Orientación del Proyecto de Sistemas de Información 2016-2017

1 Introducción

El proyecto tiene como objetivo crear un sistema de recuperación de información que funcione de forma local sobre un directorio. El sistema de recuperación de información constará de cuatro módulos fundamentales, que se comunican entre sí a través de documentos en formato JSON.

Cada módulo debe ejecutarse de manera independiente, "por ejemplo", para un proyecto implementado en Python, cada módulo debe ser un archivo diferente; para un proyecto implementado en C#, cada módulo debe ser una aplicación de consola diferente. Se permite utilizar archivos, o implementar bibliotecas de clases adicionales con funcionalidades comunes o utilitarias que sean utilizados por el resto del sistema. Se deben procesar documentos en formato txt y pdf, opcionalmente los estudiantes pueden procesar otros. Estas funcionalidades adicionales deben estar debidamente documentadas en el reporte del proyecto.

Cada módulo del proyecto recibirá los datos de entrada en formato JSON. En cada caso el documento JSON incluye un campo `"action"` que indica cuál de las posibles acciones disponibles para ese módulo debe ejecutarse, así como campos extras para los datos correspondientes a dicha acción. En la descripción de cada módulo se definen las acciones posibles y los formatos exactos de entrada y salida.

2 Módulos del Proyecto

A continuación se listan los módulos obligatorios del proyecto. Para cada módulo se define la interfaz JSON de comunicación con el resto del sistema. Se permite escoger cualquier tecnología, framework, lenguaje de programación, y/o herramientas adicionales.

2.1 Módulo de Interfaz de Usuario

Se debe implementar una interfaz de usuario muy sencilla, que permita escoger un directorio donde se encuentran los documentos a indizar, escribir y ejecutar una consulta, y mostrar la lista de documentos ordenada por relevancia.

Este módulo genera un JSON con dos posibles valores de `"action": "build"` y `"query"`.

Para inicializar el sistema se emplea `"action": "build"`, que tiene el siguiente formato:

```
{
    "action": "build",
    "path":  "/directorio/donde/se/encuentran/los/documentos",
}
```

Este JSON será utilizado por el módulo de recuperación para construir el modelo y el módulo de índices para construir los índices.

Para realizar una consulta se emplea `"action": "query"`, que tiene el siguiente formato:

```
{
    "action": "query",
    "query": "consulta a realizar en lenguaje natural",
    "count": "cantidad máxima de documentos a recuperar",
}
```

Este JSON será utilizado por el módulo de procesamiento de texto para iniciar el proceso de consulta.

Al ser procesada la consulta por los módulos correspondientes, este módulo recibirá un JSON con `"action": "report"` que tiene la siguiente estructura:

```
{
    "action": "report",
    "sucess": "true",
    "results": [
        {
            "document": "nombre del documento 1.pdf",
            "match": "valor mayor o igual que 0 que determina la relevancia",
        },
        {
            "document": "nombre del documento 2.pdf",
            "match": "valor mayor o igual que 0 que determina la relevancia",
        },
        // ... Resto de los documentos hasta un máximo de 'count'
        // ... definido en el JSON de salida.
    ],
}
```

El módulo debe procesar este JSON y reportar en la interfaz la lista ordenada por relevancia de los documentos recuperados con vínculos a los documentos originales.

2.2 Módulo de Procesamiento de Texto

Este módulo recibe un JSON con `"action": "process"` para procesar un documento, con el formato:

```
{
    "action": "process",
    "data": "texto plano del documento/consulta a procesar",
}
```

Devuelve un JSON con la lista de términos, después de procesados:

```
{
    "terms": ["termino1", "termino2", /* ... */],
}
```

Este módulo debe realizar las siguientes operaciones en cada documento:

- Análisis léxico
- Eliminación de *stopwords*
- *Stemming*

Se premiarán operaciones de procesamiento más avanzadas.

2.3 Módulo de Modelado

Este módulo consiste en la implementación de uno de los modelos de recuperación de información estudiados en clase. Como mínimo se debe implementar uno de los siguientes modelos:

- Vectorial
- Booleano
- Probabilístico

Se premiarán implementaciones de modelos más avanzados, o modificaciones a estos modelos que mejoren su rendimiento.

Este módulo recibe un JSON con `"action": "build"`, para construir el modelo de recuperación; o un JSON con `"action": "query"` para realizar la consulta. En el caso de `"action": "query"` se devuelve un JSON con `"action": "report"` con los datos de la respuesta. En ambos casos el formato se especifica en la sección 2.1.

2.4 Módulo de Índices

Este módulo consiste en una implementación eficiente de un diccionario, donde se asocia a cada término (llave del diccionario) un valor (estructura JSON). Se puede implementar y consultar el índice completamente en RAM, sin necesidad de contar con un B-Tree u otra estructura de datos para almacenar los índices en disco.

Para construir el índice recibe un JSON con `"action": "create"`, y el siguiente formato:

```
{
    "action": "create",
    "data": [
        {
            "key": "llave 1",
            "value": // ... Un objeto JSON
        },
        {
```

```
            "key": "llave 2",
            "value": // ... Un objeto JSON
        },
        // ... Resto de las llaves
    ],
}
```

Note que el valor asociado a una llave puede ser cualquier objeto JSON, tan complicado como requiera el modelo correspondiente. El formato de los valores es abstracto para el índice, y puede ser almacenado como texto, siempre que sea recuperado exactamente de la misma forma en que fue introducido.

Por ejemplo, para el MRI Vectorial, un posible formato para el índice sería:

```
{
    "action": "create",
    "data": [
        {
            "key":
            "término1",
            "value": {
                "idf": 14.513,
                "documents": [
                    {
                        "document": "nombre del documento 1.pdf",
                        "tf": 0.547,
                    },
                    {
                        "document": "nombre del documento 2.pdf",
                        "tf": 0.189,
                    },
                    // ... Resto de los documentos
                ],
            },
        },
        {
            "key":
            "término2",
            "value": {
                "idf": 3.513,
                "documents": [
                    {
                        "document": "nombre del documento 3.pdf",
                        "tf": 0.365,
                    },
                    {
                        "document": "nombre del documento 4.pdf",
                        "tf": 0.124,
                    },
                    // ... Resto de los documentos
                ],
            },
        },
```

```
        // ... Resto de las llaves
        ],
}
```

Note que se almacena en cada valor todos los datos que necesita el modelo para funcionar, en este caso, el valor calculado de "idf" para cada término, así como el valor "tf" para cada uno de los documentos donde aparece. El índice simplemente almacenará todo el objeto JSON asociado a "value" como texto, y lo devolverá exactamente de la misma forma en que fue introducido. Es responsabilidad de la implementación de cada modelo utilizar el índice de forma eficiente, almacenando los datos necesarios.

Este módulo recibe además un JSON con "action": "add" y el formato:

```
{
    "action": "add",
    "key": "nueva llave",
    "value": // ... Un objeto JSON
}
```

que adiciona una nueva llave al índice, con un valor asociado. Igualmente, este valor puede ser un objeto JSON tan complicado como requiera el modelo.

o un JSON con "action": "update" que actualiza un documento existente, el formato es:

```
{
    "action": " update ",
    "key": "nueva llave",
    "value": // ... Un objeto JSON
}
```

o un JSON con "action": "delete" que elimina un documento existente, el formato:

```
{
    "action": "delete",
    "key": "nueva llave",
}
```

Recibe además un JSON con "action": "get" para consultar el índice, con el siguiente formato:

```
{
    "action": "get",
    "key": "llave",
}
```

En este caso debe devolver un JSON con el valor asociado a dicha llave y un indicador de si la operación terminó correctamente o no:

```
{
    "sucess": "true",
    "value": // ... El objeto JSON almancenado para esa llave
}
```

3 Evaluación del Sistema

Se debe implementar además las medidas de evaluación estudiadas en clase:

- Precisión
- Recobrado
- F-Medida
- E-Medida
- R-Precisión

Con estas medidas y un conjunto de documentos de prueba diseñado por usted, debe evaluar la calidad de su implementación, y documentar los resultados de esta evaluación en su reporte. Se deben brindar funcionalidades en la interfaz gráfica para ejecutar las medidas.

4 Funcionalidades Opcionales

A continuación se listan funcionalidades opcionales que recibirán bonificaciones:

- Expansión de consultas.
- Recomendación de documentos.
- Retroalimentación.

Para implementar estas funcionalidades, se deben definir módulos con una interfaz JSON, similar a los utilizados en el proyecto. Estos módulos deben ser independientes entre sí. Se debe documentar la interfaz y la implementación de estos módulos adicionales. Se tendrá en cuenta en la bonificación la calidad del diseño de estos módulos, su extensibilidad, y la facilidad con que se integran con los módulos iniciales del proyecto.