RENDIMIENTO DE LA BÚSQUEDA DE PALABRAS MEDIANTE 3 ALGORITMOS DE BÚSQUEDA DISTINTOS DE PYTHON.

VLADIMIR OSVALDO CURIEL, ARMANDO JOSÉ GONZÁLEZ LÓPEZ
VOCO0001@CE.PUCMM.EDU.DO, AJGL0001@CE.PUCMM.EDU.DO

POSTER DE INVESTIGACIÓN

INTRODUCCIÓN

El propósito de esta investigación es evaluar y determinar cuál de tres algoritmos de búsqueda de palabras entre, findall, finditer y search usado en bucle, es el más eficiente en términos de tiempo al realizar búsquedas de palabras en grandes volúmenes de datos. Este análisis permitirá identificar el algoritmo que ofrece los mejores resultados de rapidez en el procesamiento, lo cual es crucial para manejar eficientemente grandes conjuntos de datos. Aunque las diferencias de tiempo puedan ser menos significativas con volúmenes pequeños, conocer el algoritmo más eficiente para grandes volúmenes puede mejorar el rendimiento general.

OBJETIVOS

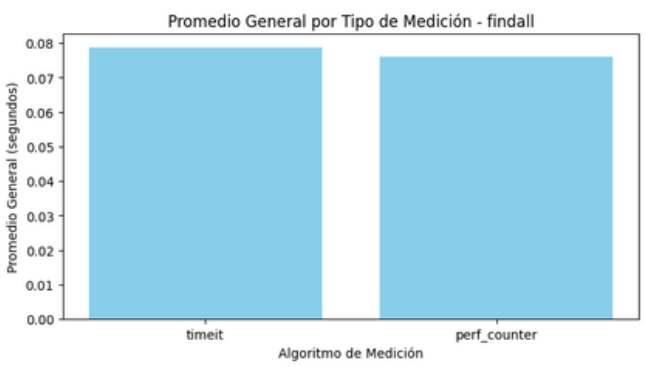
- Comparar el tiempo de ejecución de cada uno de los algoritmos y el lenguaje de programación Python con la librería **re**.
- Evaluar el rendimiento de los algoritmos de búsqueda de palabras **finditer**, **findall** y **search** (en bucle).
- Determinar cuál de los algoritmos muestra el mejor rendimiento en términos de rapidez al analizar textos, para identificar el más eficiente en la búsqueda de palabras.

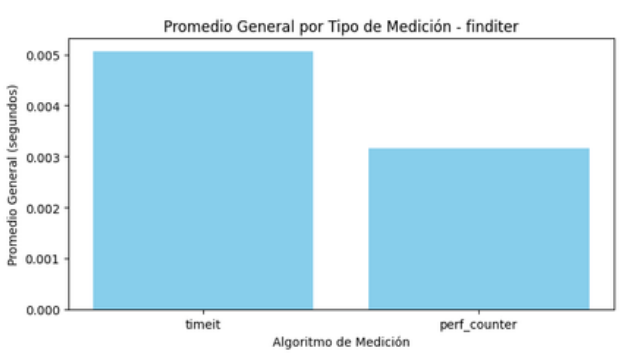
METODOLOGÍA

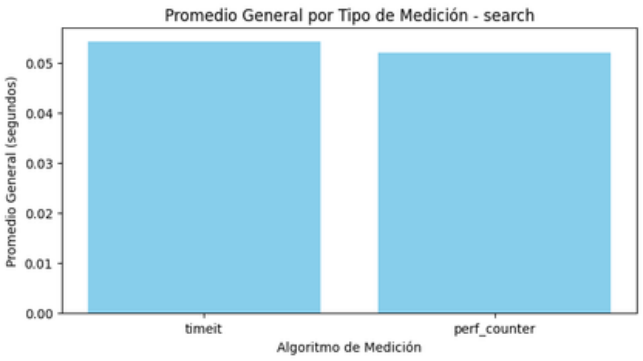
La población para esta investigación está constituida por textos arbitrarios en el idioma inglés del área de Ciencias en Computación. Asimismo, esta mantiene un enfoque cuantitativo el cual se basa en analizar datos numéricos y estadísticas relacionadas con los textos seleccionados, como la distribución de términos en los abstracts. Por otro lado, el procedimiento de evaluación implica seleccionar textos arbitrados en inglés y clasificar los abstracts según las categorías de editoriales, revisiones y capítulos de libros, como también, analizar el vocabulario y las palabras claves que contienen cada una.

RESULTADOS

Los resultados se presentan en gráficos que muestran los tiempos promedio y sus distribuciones normales, permitiendo una visualización clara de la eficiencia de cada algoritmo. A pesar de la variabilidad en los datos, todos los resultados indican que el algoritmo más rápido es **finditer**, con una eficiencia del 93.51% en comparación con los otros algoritmos. Los gráficos destacan que **finditer** ofrece el mejor rendimiento, con diferencias notables entre los tres algoritmos, donde una menor altura en los gráficos indica un mejor resultado.

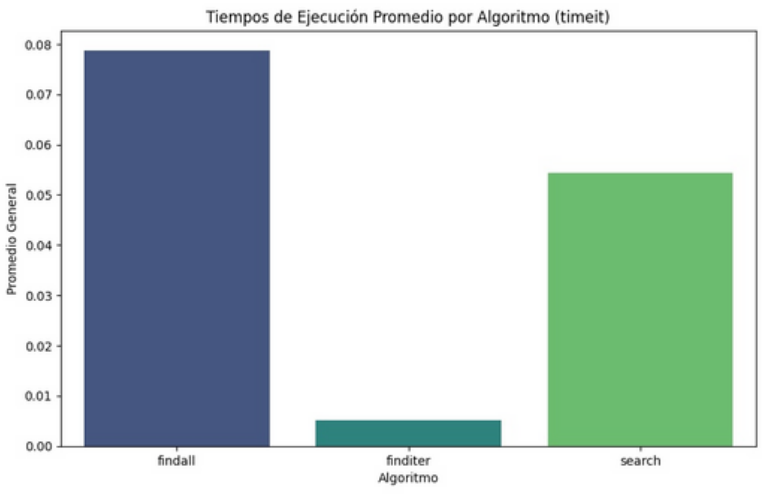


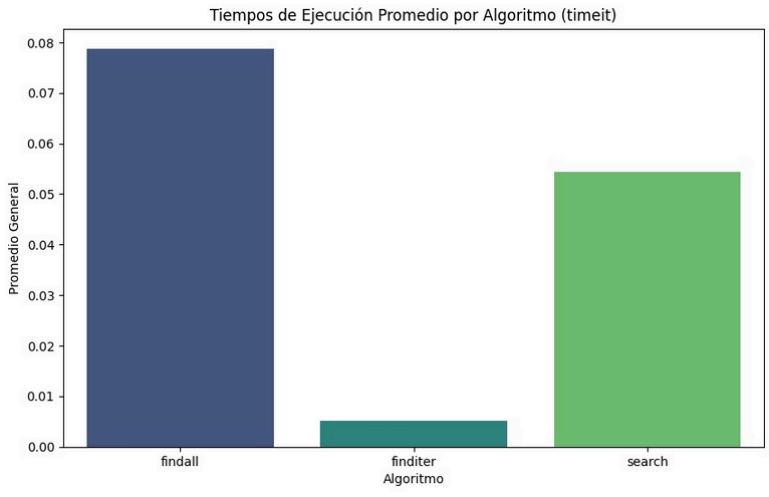




CONCLUSIÓN

En este estudio se compararon tres métodos de búsqueda de palabras de la librería **re** de Python: **findall**, **search** y **finditer**, evaluando su rendimiento en términos de tiempo. Utilizando un enfoque cuantitativo y un muestreo estratificado, se analizaron 374 resúmenes de Scopus para identificar las 10 palabras más frecuentes, midiendo el tiempo de ejecución con **timeit** y **perf_counter** en 10,000 repeticiones, sumando un total de 224,400,000 iteraciones. Los resultados muestran que **finditer** es el algoritmo más eficiente, con una diferencia del 93.51% en eficiencia en comparación con los otros métodos. Esto subraya la importancia de elegir el algoritmo más rápido, especialmente en aplicaciones que manejan grandes volúmenes de datos, donde **finditer** se destaca por su alta eficiencia, mientras que **findall** y **search**, aunque funcionales, presentan tiempos de ejecución significativamente mayores.





REFERENCIAS BIBLIOGRÁFICAS

Vásquez, Augusto Cortez, et al. “Procesamiento de Lenguaje Natural.” Revista de Investigación de Sistemas e Informática, vol. 6, no. 2, 2009, pp. 45–54. https://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923/5121

Martínez, B. Beltrán. “Minería de Datos.” Cómo Hallar Una Aguja En Un Pajar. Ingenierías, vol. 14, no. 53, 2001, pp. 53–66. https://www.cs.buap.mx/~bbeltran/NotasMD.pdf

Farías, Gustavo. Expresiones Regulares En GNU/Linux. 2023. https://repositorio.cfe.edu.uy/handle/123456789/228

re Match — Regular expression operations. (n.d.). Python Documentation. https://docs.python.org/3/librar re.html#re.Match

re Search — Regular expression operations. (n.d.). Python Documentation. https://docs.python.org/3/library re.html#re.Search

re Findall — Regular expression operations. (n.d.). Python Documentation. https://docs.python.org/3/library/ re.html#re.finding-all-adverbs

re Finditer — Regular expression operations. (n.d.). Python Documentation. https://docs.python.org/3/library/ re.html#re.finding-all-adverbs-and-their-positions

Muestreo estratificado, Walpole, R. E. (2012). PROBABILIDAD y ESTADISTICA PARA INGENIEROS y CIENCIAS, 9ED. (p.8)

Elsevier. (n.d.). What is Scopus Preview? - Scopus: Access and use Support Center. https://service.elsevier.com/app/answers/ detail/a_id/15534/supporthub/scopus/#tips

time – perf_counter() — Time access and conversions. (n.d.). Python Documentation. https://docs.python.org/3/library/ time.html#time.perf_counter

timeit — Measure execution time of small code snippets. (n.d.). Python Documentation. https://docs.python.org/es/3/ library/timeit.html#timeit.timeit

Muestreo estratificado, Walpole, R. E. (2012). PROBABILIDAD y ESTADISTICA PARA INGENIEROS y CIENCIAS, 9ED. (p.172)