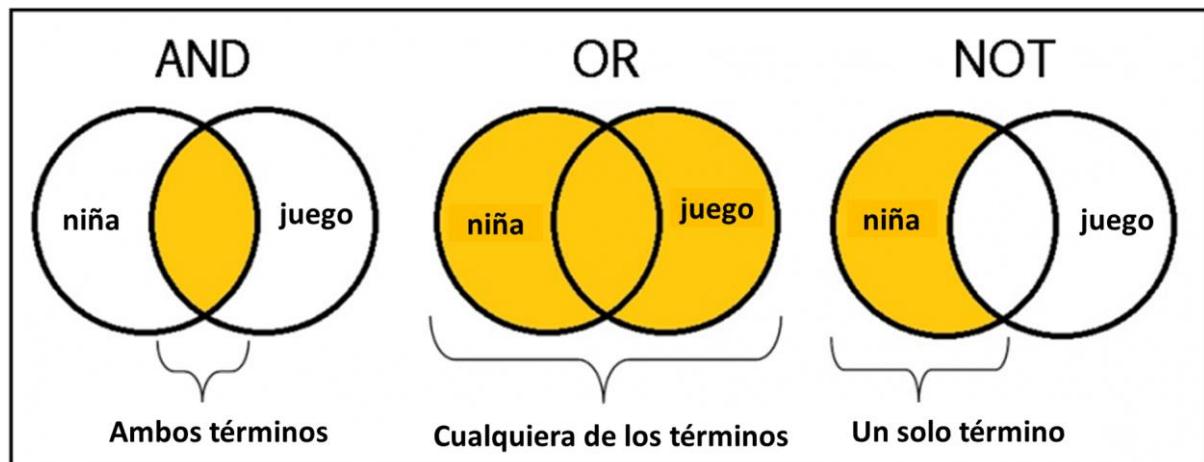




CLAVES BOOLEANAS

Tp N°8 de técnicas de procesamiento del habla



Alumno = Nicolas Mesquiatti

Materia = Técnicas procesamiento del habla

Modelo de Claves Booleanas

Consigna 1:

Crea un programa en Python que permita al usuario realizar búsquedas booleanas (AND, OR, NOT) en un conjunto de documentos utilizando NLTK.

◆ Pasos a seguir:

-Se tienen 5 documentos con información sobre inteligencia artificial y aprendizaje automático.

"doc1": "La inteligencia artificial está revolucionando la tecnología.",

"doc2": "El aprendizaje automático es clave en la inteligencia artificial.",

"doc3": "Procesamiento del lenguaje natural y redes neuronales.",

"doc4": "Las redes neuronales son fundamentales en deep learning.",

"doc5": "El futuro de la IA está en el aprendizaje profundo."

- -Implementa una función que tokenice y limpie los documentos.
- -Crea un índice invertido (asocia cada palabra clave a los documentos en los que aparece).

Pasos para seguir:

- -Permite que el usuario ingrese una consulta booleana y devuelva los documentos relevantes.

Consultas de Prueba

Ingresar una consulta booleana (o 'salir' para terminar): inteligencia AND artificial

 Documentos encontrados: {'doc1', 'doc2'}

Ingresar una consulta booleana (o 'salir' para terminar): redes OR aprendizaje

 Documentos encontrados: {'doc2', 'doc3', 'doc4', 'doc5'}

Ingresar una consulta booleana (o 'salir' para terminar): inteligencia NOT automático

 Documentos encontrados: {'doc1'}

Ingresar una consulta booleana (o 'salir' para terminar): salir

Consigna 2:

Crea un programa en Python que permita al usuario realizar búsquedas booleanas (AND, OR, NOT) en un conjunto de documentos utilizando NLTK.

◆ Pasos para seguir:

-Se tienen 5 documentos con información sobre civilizaciones antiguas.

Se tienen 5 documentos con información sobre civilizaciones antiguas.

"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.

-Implementa una función que tokenice y limpie los documentos.

-Crea un índice invertido (asocia cada palabra clave a los documentos en los que aparece)

-Permite que el usuario ingrese una consulta booleana y devuelva los documentos relevantes.

Consultas de Prueba

Ingrese una consulta booleana (o 'salir' para terminar): egipcios AND pirámides

 **Documentos encontrados: {'doc1'}**

Ingrese una consulta booleana (o 'salir' para terminar): escritura OR astrónomos

 **Documentos encontrados: {'doc1', 'doc3', 'doc5'}**

Ingrese una consulta booleana (o 'salir' para terminar): romano NOT griegos

 **Documentos encontrados: {'doc2'}**

1) Se importa NLTK para tokenizar textos y eliminar stopwords (palabras como “la”, “y”, “de”).

2) Se crean dos grupos de documentos:

- "ia" → textos sobre inteligencia artificial.
- "civilizaciones" → textos sobre civilizaciones antiguas.

```

colecciones = {
    "ia": {
        "doc1": "La inteligencia artificial está revolucionando la tecnología.",
        "doc2": "El aprendizaje automático es clave en la inteligencia artificial.",
        "doc3": "Procesamiento del lenguaje natural y redes neuronales.",
        "doc4": "Las redes neuronales son fundamentales en deep learning.",
        "doc5": "El futuro de la IA está en el aprendizaje profundo."
    },
    "civilizaciones": {
        "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."
    }
}

```

3) Realizamos una función para limpiar y tokenizar

Convierte el texto a minúsculas, también elimina palabras comunes, puntuación y acentuación

```

def limpiar_tokenizar(texto):
    texto = quitar_tildes(texto.lower()) # quitar tildes y minúsculas
    tokens = word_tokenize(texto)
    stop_words = set(stopwords.words('spanish'))
    tokens_limpios = [
        palabra for palabra in tokens
        if palabra not in stop_words and palabra not in string.punctuation
    ]
    return set(tokens_limpios)

```

¿Porque eliminar acentuación?

Acá hice una consulta de inteligencia NOT automático

```

Ingrrese una consulta booleana (o 'salir' para terminar): inteligencia NOT automático
📄 Documentos encontrados: {'doc2', 'doc1'}
Ingrrese una consulta booleana (o 'salir' para terminar): inteligencia NOT automático
📄 Documentos encontrados: {'doc1'}

```

Me tendría que haber devuelto doc1 como lo hizo en la segunda consulta, pero no lo hizo porque 'automático' no lo reconoce, ya que en los documentos aparece con acento 'automático' si me lo reconoció, por eso es que cree una función específica para quitar los tildes a la hora de la consulta

```

def quitar_tildes(texto):
    return ''.join(
        c for c in unicodedata.normalize('NFD', texto)
        if unicodedata.category(c) != 'Mn'
    )

```

4) Construimos el índice invertido

Un **índice invertido** es una estructura de datos que permite saber **en qué documentos aparece cada palabra clave**

```
def construir_indice(documentos):
    indice = {}
    for doc_id, texto in documentos.items():
        tokens = limpiar_tokenizar(texto)
        for token in tokens:
            if token not in indice:
                indice[token] = set()
            indice[token].add(doc_id)
    return indice
```

5) Luego una función para buscar documentos por palabras, Busca en el índice qué documentos contienen una palabra. Si no está, devuelve un conjunto vacío.



```
# === 4. Función para obtener documentos de una palabra ===

def obtener_docs(termino, indice):
    return indice.get(termino, set())
```

6) Procesar la consulta booleana del usuario, En el que realiza consultas del siguiente estilo

Puede interpretar lo que escribe el usuario ej. operaciones sobre

| Operador | Acción |
|-------------|--|
| AND | Intersección (A \cap B) – documentos con ambas palabras. |
| OR | Unión (A \cup B) – documentos con al menos una. |
| NOT | Diferencia (A $-$ B) – documentos con la primera pero no la segunda. |
| NOT palabra | Todos los documentos menos los que tienen esa palabra. |

```

def procesar_consulta(consulta, indice, documentos):
    consulta = quitar_tildes(consulta.lower())
    palabras = consulta.split()

    def normalizar(palabra):
        return stemmer.stem(palabra)

    if len(palabras) == 3:
        op1, operador, op2 = map(normalizar, palabras)
        docs1 = obtener_docs(op1, indice)
        docs2 = obtener_docs(op2, indice)
        if operador == 'and':
            return docs1 & docs2
        elif operador == 'or':
            return docs1 | docs2
        elif operador == 'not':
            return docs1 - docs2
    elif len(palabras) == 2 and palabras[0] == 'not':
        docs = obtener_docs(normalizar(palabras[1]), indice)
        return set(documentos.keys()) - docs
    else:
        return obtener_docs(normalizar(palabras[0]), indice)

```

7) Finalmente, se realiza la interacción con el usuario: en el que,

Muestra **los temas disponibles** para buscar: "ia" o "civilizaciones".

Pide **al usuario que elija un tema válido** y crea el índice invertido para ese conjunto de documentos y El usuario escribe una consulta booleana (como palabra1 AND palabra_2). Se muestra qué documentos contienen esa combinación.

```

print("💡 Buscador Booleano")
print("Temas disponibles: ia, civilizaciones")

tema = input("Seleccioná el tema para buscar: ").lower()
while tema not in colecciones:
    tema = input("Tema no válido. Elegí 'ia' o 'civilizaciones': ").lower()

documentos = colecciones[tema]
indice_invertido = construir_indice(documentos)

print(f"Índice creado para tema: {tema}")

# Bucle principal
while True:
    entrada = input("Ingrese una consulta booleana (o 'salir' para terminar): ")
    if entrada.lower() == 'salir':
        break
    resultado = procesar_consulta(entrada, indice_invertido, documentos)
    print("📄 Documentos encontrados:", resultado)

```

CONSULTAS SOBRE LA CONSIGNA UNO (documento de ia)

```

Temas disponibles: ia, civilizaciones
Seleccioná el tema para buscar: ia
Índice creado para tema: ia
Ingrese una consulta booleana (o 'salir' para terminar): inteligencia and artificial
📄 Documentos encontrados: {'doc2', 'doc1'}
Ingrese una consulta booleana (o 'salir' para terminar): redes or aprendizaje
📄 Documentos encontrados: {'doc5', 'doc2', 'doc4', 'doc3'}
Ingrese una consulta booleana (o 'salir' para terminar): inteligencia not automatico
📄 Documentos encontrados: {'doc1'}
Ingrese una consulta booleana (o 'salir' para terminar): redes or neuronales
📄 Documentos encontrados: {'doc4', 'doc3'}
Ingrese una consulta booleana (o 'salir' para terminar): 

```

CONSULTAS SOBRE LA CONSIGNADOS (documento de civilizaciones)

```

Temas disponibles: ia, civilizaciones
Seleccioná el tema para buscar: civilizaciones
Índice creado para tema: civilizaciones
Ingrese una consulta booleana (o 'salir' para terminar): egipcios and piramides
📄 Documentos encontrados: {'doc1'}
Ingrese una consulta booleana (o 'salir' para terminar): escritura or astronemos
📄 Documentos encontrados: {'doc1', 'doc3', 'doc5'}
Ingrese una consulta booleana (o 'salir' para terminar): romano not griegos
📄 Documentos encontrados: {'doc2'}
Ingrese una consulta booleana (o 'salir' para terminar): civilizacion or bases
📄 Documentos encontrados: {'doc4', 'doc2'}
Ingrese una consulta booleana (o 'salir' para terminar): 

```

En la consulta de “ROMANO NOT GRIEGOS” anteriormente no me devolvía nada, ya que busca romano y griegos y no encuentra, ya que en los doc. no están especificados de esa manera, sino que aparecen como (“romana o Grecia”), luego le aplique la lematización de mejor forma y funciono.