Informe sobre el Lab 6.2 de Base de Datos 2.

INTEGRANTES
- Ricardo Amiel Acuña Villogas
- Rodrigo Lauz Nakasone

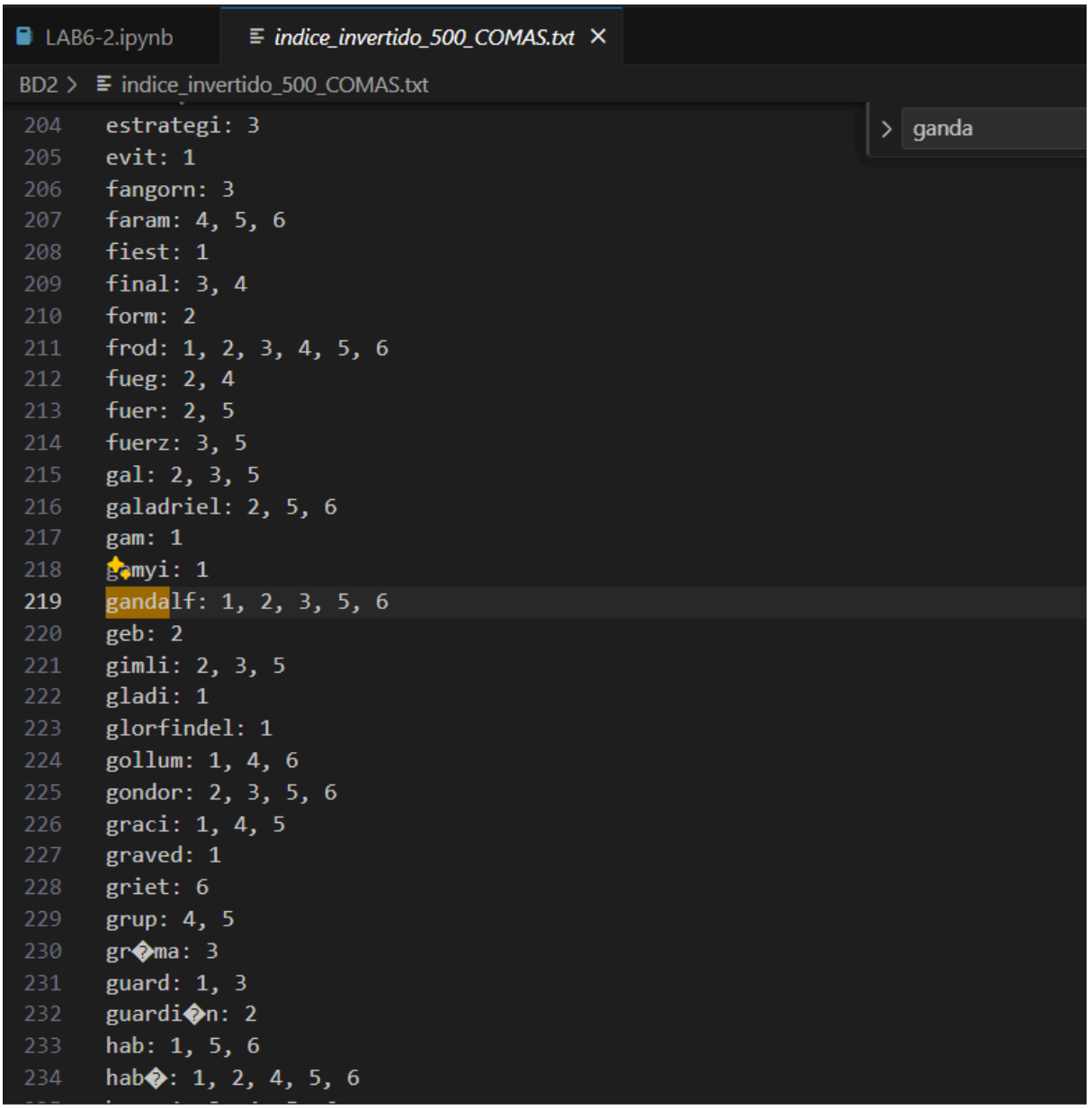
1. **IMPLEMENTACIÓN DEL ÍNDICE INVERTIDO:**

- Se ocupó la librería **Collections**, la cual cuenta con un contador de frecuencias por cada elemento, justo lo que necesitamos. Además, cuenta con un método most common(n), el cual recibe como parámetro un entero, por lo que de esa manera obtuvimos las 500 palabras más repetidas.
- La función toma dos listas: la que guarda los 6 libros y la de las n palabras más comunes.
- Se preprocesan todas, se ordenan alfabéticamente y luego por el libro en el que aparecen. El diccionario guarda como key **la palabra y en cuantos libros aparece, y como value guarda el libro y las veces que aparece en él,** finalmente se guarda en un archivo txt**.**
- **Código:**

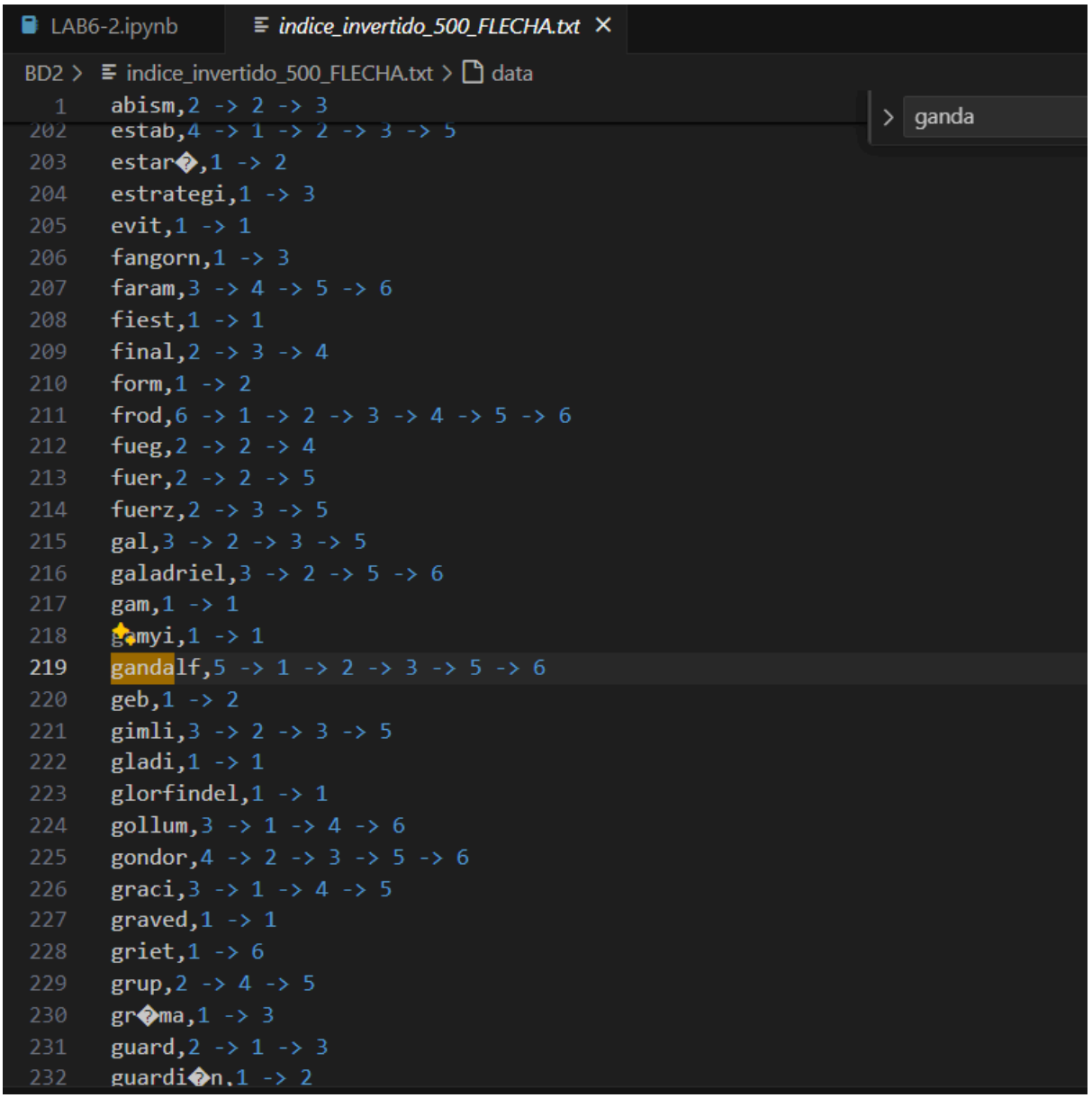
```
word_freq = Counter()
# Contar la frecuencia de cada palabra en cada libro preprocesado
for text in lista_libros:
    word_freq.update(text)

most_commom_words = word_freq.most_common(500)
most_commom_words = [word for word, freq_word in most_commom_words]
list_docs = ['libro1.txt','libro2.txt','libro3.txt','libro4.txt','libro5.txt','libro6.txt']

# a) construir el índice invertido de las 500 palabras más frecuentes [lexemas]
def indice_invertido_common_words(docs, most_common_words):
    index = defaultdict(lambda: defaultdict(int))
    for i, doc in enumerate(docs):
        words = preprocesamiento(doc)
        for word in words:
            if word in most_common_words:
                index[word][i+1] += 1

    # Convertir el índice a listas ordenadas de tuplas
    for word, doc_freqs in index.items():
        index[word] = sorted(doc_freqs.items())

    index = sorted(index.items())

    # Convertir el índice a una cadena de texto
    index_str = ""
    for word, doc_freqs in index:
        index_str += f"{word},{len(doc_freqs)} -> "
        index_str += " -> ".join(f"{doc_id}" for doc_id, _ in doc_freqs)
        index_str += "\n"

    # Guardar el índice en un archivo de texto
    with open("indice_invertido_500_FLECHA.txt", "w") as file:
        file.write(index_str)

    return index
```

- Acá hay una muestra de nuestro índice invertido(existen 2 formatos => separado por comas, separado por flechas):

POR COMAS

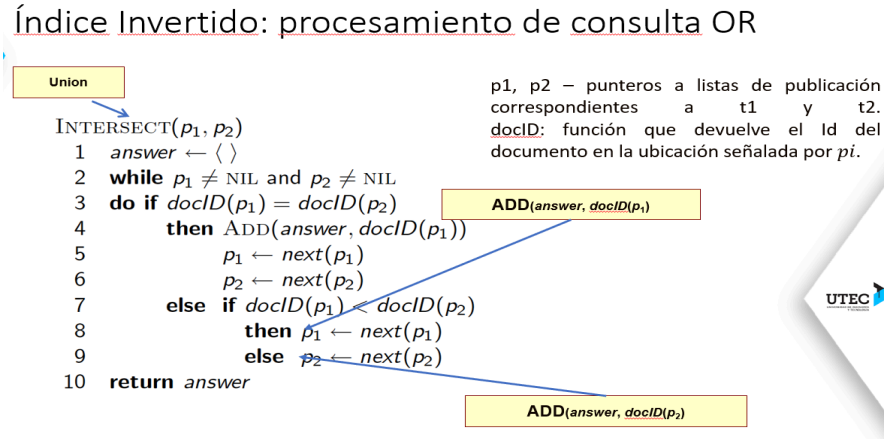


POR FLECHAS

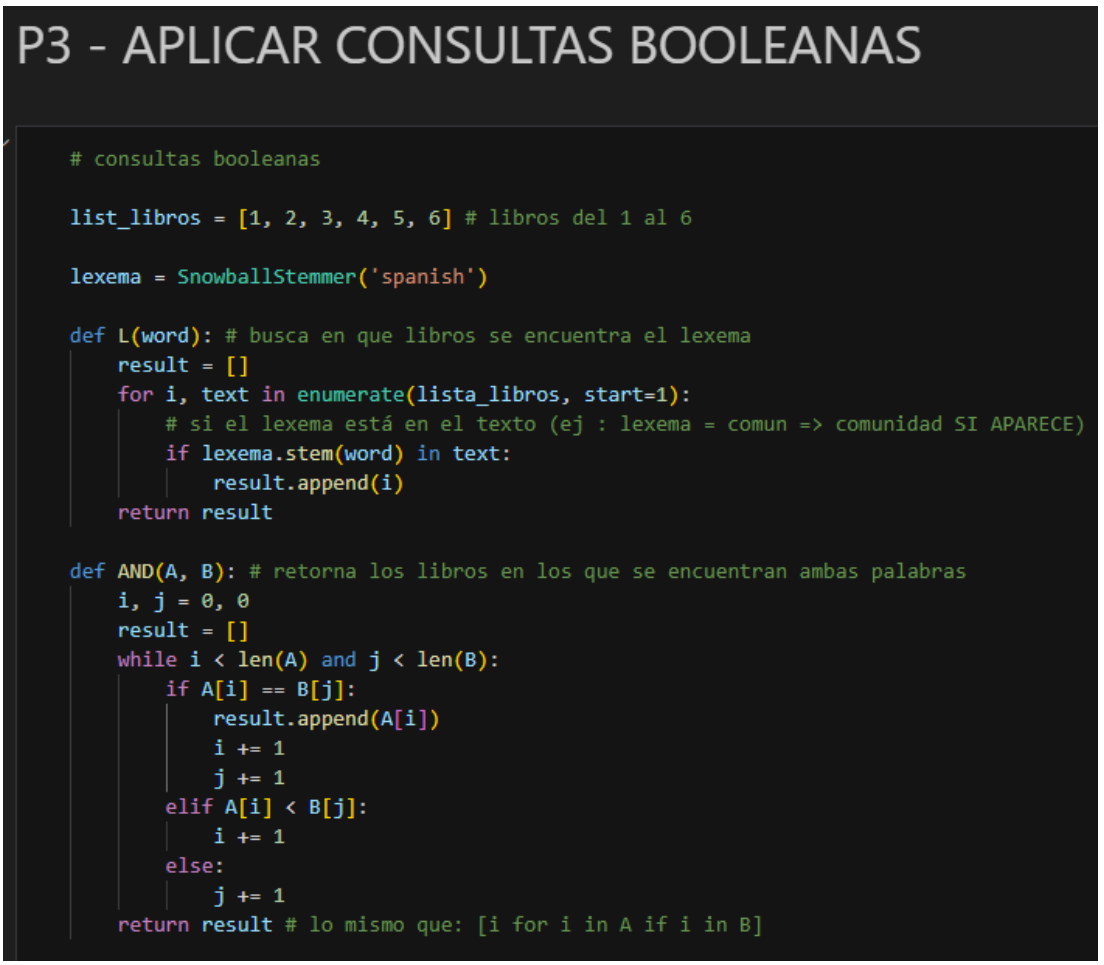

LAB6-2.ipynb
indice_invertido_500_FLECHA.txt
BD2 > indice_invertido_500_FLECHA.txt > data
ganda
1 abism,2 -> 2 -> 3
202 estab,4 -> 1 -> 2 -> 3 -> 5
203 estar,1 -> 2
204 estrategi,1 -> 3
205 evit,1 -> 1
206 fangorn,1 -> 3
207 faram,3 -> 4 -> 5 -> 6
208 fiest,1 -> 1
209 final,2 -> 3 -> 4
210 form,1 -> 2
211 frod,6 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6
212 fueg,2 -> 2 -> 4
213 fuer,2 -> 2 -> 5
214 fuerz,2 -> 3 -> 5
215 gal,3 -> 2 -> 3 -> 5
216 galadriel,3 -> 2 -> 5 -> 6
217 gam,1 -> 1
218 gamyi,1 -> 1
219 gandalf,5 -> 1 -> 2 -> 3 -> 5 -> 6
220 geb,1 -> 2
221 gimli,3 -> 2 -> 3 -> 5
222 gladi,1 -> 1
223 glorfindel,1 -> 1
224 gollum,3 -> 1 -> 4 -> 6
225 gondor,4 -> 2 -> 3 -> 5 -> 6
226 graci,3 -> 1 -> 4 -> 5
227 graved,1 -> 1
228 griet,1 -> 6
229 grup,2 -> 4 -> 5
230 gr?ma,1 -> 3
231 guard,2 -> 1 -> 3
232 guardi?n,1 -> 2

2. FUNCIONES BOOLEANAS

- Para implementación de las funciones booleanas, es sencillo tomar los libros enumerados del 1 al 6 como enteros, por lo que la comparación fue sencilla de realizar.
- Nuestra **función L recibe una palabra y la busca en la lista de listas de lexemas de cada libro, y verifica si al sacarle el lexema a la palabra coincide**. A partir de ello se construyeron las 4 operaciones lógicas AND, OR, NOT y ANDNOT. Nos basamos en el pseudocódigo de la ppt, con la única diferencia que los parámetros que recibe son una función L() y los 'pointers' actúan tal cual memoria secundaria, adjunto imagen:



- **Código:**



3. RESULTADOS

- Tomé de ejemplo las consultas del laboratorio para probar los operadores lógicos, en base a ello generé 3 consultas que usaban 3 palabras, las 2 primeras consultas ocupan las palabras comunidad, frodo y gondor, por otra parte **la tercera consulta fue elegida porque Gandalf es una de las palabras que más aparece en los libros del Señor de los Anillos.**

```
    #usar las palabras gandalf, hermana y gracias
    consulta3 = OR(ANDNOT(L('gandalf'), L('hermana')), L('gracias')) # 
    print(f'(gandalf AND-NOT hermana) OR gracias: {consulta3}')
```

```
Comunidad en los libros:  [2]
Frodo en los libros:  [1, 2, 3, 4, 5, 6]
Gondor en los libros:  [2, 3, 5, 6]

comunidad AND frodo: [2]
comunidad OR frodo: [1, 2, 3, 4, 5, 6]
NOT comunidad: [1, 3, 4, 5, 6]
frodo AND-NOT gnndor: [1, 4]

Ejemplo de consultas:

(comunidad AND frodo) AND-NOT gondor: []
(comunidad AND frodo) OR gondor: [2, 3, 5, 6]
(gandalf AND-NOT hermana) OR gracias: [1, 2, 3, 4, 5]
```

4. CONCLUSIONES

- Se ha logrado construir un índice invertido con las 500 palabras más frecuentes. Con la posibilidad de realizar búsqueda por palabra o por lexema debido a la creación de los archivos. En la versión final solo se subió lo requerido, pero se realizaron pruebas ya que el lexema lograba reducir la redundancia que se tenía al guardar por palabras.
- Se han implementado las funciones booleanas para realizar consultas, la librería nltk, re y collections fueron de ayuda por su manera sencilla de manejar los lexemas y generar contadores en los ítems de un diccionario.

REFERENCIAS

- Notas de clase
- https://docs.python.org/3/library/collections.html
- https://www.nltk.org/

repositorio: https://github.com/ricardoamiel/6lab