



Instituto Tecnológico de Costa Rica
Recuperación de información textual GR 20

Índice Invertido

Integrantes:

Christian Acuña Sáenz (2013038916)

Fernando González Ramón (2021075114)

Andrew Gutierrez Castro (2019068322)

Profesor:

Aurelio Sanabria Rodriguez

Fecha de entrega:

28 de mayo, 2022

Semestre I - 2022

Índice

- [Motivación](#)
- [Justificación de la implementación](#)
- [Análisis de resultados](#)
- [Memes](#)
- [Link Repo](#)

- **Motivación**

Lo que se busca en este proyecto es la realización de un índice invertido. Dicho instrumento es conocido como estructura de datos que tiene como principal objetivo el acelerar los procesos de búsqueda en la colección. Para esto, se arma un diccionario con ayuda las palabras que se encuentran en los documentos de la colección, para más adelante poder determinar en una serie de listas (las cuales pertenecen a cada documento) en que documento se encuentra “x” término.

Además de esto, mencionar que los índices son herramientas de gran ayuda en los trabajos que respecta a la recuperación de información, ya que como se mencionó anteriormente tienen un gran aporte en la mejora de eficiencia de las búsquedas. Por último, recalcar algunos de los parámetros de eficiencia relacionados con los índices:

- Tiempo de indexación
- Espacio para el almacenamiento
- Consultas realizadas por un determinado tiempo

- **Justificación de la implementación**

En lo que se refiere a los códigos implementados para el índice invertido su funcionamiento se basa principalmente en el trabajo de dos de ellos, el “invertedIndex.py” y el “Calculo.py”. Estos son los encargados de la construcción del mismo índice y del cálculo de las fórmulas mencionadas en la especificación, respectivamente.

Por el lado del “invertedIndex.py” este trabaja de manera que, con ayuda de un hashMap logra capturar la información de todos los documentos previamente recuperados por el Scraper y los introduce en una lista (estando los documentos en diccionarios). Después de esto, lo que procede a realizar es generar un “vocabulario” con todos los términos de los documentos y ya una vez con este procede a encasillar las diferentes palabras en el los diccionarios respectivos de cada documento individual en caso de que la expresión se encuentre dentro de él.

Y en la otra cara del proyecto se encontraría el código responsable de realizar todas las fórmulas referentes a la eficiencia del índice, estas son de distintos ámbitos y se encargan cada una de brindar un dato distinto de los diccionarios de cada documento. Entre las fórmulas se pueden encontrar:

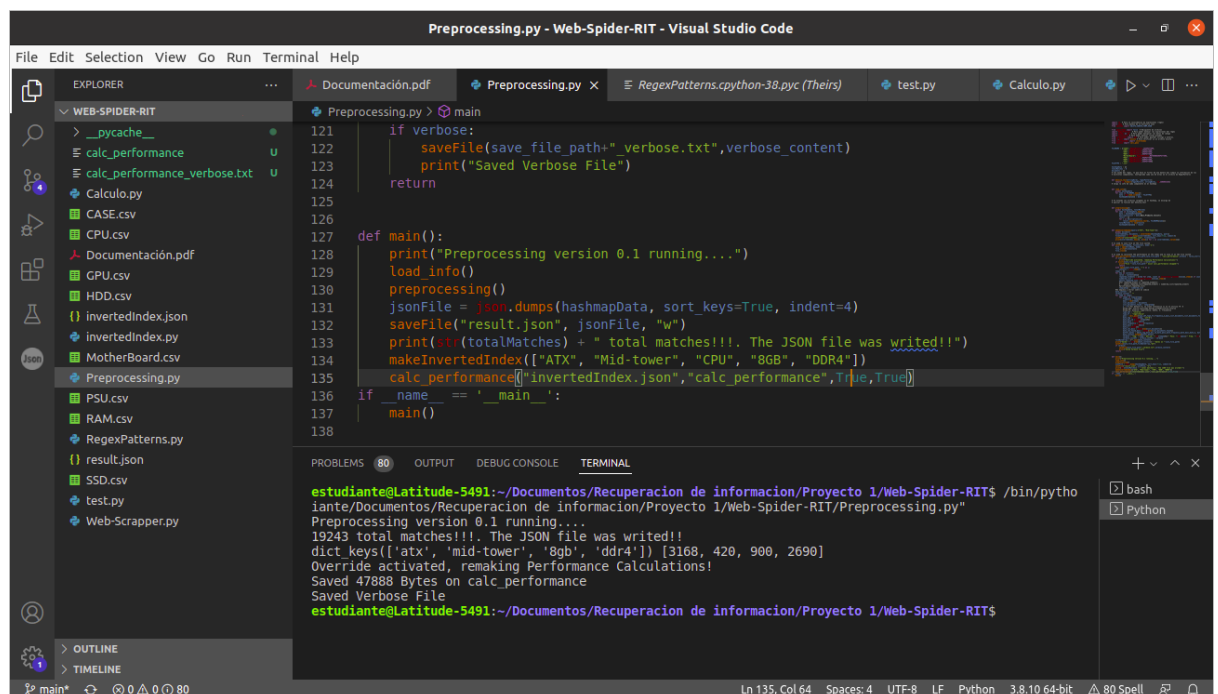
- ❖ **Peso:** Esta se encarga de calcular el peso de los diferentes términos de los documentos y también de los documentos en la colección.
- ❖ **TF:** El cálculo de tf se centra en dar prioridad a la comparación del término exacto del documento y devuelve la relevancia con la que cuenta.
- ❖ **IDF:** El idf por su parte trabaja con los valores obtenidos de los pesos de los documentos. De esta manera devuelve la relevancia que tiene ese documento comparándolo con el resto de documentos de la colección.

Mencionar que además de las fórmulas que fueron explicadas, el código de “Calculo.py” cuenta con distintas variantes de las mismas, las cuales brindan valores más exactos de alguna en específico ej: el cálculo del tf por frecuencia o el idf de frecuencia inversa máxima.

Por último, mencionar que los resultados obtenidos de ambos códigos, o sea, el mismo índice y las fórmulas de las consultas son guardados en un json y en un binario respectivamente.

- **Análisis de resultados**

Como se mencionó anteriormente, el “invertedIndex.py” y el “Calculo.py” que son los codigos mas fundamentales para el funcionamiento del índice se guardan sus respectivos archivos, guardando al índice en un .json y haciendo los cálculos de las fórmulas en un binario. A continuación, un ejemplo de todo el funcionamiento del código y su manera de trabajar.

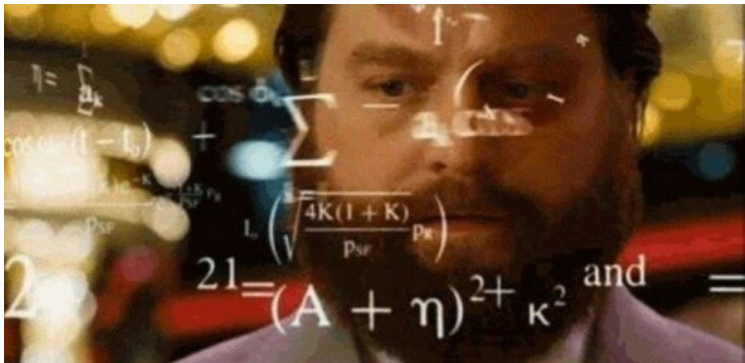


```
Preprocessing.py - Web-Spider-RIT - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
WEB-SPIDER-RIT
  __pycache__
  calc_performance
  calc_performance_verbose.txt
  Calculo.py
  CASE.csv
  CPU.csv
  Documentación.pdf
  GPU.csv
  HDD.csv
  invertedIndex.json
  invertedIndex.py
  MotherBoard.csv
  Preprocessing.py
  PSU.csv
  RAM.csv
  RegexPatterns.py
  result.json
  SSD.csv
  test.py
  Web-Scraper.py
Preprocessing.py
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
def main():
    print("Preprocessing version 0.1 running...")
    load_info()
    preprocessing()
    jsonFile = json.dumps(hashmapData, sort keys=True, indent=4)
    saveFile("result.json", jsonFile, "w")
    print(str(totalMatches) + " total matches!!!. The JSON file was writed!!!")
    makeInvertedIndex(["ATX", "Mid-tower", "CPU", "8GB", "DDR4"])
    calc_performance(["invertedIndex.json", "calc_performance", True, True])
if __name__ == '__main__':
    main()
PROBLEMS 80 OUTPUT DEBUG CONSOLE TERMINAL
estudiante@Latitude-5491:~/Documentos/Recuperacion de informacion/Proyecto 1/Web-Spider-RIT$ /bin/pytho
iante/Documentos/Recuperacion de informacion/Proyecto 1/Web-Spider-RIT/Preprocessing.py"
Preprocessing version 0.1 running...
19243 total matches!!!. The JSON file was writed!!
dict keys(['atx', 'mid-tower', '8gb', 'ddr4']) [3168, 420, 900, 2690]
Override activated, remaking Performance Calculations!
Saved 47888 Bytes on calc_performance
Saved Verbose File
estudiante@Latitude-5491:~/Documentos/Recuperacion de informacion/Proyecto 1/Web-Spider-RIT$
Ln 135, Col 64 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit 80 Spell
```

Primero se termina de correr el archivo, el cual previamente empezó a generar el índice, luego de esto se empezaron a calcular los pesos utilizando las fórmulas de: idf de frecuencia inversa la cual se multiplicaría con la fórmula de tf de normalización logarítmica.

- Memes

Christian Acuña Sáenz



btw these large scary math symbols
are just for-loops

Summation <small>(capital sigma)</small>	$\sum_{n=0}^4 3n$	<pre>sum = 0; for(n=0; n<=4; n++) sum += 3*n;</pre>
Product <small>(capital pi)</small>	$\prod_{n=1}^4 2n$	<pre>prod = 1; for(n=1; n<=4; n++) prod *= 2*n;</pre>





Fernando González Ramón

WHO WOULD WIN?

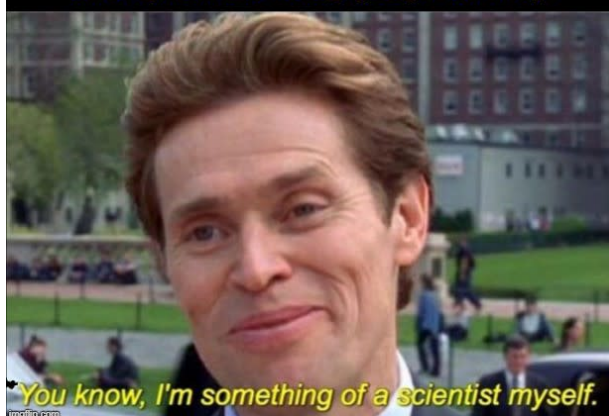
a computer program with
millions of lines of code



one C U R L Y B O Y
with no friend



When you do a regex expression correctly
first try without using google for help

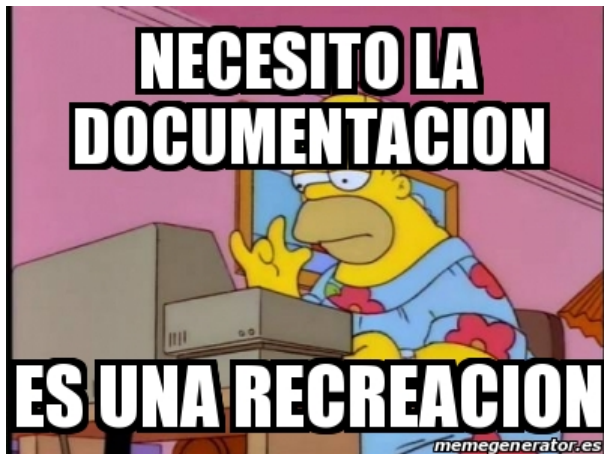


Cuando ya va a acabar el semestre y los maestros siguen dando trabajos



Andrew Gutierrez Castro







- Repo Gitlab

<https://gitlab.com/trabajos-rit/indice-invertido>