

# RAIn 2025

## Trabajo Práctico N° 1

### Tema: Introducción

**Fecha Inicio:** 01/04/2025 **Fecha de Entrega:** 15/04/2025

**Autor:** Jorge Justino Riera **LU:** 5575 **Carrera:** Ing. Inf **Plan:** 2010

**Repositorio código fuente:** <https://github.com/nicrom8b/rain/tree/main/tp1>

### Indice

|  |          |
|--|----------|
| <b>Tema: Introducción</b>  | <b>1</b> |
| Fecha Inicio: 01/04/2025 Fecha de Entrega: 15/04/2025  | 1        |
| Autor: Jorge Justino Riera LU: 5575 Carrera: Ing. Inf Plan: 2010   | 1        |
| Repositorio código fuente: <a href="https://github.com/nicrom8b/rain/tree/main/tp1">https://github.com/nicrom8b/rain/tree/main/tp1</a> | 1        |
| <b>Actividades</b>   | <b>1</b> |
| Fichero invertido con frecuencias  | 8        |
| Objetivo del Proyecto:   | 8        |
| Funciones Principales:   | 8        |
| Fichero invertido posicional   | 9        |
| Objetivo del Proyecto:   | 9        |
| Funciones Principales:   | 9        |
| Consultas y resultados   | 11       |
| Reconstrucción de texto  | 13       |
| Objetivos del proyecto:  | 13       |
| Funciones Principales:   | 13       |
| Ejecución  | 14       |

# Actividades

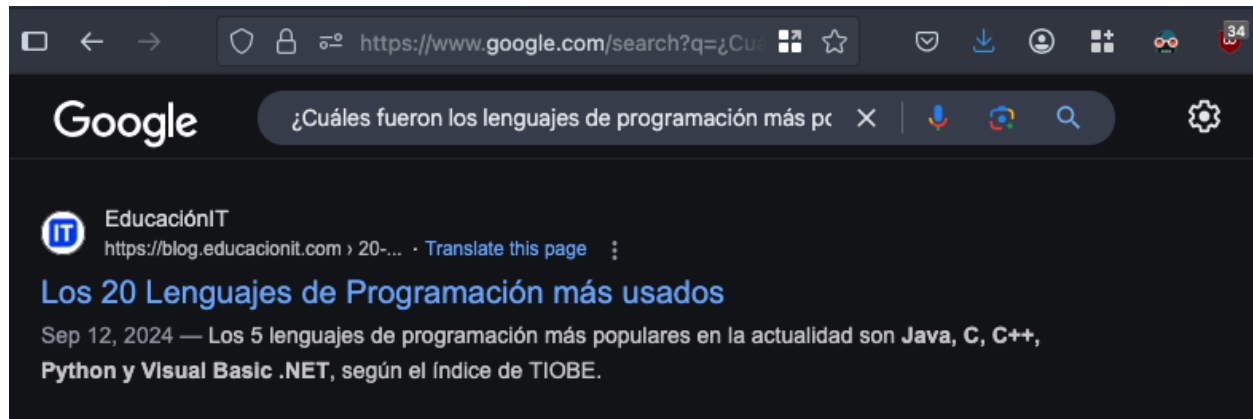
1. Utilizando como recurso el motor de búsqueda Google trate de satisfacer las siguientes necesidades de información. Indique en cada caso la cantidad de consultas que tuvo que realizar para llegar al objetivo (si llegó), cuántas referencias de la respuesta revisó en cada caso y en cuál encontró lo buscado (si es necesario, puede escribir las consultas en inglés). Suponga que usted no sabe la respuesta a ninguna de las preguntas.
  - a. ¿Quién fue el primer campeón olímpico de maratón de los juegos olímpicos? Usando google en lenguaje Español Latinoamericano, realice la búsqueda textual, obteniendo en la primera página la respuesta.



- b. ¿Cuándo se realizó el primer trasplante de corazón?  
Obtención de la respuesta en la primera página.

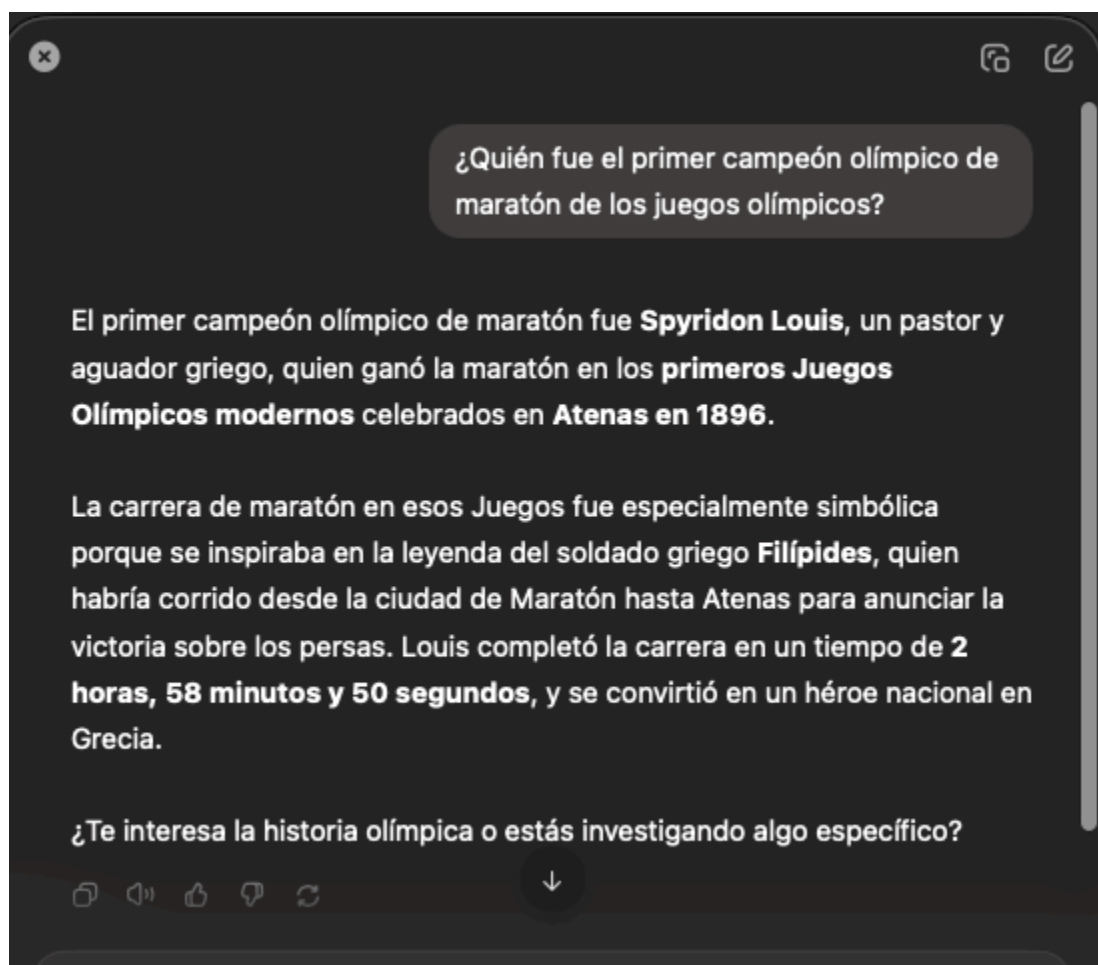
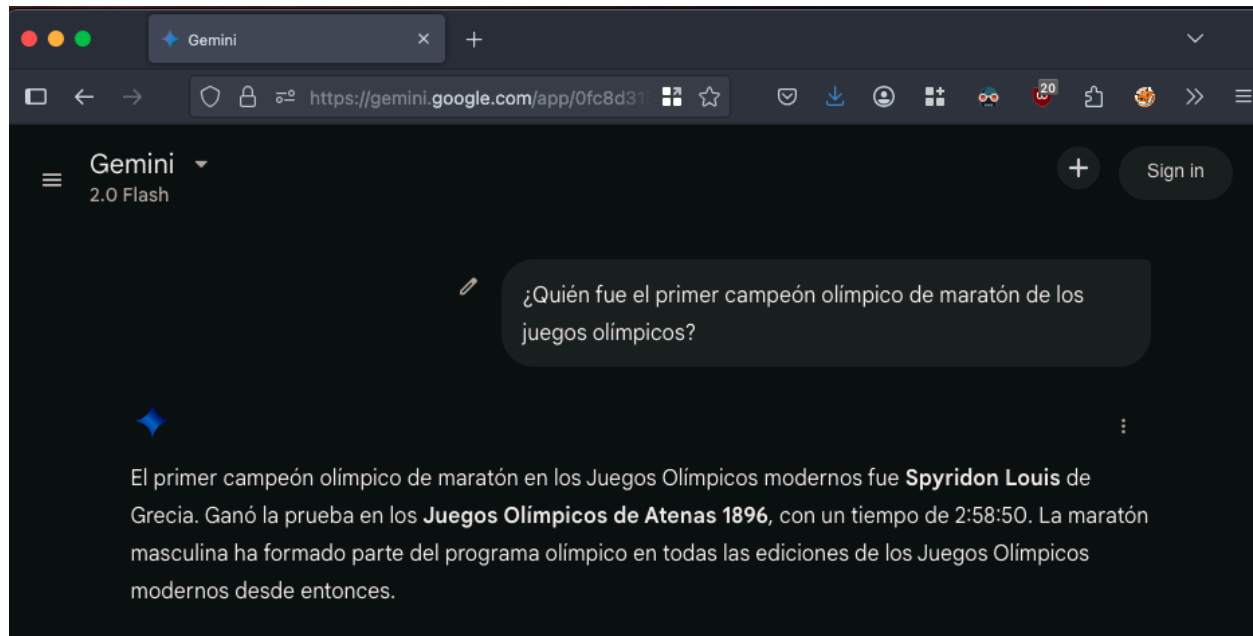


- c. ¿Cuáles fueron los lenguajes de programación más populares el año pasado?  
Al realizar la búsqueda textual de esta pregunta, no se ha encontrado una respuesta realmente a mi criterio satisfactoria dado que “el año pasado” es un tiempo relativo, por lo que para nosotros hoy en día es 2024. Para el buscador no infiere que nos referimos al 2024.  
Aun así creo que esta respuesta es la más acorde teniendo en cuenta que nos muestra la fecha de publicación y esta es la más acorde a nuestra búsqueda.

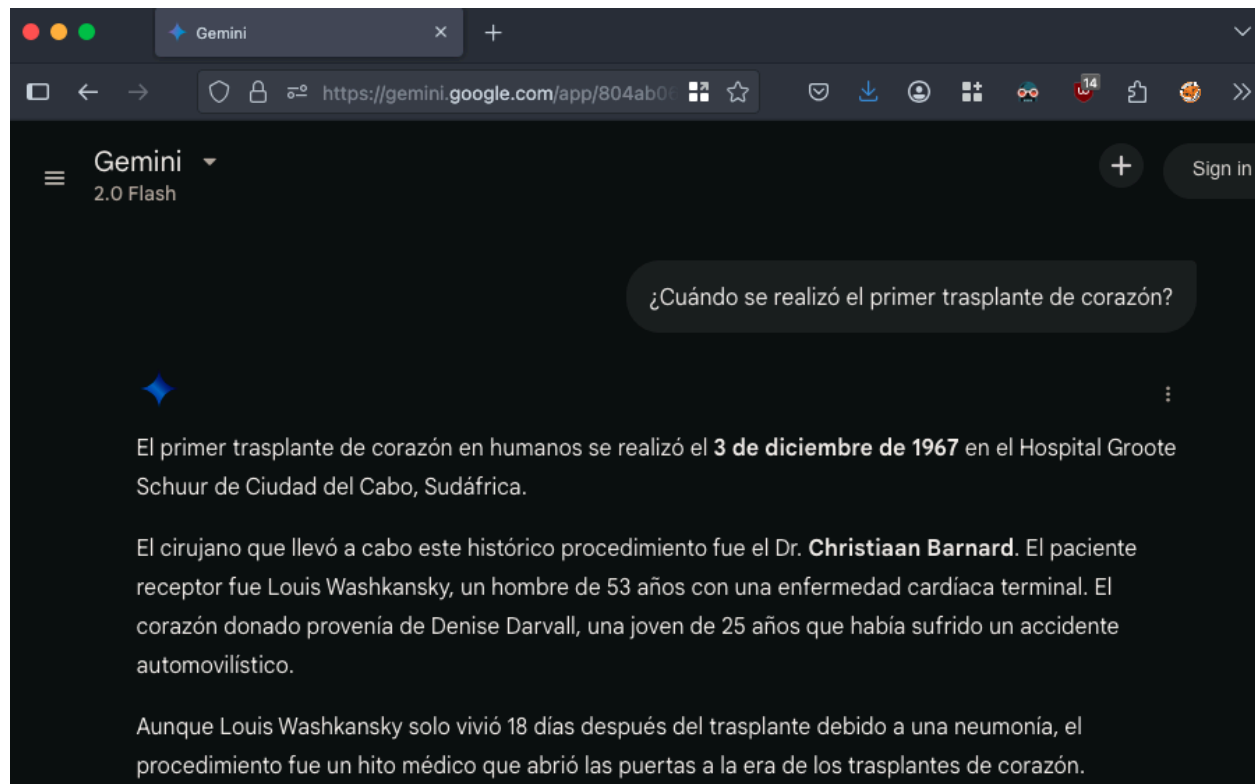


2. Ahora realice las mismas consultas del punto 1) pero empleando ChatGPT y Gemini
- a. ¿Quién fue el primer campeón olímpico de maratón de los juegos olímpicos?  
Usando Gemini, la información es obtenida fácilmente. Además debo decir que en la búsqueda de google, Gemini aparece como una primera opción en los resultados.

Usando Chat GPT la información es igualmente precisa.

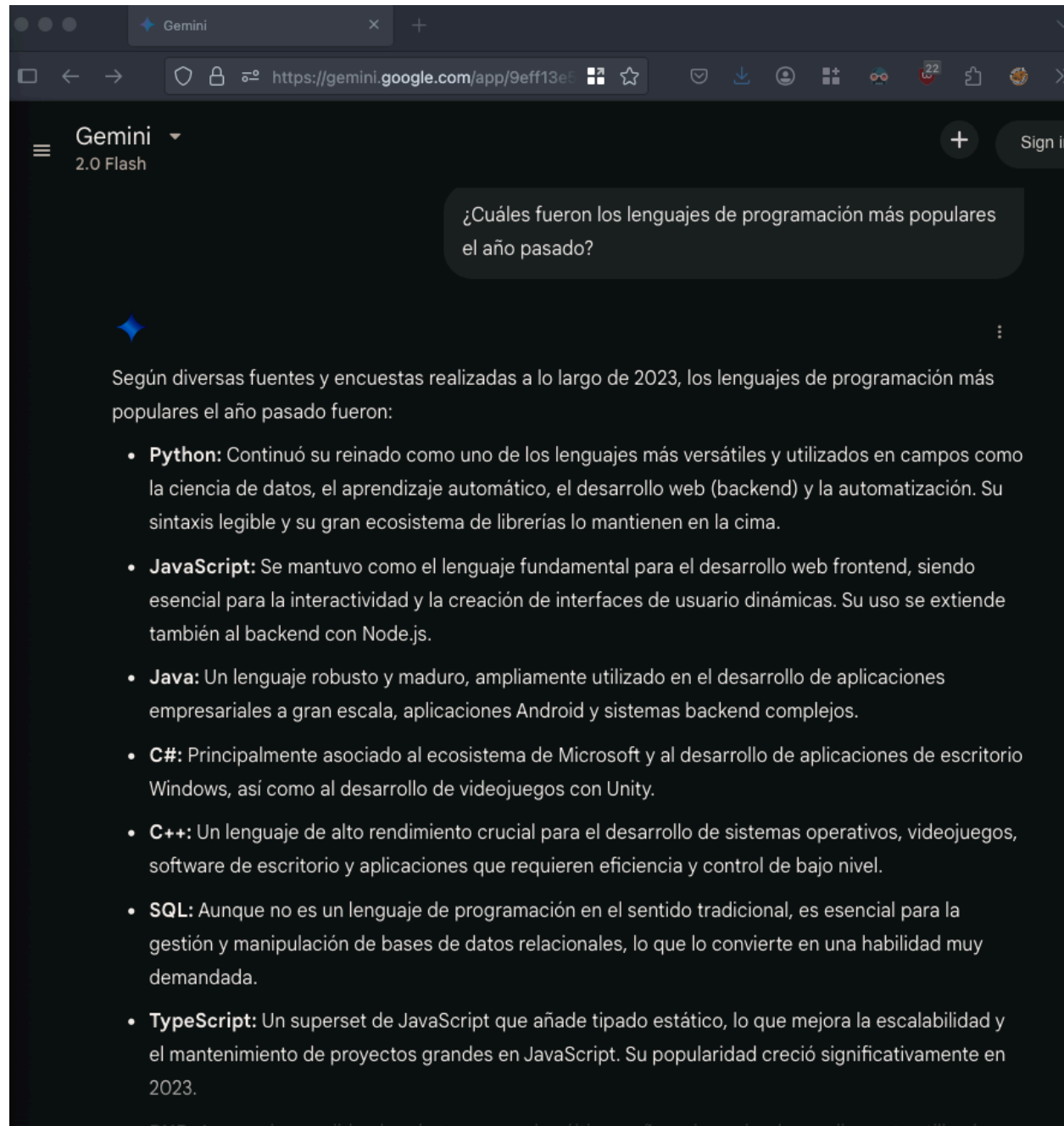


b. ¿Cuándo se realizó el primer trasplante de corazón?



c. ¿Cuáles fueron los lenguajes de programación más populares el año pasado?

Siendo esta la pregunta que requiere inferir el año, los resultados de Gemini vs ChatGPT discrepan, dado que Gemini mostró resultados del 2023 y ChatGPT del 2024. Esto supongo yo tiene que ver con el grado de madurez del modelo LLM y su nivel de actualización en cuanto al conocimiento.



The screenshot shows a web browser window with the Gemini 2.0 Flash interface. The address bar shows the URL `https://gemini.google.com/app/9eff13e5...`. The page header includes the Gemini logo and version "2.0 Flash". A dark input box contains the question: "¿Cuáles fueron los lenguajes de programación más populares el año pasado?". Below the input, a blue star icon indicates a response. The response text states: "Según diversas fuentes y encuestas realizadas a lo largo de 2023, los lenguajes de programación más populares el año pasado fueron:". This is followed by a bulleted list of programming languages and their popularity in 2023.

Según diversas fuentes y encuestas realizadas a lo largo de 2023, los lenguajes de programación más populares el año pasado fueron:


- **Python:** Continuó su reinado como uno de los lenguajes más versátiles y utilizados en campos como la ciencia de datos, el aprendizaje automático, el desarrollo web (backend) y la automatización. Su sintaxis legible y su gran ecosistema de librerías lo mantienen en la cima.
- **JavaScript:** Se mantuvo como el lenguaje fundamental para el desarrollo web frontend, siendo esencial para la interactividad y la creación de interfaces de usuario dinámicas. Su uso se extiende también al backend con Node.js.
- **Java:** Un lenguaje robusto y maduro, ampliamente utilizado en el desarrollo de aplicaciones empresariales a gran escala, aplicaciones Android y sistemas backend complejos.
- **C#:** Principalmente asociado al ecosistema de Microsoft y al desarrollo de aplicaciones de escritorio Windows, así como al desarrollo de videojuegos con Unity.
- **C++:** Un lenguaje de alto rendimiento crucial para el desarrollo de sistemas operativos, videojuegos, software de escritorio y aplicaciones que requieren eficiencia y control de bajo nivel.
- **SQL:** Aunque no es un lenguaje de programación en el sentido tradicional, es esencial para la gestión y manipulación de bases de datos relacionales, lo que lo convierte en una habilidad muy demandada.
- **TypeScript:** Un superset de JavaScript que añade tipado estático, lo que mejora la escalabilidad y el mantenimiento de proyectos grandes en JavaScript. Su popularidad creció significativamente en 2023.

¿Cuáles fueron los lenguajes de programación más populares el año pasado?

Durante el año 2024, los lenguajes de programación más populares fueron los siguientes:

1. **Python:** Se consolidó como el lenguaje más utilizado, gracias a su versatilidad y facilidad de aprendizaje. Es ampliamente empleado en análisis de datos, automatización, inteligencia artificial y desarrollo web. DIARIO LA REPÚBLICA
2. **JavaScript:** Aunque fue superado por Python, sigue siendo esencial para el desarrollo web, tanto en el frontend como en el backend con Node.js. ROOTSTACK
3. **TypeScript:** Este superset de JavaScript ha ganado popularidad por su robustez y seguridad, especialmente en proyectos de gran escala. INGJOSEMARO
4. **Java:** Mantiene su relevancia en aplicaciones empresariales y desarrollo de aplicaciones Android, debido a su estabilidad y escalabilidad. ROOTSTACK
5. **C#:** Popular en el desarrollo de aplicaciones de escritorio, videojuegos con Unity y aplicaciones web, gracias a su integración con la plataforma .NET de Microsoft. ROOTSTACK

Estos lenguajes lideraron en términos de uso y demanda laboral durante 2024. Si estás interesado en aprender alguno de ellos, puedo proporcionarte recursos y cursos gratuitos para comenzar. INGJOSEMARO

 Sources

Ask anything



3. Cree una aplicación que “indexe” los siguientes 5 documentos. Debe formar una colección con todas las palabras, asegúrese de eliminar las stop-words. La aplicación debe leer los documentos .PDF previamente descargados a una ubicación del disco rígido y armar un fichero invertido con frecuencia y por separado un fichero invertido posicional. Con la última representación, genere 4 consultas distintas y muestre los resultados.

<https://sedici.unlp.edu.ar/handle/10915/176466>

<https://sedici.unlp.edu.ar/handle/10915/176745>

<https://sedici.unlp.edu.ar/handle/10915/177333>

<https://sedici.unlp.edu.ar/handle/10915/177453>

<https://sedici.unlp.edu.ar/handle/10915/176292>

## Fichero invertido con frecuencias

### Objetivo del Proyecto:

- Crea un índice que registra cuántas veces aparece cada palabra en cada documento PDF
- Elimina palabras comunes (stopwords) y signos de puntuación

### Funciones Principales:

- **extract\_text\_from\_pdf**: Extrae el texto de los PDFs. Para ello utilice la librería PyPDF2, con la cual abro el PDF en modo binario ('rb'), cuenta el número de páginas y va página por página extrayendo texto, el cual es concatenado en un solo string. Además va controlando si el PDF está roto o si una página no se puede procesar.
- **preprocess\_text**: Limpia el texto (minúsculas, sin puntuación, sin stopwords). Para ello procesa el texto transformando todo a minúsculas y removiendo signos de puntuación. Utilizo también una función extra **get\_stopwords\_from\_github** para descargar las stopwords requeridas. Además que se filtran palabras de 1 letra.
- **create\_inverted\_index**: Crea el índice de frecuencias. Para ello crea un diccionario que mapea cada palabra con su frecuencia en cada documento.

Para ello muestro a continuación una simplificación del código, sin el manejo de errores.

```
# Para cada PDF en la lista de archivos:
for pdf_file in pdf_files:
    # 1. Extrae el texto del PDF
    texto = extract_text_from_pdf(pdf_file)

    # 2. Preprocesa el texto (limpieza)
    tokens = preprocess_text(texto)

    # 3. Cuenta frecuencias
```



```

for palabra in tokens:
    if palabra not in indice:
        indice[palabra] = {}
    if pdf_file.name not in indice[palabra]:
        indice[palabra][pdf_file.name] = 0
    indice[palabra][pdf_file.name] += 1

```

Se va procesando un PDF a la vez.

- **save\_inverted\_index:** Guarda el índice invertido en JSON en la estructura del diccionario.

Estructura del Índice:

```

{
  "palabra1": {
    "documento1.pdf": 5,  // aparece 5 veces
    "documento2.pdf": 3   // aparece 3 veces
  }
}

```

## Fichero invertido posicional

### Objetivo del Proyecto:

Este proyecto crea un índice invertido a partir de documentos PDF, donde se registran las posiciones de cada palabra en el texto y permite calcular la distancia mínima entre dos palabras.

### Funciones Principales:

- **get\_stopwords\_from\_github:** En esta función se obtiene las stopwords en español descargando del repositorio github. Se usa una cache para evitar multiples descargas.
- **extract\_text\_from\_pdf:** Esta funcion es igual a la anterior usada para el índice invertido de frecuencias.
- **create\_inverted\_index:** Esta funcion tiene como entrada la lista de rutas de los archivos PDFs y como salida el diccionario del índice posicional. Con la siguiente estructura:

```

{
  • "palabra1": {
  •   "doc1.pdf": [1, 30, 45], # Posiciones en doc1
  •   "doc2.pdf": [5, 20]      # Posiciones en doc2
  • },

```

```

•     "palabra2": {
•         "doc1.pdf": [10, 35]
•     }
• }

```

Para la construcción de este índice la parte central de código es:

```

for position, token in enumerate(tokens):
    if token not in inverted_index:
        inverted_index[token] = {}
    if pdf_file.name not in inverted_index[token]:
        inverted_index[token][pdf_file.name] = []
    inverted_index[token][pdf_file.name].append(position)

```

En el cual se usa enumeración para mantener la posición y se va creando una estructura jerárquica. Se crea un diccionario por cada nueva palabra y se crea una lista para cada nuevo documento, añadiendo la posición actual.

- **Calculate\_min\_distance:** Esta función calcula la distancia mínima entre dos palabras, las cuales antes son normalizadas (es decir, las pasamos a minúsculas). Primeramente realizamos una verificación de las palabras existen en el índice que recibe la función.

```

• if word1 not in inverted_index or word2 not in inverted_index:
•
•     print(f"Palabras disponibles en el índice:
• {list(inverted_index.keys())[:10]}...")
•
•     return -1, "Palabra no encontrada", []

```

Seguidamente realizamos una búsqueda en documentos comunes, aquí usamos intersección de conjuntos para encontrar documentos que contienen ambas palabras.

```

common_docs = set(inverted_index[word1].keys()) & set(inverted_index[word2].keys())

```

Y por último la parte central donde realizamos el cálculo de distancias:

```

for doc in common_docs:
    positions1 = inverted_index[word1][doc]
    positions2 = inverted_index[word2][doc]

    for pos1 in positions1:
        for pos2 in positions2:
            distance = abs(pos1 - pos2)
            if distance < min_distance:
                min_distance = distance
                document_name = doc
                position_pairs = [(pos1, pos2)]
            elif distance == min_distance:
                position_pairs.append((pos1, pos2))

```

Aquí iteramos sobre los documentos comunes y para cada documento vamos obteniendo las posiciones de ambas palabras, calculando todas las distancias posibles pero solo vamos manteniendo la distancia mínima y sus posiciones. Se acumulan múltiples pares si es que hay mismas distancias mínimas.

- **save\_inverted\_index:** idem a lo explicado para caso de índice de frecuencias.
- **Show\_menu:** Es una función donde se muestra el menu interactivo.

## Consultas y resultados

En la siguiente imagen se muestra como luego de generar el indice, se realizan dos consultar pre-guardadas y sus resultados

```
== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir
=====
Seleccione una opción (1-5): 2
Distancia mínima entre 'proyecto' y 'investigación': 1 en el documento Analisis-sericio-urgencias.pdf
Posiciones encontradas: [(106, 107), (488, 487)]

== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir
=====
Seleccione una opción (1-5): 3
Distancia mínima entre 'inteligencia' y 'moderno': 3 en el documento Algoritmo-aprendizaje-automatizado.pdf
Posiciones encontradas: [(812, 815), (841, 844)]

== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir
=====
Seleccione una opción (1-5): █
```

En la siguiente imagen realizamos dos consultas mas, pero ingresando manualmente dos palabras cuando elegimos la opcion 4

```

== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir

=====
Seleccione una opción (1-5): 4
Ingrese la primera palabra: red
Ingrese la segunda palabra: historia
Distancia mínima entre 'red' y 'historia': 239 en el documento Algoritmo-aprendizaje-automatizado.pdf
Posiciones encontradas: [(610, 849)]

== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir

=====
Seleccione una opción (1-5): 4
Ingrese la primera palabra: siddhartha
Ingrese la segunda palabra: processing
Distancia mínima entre 'siddhartha' y 'processing': 11 en el documento Algoritmo-vinculacion-relatos-denuncias.pdf
Posiciones encontradas: [(1661, 1672)]

== Menú de Opciones ==
1. Generar índice invertido
2. Calcular distancia entre 'proyecto' e 'investigación'
3. Calcular distancia entre 'inteligencia' y 'moderno'
4. Calcular distancia entre palabras personalizadas
5. Salir

=====
Seleccione una opción (1-5): █

```

- El siguiente es un resumen que se encuentra construido con un índice invertido, se indica el tamaño del mismo y cada palabra posee entre corchetes la posición que ocupa dentro del texto. Cree una aplicación que permita recibir estos dos parámetros, el tamaño del índice y el índice invertido, y devuelva el resumen completo. Cree las estructuras de datos que considere necesario tanto para la reconstrucción del texto como para el pasaje de parámetros.

Tamaño del Índice: 184

Índice Invertido:

```

{"COVID-19": [0], "caused": [1], "by": [2], "the": [3, 16, 26, 31, 36, 54, 62, 70, 78, 85, 97, 110, 117, 125, 146, 162, 173, 178], "coronavirus": [4], "SARS-CoV2": [5], "continues": [6], "to": [7, 58, 73, 77, 129], "pose": [8], "a": [9, 170], "global": [10, 37], "threat": [11], "The": [13], "consequences": [14], "from": [15, 116], "pandemic": [17], "are": [18], "devastating": [19], "not": [20, 141], "only": [21], "for": [22, 35, 124, 172], "human": [23], "health": [24, 28], "and": [25, 43, 82, 103, 132], "national": [27], "systems": [29], "throughout": [30], "world": [32], "but": [33], "also": [34, 107], "economy": [38], "Bats": [40], "pangolins": [41], "muskrats": [42], "other": [44, 136], "wild": [45], "animals": [46, 128, 137], "have": [47, 74], "been": [48, 122], "implicated": [49], "as": [50, 87], "possible": [51], "hosts": [52], "of": [53, 65, 84, 93, 100, 112, 127, 148, 164, 175], "virus": [55], "According": [57], "recent": [59], "research": [60], "data": [61], "increased": [63], "concentration": [64], "particulate": [66], "air": [67, 94], "matter": [68], "in": [69, 145, 152, 177], "atmosphere": [71], "appears": [72, 108], "contributed": [75], "significantly": [76], "spread": [79], "airborne": [80], "transmission": [81], "aggressiveness": [83], "disease": [86], "

```

```
large": [88], "cities": [89], "with": [90, 135], "high": [91], "levels": [92], "pollution": [95], "showed": [96], "highest": [98], "numbers": [99], "SARS-CoV2": [101], "cases": [102], "deaths": [104], "It": [106, 157], "that": [109, 138, 161], "collapse": [111], "natural": [113], "ecosystems": [114], "resulting": [115, 144], "ever-evolving": [118], "climate": [119], "change": [120], "has": [121, 181], "crucial": [123], "migration": [126], "new": [130, 149], "areas": [131], "their": [133], "interaction": [134], "they": [139], "would": [140], "normally": [142], "encounter,": [143], "emergence": [147], "pathogens,": [150], "which": [151, 167], "turn": [153], "infected": [154], "humans": [155], "is": [158], "now": [159], "clear": [160], "era": [163], "climate": [165], "medicine": [166], "will": [168], "be": [169], "cornerstone": [171], "practice": [174], "medicine": [176], "21st": [179], "century,": [180], "already": [182], "begun": [183]}
```

## Reconstrucción de texto

### Objetivos del proyecto:

El script permite reconstruir un texto original utilizando un índice invertido previamente generado, donde cada palabra está asociada con sus posiciones en el texto original.

### Funciones Principales:

- **process\_reconstruction:** Esta función orquesta el proceso completo de reconstrucción. Va manejando la secuencia de operaciones:
  - **get\_index\_file:** solicita y valida la ruta del archivo json del índice.
  - **get\_text\_length:** Obtiene la longitud del texto a construir
  - **load\_inverted\_index:** Lee el archivo y retorna el diccionario con palabras y posiciones.
  - **reconstruct\_text:** Esta es la función principal, en la cual se crea una lista vacía del tamaño original y se coloca cada palabra con sus posiciones correspondientes. Finalmente une las palabras para formar el texto final

```
try:
    # Crear lista de palabras vacías
    reconstructed = [''] * text_length
    # Colocar cada palabra en su posición
    for word, positions in inverted_index.items():
        for pos in positions:
            if pos < text_length:
                reconstructed[pos] = word
            else:
                logger.warning(f"Posición {pos} fuera de rango para la palabra '{word}'")
```

```

# Unir las palabras con espacios
text = ' '.join(reconstructed)

logger.info("Texto reconstruido exitosamente")

return text

except Exception as e:

    logger.error(f"Error al reconstruir el texto: {str(e)}")

    raise

```

## Ejecución

```

(reconstruir-texto) (* |infra:bsee)jriera:reconstruir-texto/ (main*) $ python main.py
[23:18:55]

=== Menú de Reconstrucción de Texto ===
1. Reconstruir texto desde índice invertido
2. Salir
=====

Seleccione una opción (1-2): 1

Ingrese la ruta del archivo de índice (ej: index.json): index.json

Ingrese la longitud del texto (número de palabras): 184
2025-04-15 23:19:13,898 - INFO - Índice invertido cargado desde: index.json
2025-04-15 23:19:13,899 - INFO - Texto reconstruido exitosamente

=== Texto Reconstruido ===
COVID-19, caused by the coronavirus SARS-CoV2, continues to pose a global threat. The consequences from the pandemic are devastating, not only for human health and the national health systems throughout the world, but also for the global economy. Bats, pangolins, muskrats and other wild animals have been implicated as possible hosts of the virus. According to recent research data, the increased concentration of particulate air matter in the atmosphere appears to have contributed significantly to the spread (airborne transmission) and aggressiveness of the disease, as large cities with high levels of air pollution showed the highest numbers of SARS-CoV2 cases and deaths. It also appears that the collapse of natural ecosystems resulting from the ever-evolving change has been crucial for the migration of animals to new areas and their interaction with other animals that they would not normally encounter, resulting in the emergence of new pathogens, which in turn infected humans. It is now clear that the era of climate change, which will be a cornerstone for the practice of medicine in the 21st century, has already begun.

=====
2025-04-15 23:19:13,900 - INFO - Texto reconstruido guardado en: reconstructed_text.txt

Proceso completado exitosamente!
Texto reconstruido guardado en: reconstructed_text.txt
Logs guardados en: /Users/jriera/Documents/unju/rain/tp1/reconstruir-texto/reconstruction_logs.txt

```