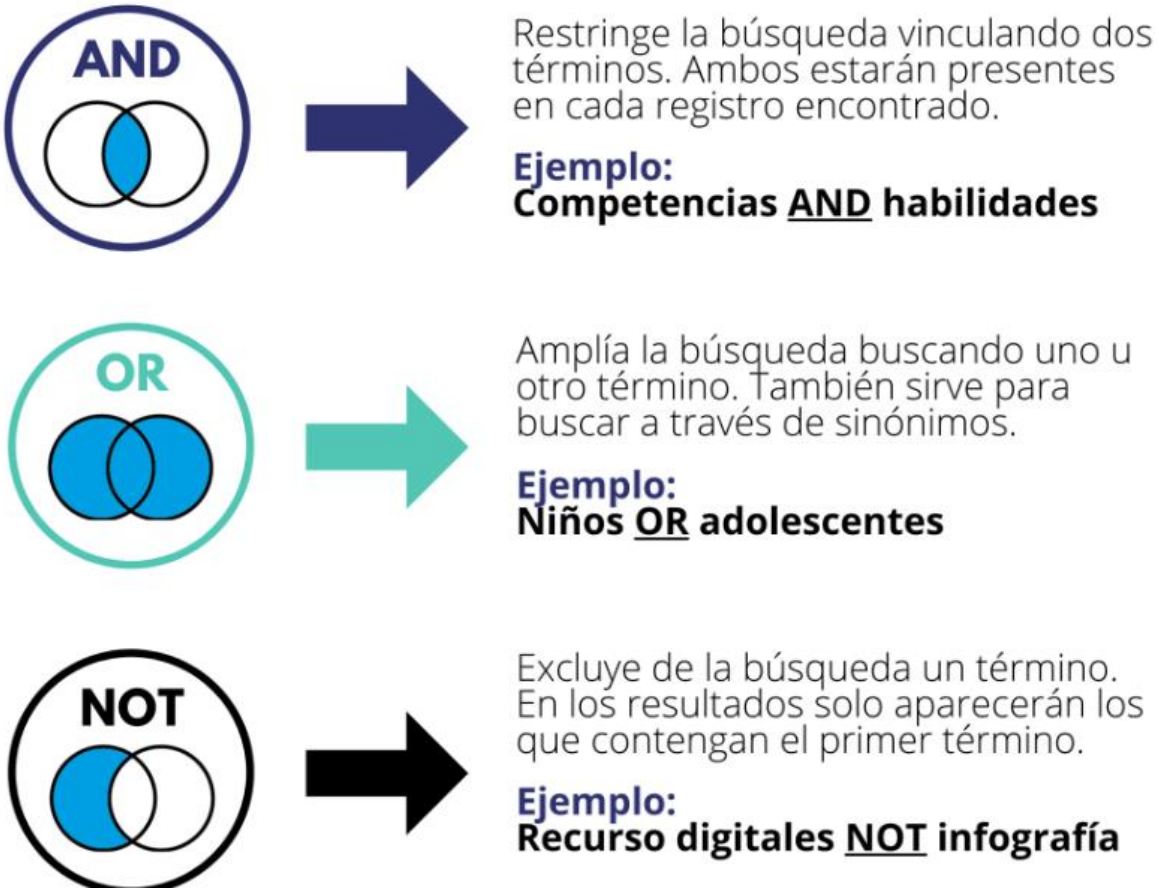
INSTITUTO TECNOLÓGICO BELTRÁN
Centro de Tecnología e Innovación

Búsqueda booleana: Práctica 2

Alumno: Marco Medina
Fecha: 23/06
Instituto: Instituto Tecnologico Beltran



- En esta práctica si bien tiene el mismo objetivo que la práctica 1, este en particular usa bibliotecas distintas que luego nos ayudarán a mejorar la búsqueda. Por ejemplo:
    - re: modulo para usar expresiones regulares.
    - spacy: biblioteca para procesamiento del lenguaje natural.
    - nlp: objetivo que spaCy usa para analizar texto. Aquí cargamos el modelo entrenado para español, que conoce la gramática y morfología del idioma.

Se realiza la descarga de las bibliotecas:

```
import re
import spacy

# Cargar el modelo de spaCy para español
nlp = spacy.load("es_core_news_sm")
```

- Teniendo las bibliotecas ya se procede a limpiar y preparar el texto.

```
# Función para limpiar, tokenizar y lematizar
def limpiar_y_tokenizar(texto):
    texto = texto.lower()
    doc = nlp(texto)
    lemas = [token.lemma_ for token in doc if token.is_alpha]
    return lemas
```

- Luego si crea el índice invertido:

```
def crear_indice_invertido(documentos):
    indice = {}
    for doc_id, contenido in documentos.items():
        tokens = limpiar_y_tokenizar(contenido)
        for token in tokens:
            indice.setdefault(token, set()).add(doc_id)
    return indice
```

- Ya teniendo el texto preparado se realiza la funcion que realizara la busqueda, en esta:
    - Para cada palabra de la consulta, spaCy lematiza el término.
    - Se usa el lema para consultar el índice y obtener los documentos donde aparece.
    - Se hace luego la operación lógica correspondiente (intersección, unión o resta).

```python
# Procesar consultas booleanas
def procesar_consulta(consulta, indice):
    consulta = consulta.lower()
    resultado = set()

    if ' and ' in consulta:
        partes = consulta.split(' and ')
        sets = [indice.get(nlp(p.strip())[0].lemma_, set()) for p in partes]
        resultado = sets[0].intersection(*sets[1:])

    elif ' or ' in consulta:
        partes = consulta.split(' or ')
        sets = [indice.get(nlp(p.strip())[0].lemma_, set()) for p in partes]
        resultado = set().union(*sets)

    elif ' not ' in consulta:
        partes = consulta.split(' not ')
        incluyente = indice.get(nlp(partes[0].strip())[0].lemma_, set())
        excluyente = indice.get(nlp(partes[1].strip())[0].lemma_, set())
        resultado = incluyente - excluyente

    else:
        resultado = indice.get(nlp(consulta.strip())[0].lemma_, set())

    return resultado
```

- Finalmente corremos las consultas del siguiente texto desde el mismo código.

```
"doc1": "Los egipcios construyeron las pirámides y desarrollaron una escritura jeroglífica.",
"doc2": "La civilización romana fue una de las más influyentes en la historia occidental.",
"doc3": "Los mayas eran expertos astrónomos y tenían un avanzado sistema de escritura.",
"doc4": "La antigua Grecia sentó las bases de la democracia y la filosofía moderna.",
"doc5": "Los sumerios inventaron la escritura cuneiforme y fundaron las primeras ciudades."
```

consultas:

```python
# Construcción y prueba
indice = crear_indice_invertido(documentos)

consultas = [
    "egipcios AND pirámides",
    "escritura OR astrónomos",
    "romano NOT griegos"
]

for c in consultas:
    resultado = procesar_consulta(c, indice)
    print(f"Documentos encontrados :: {resultado}")
```