Creación de un Sistema de Recuperación de Información

Objetivo General

El objetivo de esta práctica es desarrollar un sistema completo de Recuperación de Información compuesto por:

- Un **backend** implementado con **FastAPI**, encargado del procesamiento de documentos y la indexación.
- Un **frontend** desarrollado en **ReactJS**, que permita enviar consultas y mostrar resultados.
- Una **colección documental superior a 10 GB**. Si se obtenen mediante técnicas de crawling (araña web) será puntuado aparte.

Los estudiantes deberán entregar:

- Memoria explicativa del proceso completo.
- Código del backend y frontend.
- Enlace a un repositorio de GitHub con todo el proyecto.

1. Creación del Backend con FastAPI

Para iniciar el proyecto, se debe crear un entorno virtual:

```
python -m venv venv
source venv/bin/activate   (Linux/Mac)
venv\Scripts\activate      (Windows)
```

Instalación de FastAPI y Uvicorn:

```
pip install fastapi uvicorn nltk
```

1.1 Ejemplo Hello World en FastAPI

Archivo `main.py`:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def hello():
    return {"message": "Hello World from FastAPI!"}
```

Ejecución del servidor:

```
uvicorn main:app --reload
```

1.2 Procesos del Sistema de Recuperación de Información

Durante la práctica se implementarán los siguientes pasos, tal como se explicó en clase:

1. **Colección documental (Crawler / Araña)** Obtención automática de más de **10 GB de documentos** procedentes de la web. (Opcional)
2. **Análisis léxico** Revisión inicial del corpus para normalizar codificación, formato y estructura.
3. **Tokens (términos)** Segmentación del texto en unidades léxicas.
4. **Eliminación de palabras vacías** Eliminación de stopwords según el idioma del corpus.
5. **Tokens con significado** Primer filtrado semántico.
6. **Lematización y stemming** Reducción de términos a su forma base o raíz morfológica.
7. **Términos (raíces morfológicas)** Obtención del vocabulario básico del sistema.
8. **Ponderación de términos** Aplicación de métodos como TF, IDF o TF-IDF.
9. **Términos ponderados** Representación numérica de cada documento.
10. **Selección de términos** Selección de los términos relevantes para la recuperación.
11. **Términos de indexación** Vocabulario final para el índice.
12. **Indexación** Construcción del índice invertido.
13. **Índice** Estructura final que permite la recuperación eficiente.

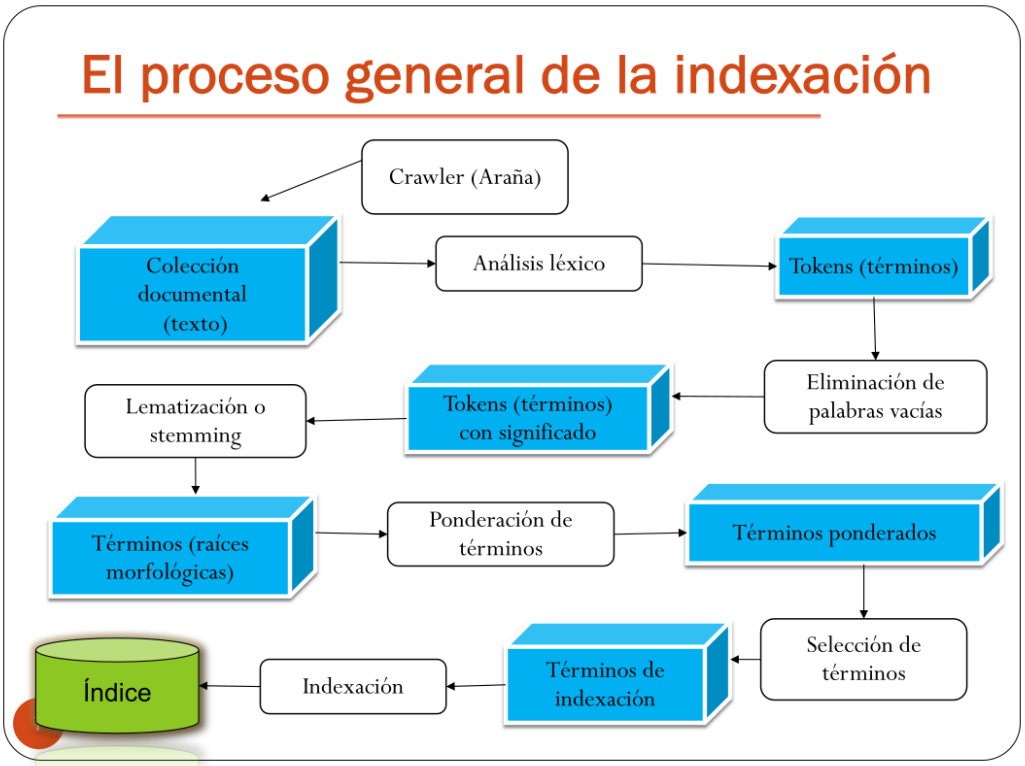

Figura 1: Proceso de indexación

1.3 Cabeceras del Backend para Procesamiento RI

(Solo se incluyen las cabeceras, no la implementación completa)

```
@app.post("/crawl")
def crawl_web(url: str, depth: int):
    pass

@app.post("/lexical_analysis")
def lexical_analysis(document: str):
    pass

@app.post("/tokenize")
def tokenize(document: str):
    pass

@app.post("/remove_stopwords")
def remove_stopwords(tokens: list):
    pass

@app.post("/lemmatize")
def lemmatize(tokens: list):
    pass

@app.post("/weight_terms")
def weight_terms(terms: list):
    pass

@app.post("/select_terms")
def select_terms(weighted_terms: list):
    pass

@app.post("/index")
def index_documents(selected_terms: list):
    pass

@app.get("/search")
def search(query: str):
    pass
```

2. Creación del Frontend en ReactJS

2.1 Instalación y Creación del Proyecto

Para crear un proyecto React:

```
npx create-react-app buscador-ri
cd buscador-ri
npm start
```

2.2 Ejemplo Hello World en ReactJS

Archivo `src/App.js`:

```
function App() {
  return (
    <div>
      <h1>Hello World desde ReactJS</h1>
    </div>
  );
}

export default App;
```

2.3 Comunicación con el Backend

El frontend debe permitir enviar consultas al backend mediante `fetch()`:

```
fetch("http://localhost:8000/search?query=" + consulta)
```

3. Selección de la Colección Documental

Cada estudiante podrá elegir su propia colección, pero deberá tener **al menos 10 GB**. Por ejemplo:

- Noticias de medios digitales.
- Artículos de Wikipedia obtenidos mediante APIs o crawling.
- Conversaciones o foros públicos con licencia abierta.
- Documentación técnica, manuales, libros de dominio público.
- Repositorios de texto como Common Crawl, Kaggle o Project Gutenberg.

Se debe justificar:

- El origen de los datos.
- Los métodos de recopilación.
- La estructura final del corpus.

4. Entrega Final

El alumno deberá entregar los siguientes items:

1. **Memoria** detallada incluyendo:
   - Proceso completo del Sistema de Recuperación de Información.
   - La arquitectura del sistema.
   - Capturas del frontend.
   - Resultados de algunas búsquedas.
2. **Código fuente** del backend y frontend.
3. **Repositorio GitHub** con:
   - Código completo.
   - Documentación.
   - Datos o ejemplo de corpus.
   - Instrucciones de instalación.