandes corporaciones presenta la imposibilidad del código abierto o del control total del uso de los datos. Por ello este proyecto no solo aborda una necesidad técnica, sino también una demanda creciente de transparencia y control por parte de los usuarios en el entorno digital.

Referencias:

Pariser, Eli (2017): El filtro burbuja. Madrid/Barcelona: Taurus
https://www.elboomeran.com/upload/ficheros/obras/documentofiltro.pdf

Epstein, Robert and Robertson, Ronald (2015): “The search engine manipulation effect (SEME) and its possible impact on the outcomes of elections”. PNAS, 112(33), E4512- E4521. Doi:
doi.org/10.1073/pnas.1419828112

LegalProd. (2024, mayo 17). *Transparencia del algoritmo*. LegalProd.
https://www.legalprod.com/es/transparencia-del-algoritmo/

Benítez Eyzaguirre, L. (2019). Ética y transparencia para la detección de sesgos algorítmicos de género. *Estudios sobre el Mensaje Periodístico, 25*(3).
https://doi.org/10.5209/esmp.66989

Correa Gorospe, J. M., Losada Iglesias, D., & Gutiérrez-Cabello Barragán, A. (2021). Big Data y la alfabetización posthumana del futuro profesorado. *Sociología y tecnociencia: Revista digital de sociología del sistema tecnocientífico, 11*(Extra 2), 102-122.
https://dialnet.unirioja.es/servlet/articulo?codigo=8155417

Herce Maza, J. I. (2022). Buena administración de la transparencia de los algoritmos de la Administración Pública: Un instrumento para el control de las cajas negras decisiones. *Capítulos de Libros*.
https://burjcdigital.urjc.es/handle/10115/30395