Estructuras de Datos

Universidad Nacional de Tres de Febrero

Trabajo Práctico Nº 2

Fechas de entrega

Comisión de los días martes: 5/11/5019

Comisión de los días jueves: 7/11/2019

El objetivo de este trabajo práctico, es poner en práctica y sobre todo en código, los conocimientos adquiridos durante la cursada, para lo cual vamos a construir un sistema de indexación y recuperación de la información orientado a recolectar e indexar noticias publicadas en la web.
En concreto en este trabajo se deberá:

1. Recolectar noticias de los principales medios de comunicación de Argentina
2. Construir un índice (sin compresión) sobre el cuerpo de noticias recolectadas e implementar un sistema de búsqueda que soporte consultas booleanas.
3. Construir un índice comprimido, sobre el mismo cuerpo de noticias usando la técnica de codificación de longitud variable
4. Escribir un informe.

Cuerpo de noticias

La primera parte consiste en recopilar una buena cantidad de noticias para armar nuestro corpus para analizar.
Se deberá desarrollar este módulo cuanto antes a fin de recolectar noticias de la web. El tamaño mínimo del cuerpo de noticias aceptable será de 200 Mbytes
Nos vamos a enfocar en las fuentes web, que suelen publicar sus noticias, clasificadas en diferentes categorías o canales, lo que se conoce como redifusión web o sindicación web (https://es.wikipedia.org/wiki/Redifusión_web). Generalmente la información se transmite en formato XML siendo los principales esquemas RSS (https://es.wikipedia.org/wiki/RSS) y Atom (https://es.wikipedia.org/wiki/Atom_(formato_de_redifusión)). Estos esquemas proveen de cierta estructura a la información: título, descripción, links, fecha de publicación, etc.
Para la primera parte que consiste en recopilar las noticias se deberá almacenar información de al menos los siguientes canales: últimas noticias, política, sociedad, economía, internacionales, de al menos cinco medios (diarios o agencias de noticias) argentinos, por ejemplo: Telam, Clarín, La Nación, Diario Popular, La Gaceta, La Voz del Interior, etc.
Por ejemplo en la página http://www.telam.com.ar/rss se encuentran las URLs para acceder a los siguientes canales:

- Últimas Noticias
- Política
- Economías
- Sociedad
- Deportes
- Policiales
- Internacional
- Latinoamérica
- Cono Sur
- Provincias
- Agropecuario
- Tecnología
- Cultura
- Espectáculos
- Turismo
- Salud
- Educación
- Redes

En este trabajo sólo nos concentraremos en canales RSS

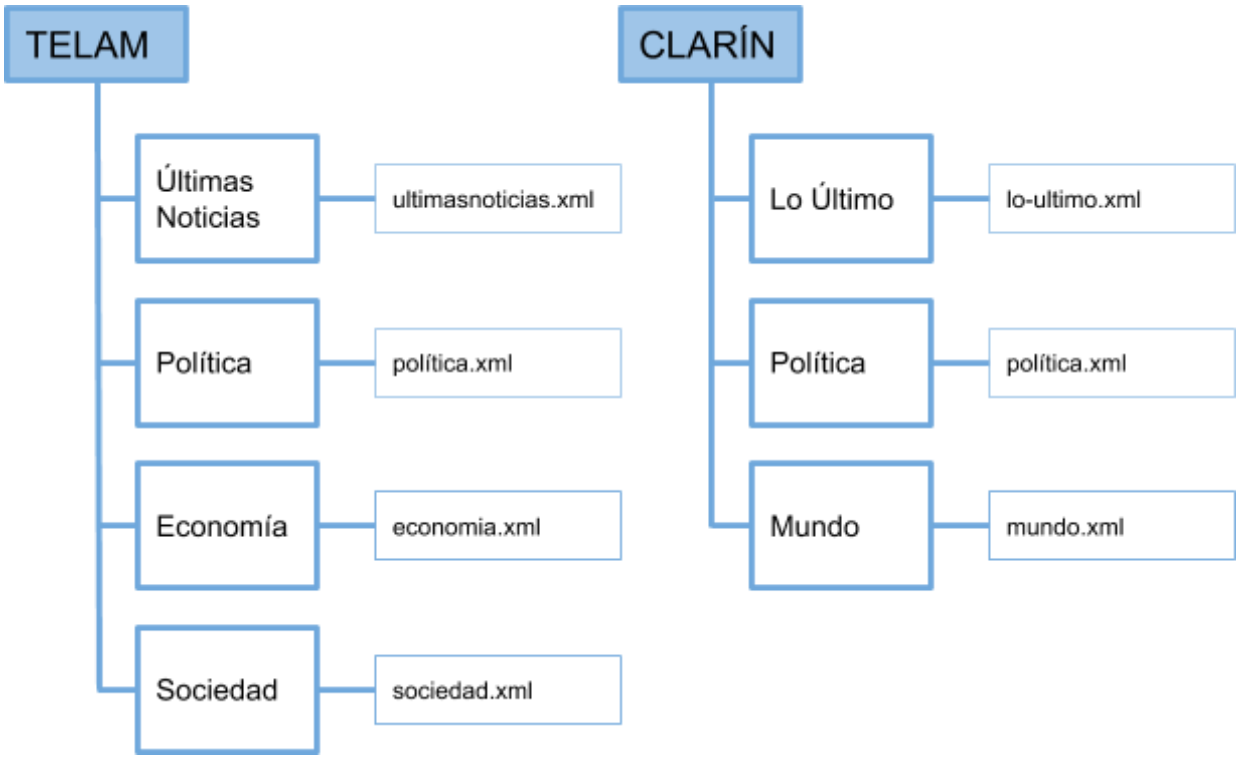
Archivo de configuración

En un archivo de configuración se deberá proveer las URLs para acceder a todos los canales utilizados. Se deberá utilizar el módulo de python configparser para leerlo.
El nombre del archivo de configuración deberá ser "config.ini"
Ejemplo de archivo de configuración

```
[DEFAULT]
query_interval = 600
tmp = ./temp
output = ./salida
...
[TELAM]
url_base = https://www.telam.com.ar
Últimas Noticias = /rss2/ultimasnoticias.xml
Política = /rss2/politica.xml
Economía = /rss2/economia.xml
Sociedad = /rss2/sociedad.xml
...
[CLARÍN]
url_base = https://www.clarin.com
Lo Último = /rss/lo-ultimo/
Política = /rss/politica/
Mundo = /rss/mundo/
...
```

Estructura del cuerpo de noticias

Por cada medio/canal se deberá almacenar las noticias en el formato xml, conservando el formato original (RSS o Atom). Es decir por cada medio/canal se tendrá un gran RSS o Atom con todas las noticias recolectadas



Esta primera parte que recolecta los feeds deberá consultar las URLs a un intervalo predeterminado y configurable, por ejemplo cada 10 minutos (`query interval`).

Archivos xml

Los datos recolectados se almacenarán en archivos xml, conservando del formato RSS sólo los `<item>` completos
Como una lectura puede traer varias noticias que ya fueron leídas anteriormente, para evitar llenar archivos con noticias repetidas se deberá controlar los ***títulos*** y ***fechas de publicación*** para determinar si son noticias repetidas.

Entregables

- Código Fuente
- Cuerpo de noticias comprimido en un archivo zip

Índice sin comprimir y módulo de búsqueda

En este punto se deberá crear un índice invertido sobre el cuerpo de noticias recolectado e implementar un módulo que permita realizar consultas booleanas sobre el índice.

Para construir el índice se deberá implementar el algoritmo ***Blocked Sort-Base Indexing (BSBI)*** descrito en la *Sección 4.2* del libro *Introduction to Information Retrieval* (Se recomienda una lectura completa de la sección del libro para aclarar conceptos)

- **Algoritmo BSBI**: *Blocked sort-based indexing* (simple):
  - Segmenta la colección en partes de igual tamaño.
  - Ordena los pares (id_termino, id_doc) de cada parte en memoria.
  - Almacena en disco los resultados intermedios ordenados.
  - Intercala todos los resultados intermedios en un índice final.
- El algoritmo genera pares (id_termino, id_doc) y los acumula en la memoria hasta que completa un bloque completo (parse_next_block). El tamaño del bloque se elige para que sea suficientemente grande pero al mismo tiempo se pueda hacer un sort en memoria cómodamente.
- El bloque se invierte y se almacena en disco:
  - Ordenar en memoria los pares (id_termino, id_doc).
  - Generar listas de aparición con el mismo id_termino para simplificar (guardar la lista de id_doc).
  - Guardar en disco el índice invertido.
- Intercalar todos los índices invertidos en un índice final.

> *"Para construir un índice, primero hacemos un recorrido por la colección ensamblando todos los pares term-docID. Luego ordenamos los pares con el término como clave dominante y docID como clave secundaria. Finalmente, organizamos los docID para cada término en una lista de publicaciones y calculamos estadísticas como término y frecuencia de documentos. Para colecciones pequeñas, todo esto se puede hacer en la memoria."*
>
> *"Para hacer que la construcción del índice sea más eficiente, representamos los términos como termID (en lugar de cadenas), donde cada term ID es un número de serie único. Podemos construir el mapeo de términos a termIDs sobre la marcha mientras procesamos la colección. Del mismo modo, también representamos documentos como docID (en lugar de cadenas)."*

Por lo tanto el módulo de construcción del índice deberá contar con una clase que permita mapear los términos (cadena de caracteres) a termID (número entero) y documentos (cadena de caracteres) a docID (número entero)

Lista de documentos como bytearrays

En disco se deben almacenar la lista de documentos que corresponde a cada término, para que la gestión de estas listas en disco sea eficiente, se deberán almacenar como bytearrays. La clase `UncompressedPostings` a continuación, contiene los métodos estáticos `encode(postings_list)` y `decode(encoded_postings_list)` para convertir una lista de enteros en una lista de bytearrays y viceversa

```
class UncompressedPostings:
```

```
    @staticmethod
    def encode(postings_list):
        """Encodes postings_list into a stream of bytes

        Parameters
        ----------
        postings_list: List[int]
            List of docIDs (postings)

        Returns
        -------
        bytes
            bytearray representing integers in the postings_list
        """
        return array.array('L', postings_list).tobytes()

    @staticmethod
    def decode(encoded_postings_list):
        """Decodes postings_list from a stream of bytes

        Parameters
        ----------
        encoded_postings_list: bytes
            bytearray representing encoded postings list as output by encode
            function

        Returns
        -------
        List[int]
            Decoded list of docIDs from encoded_postings_list
        """

        decoded_postings_list = array.array('L')
        decoded_postings_list.frombytes(encoded_postings_list)
        return decoded_postings_list.tolist()
```

Índice invertido sobre el disco

El índice invertido se debe construir paso a paso, a fin de simplificar tomaremos cada agencia de noticias / medio de comunicación como un bloque. El algoritmo solo debe trabajar con un único bloque en memoria, ordenar, invertir y guardar en disco.
Finalmente intercalar todos los bloques invertidos.
Cada bloque queda identificado por el directorio que contiene las noticias. Siguiendo el ejemplo anterior:
Bloque 1: Telam
Bloque 2: Clarín
…

Término:

A los efectos de este trabajo práctico, consideramos cada término como cada una de las palabras que forman parte tanto del título como del cuerpo de cada noticia (cada `<item>` del archivo xml)

Documentos:

Los documentos se deberán identificar por el **medio + ”-” + canal + ”-” + título de la noticia + ”-” + fecha de publicación.** Por ejemplo:

Telam-Política-Macri recibe a los subcampeones mundiales de básquetbol 2019-17092019

El índice invertido consistirá en un diccionario cuya clave sea el termID correspondiente y cuyo valor será una tupla con metadata útil para gestionar la lista de documentos. La tupla deberá contener los 3 datos siguientes:

- posición_inicial_archivo: es la posición en bytes del inicio de la lista de documentos donde aparece el término en cuestión, dentro del archivo que contiene todo el índice invertido
- cant_de_documentos: es la cantidad de documentos que corresponden este término
- long_en_bytes[1]: es la longitud del byte con que se codificaron los docID.

En resumen, luego de construido el índice invertido, se deberían tener los siguientes archivos en disco:
indice_invertido.ii : índice invertido como se describió anteriormente.

También se deberá guardar en disco los diccionarios que permiten pasar de términos a termID y de documentos a docID.

Todos los términos deberán normalizarse y usar un stemmer antes de procesarlos para obtener el termID.

Persistencia

El módulo de creación del índice invertido, deberá ofrecer al usuario las siguientes posiblidades:
1- Crear un índice invertido a partir de la estructura de directorio y archivos xml
2- Guardar en disco todo el índice invertido (todas las estructuras que permitan recuperarlo nuevamente de disco
3- Cargar en memoria un índice invertido previamente salvado.

[1] Para consultar el tamaño en bytes de un objeto se puede usar el método sys.getsizeof()

Módulo de búsqueda:

Este módulo permite que un usuario ingrese una lista de palabras, separadas por blancos y debe devolver la lista de documentos (medio, canal, título de la noticia, fecha de publicación) donde aparece cada palabra

Índice comprimido

En este punto se deberá comprimir la lista de apariciones codificando los saltos entre docID con un código de longitud variable.
El índice invertido de los puntos anteriores deberá funcionar con la lista de apariciones sin comprimir y comprimida.

Se recomienda el siguiente sitio para más detalles sobre la manipulación de bits en Python:
https://wiki.python.org/moin/BitManipulation

Menú

El sistema deberá presentar un menú simple con al menos las siguientes opciones:

- Recolectar noticias
- Crear Índice Invertido
- Comprimir Lista de Apariciones
- Realizar búsquedas

Pautas de trabajo

Podrán trabajar en grupos de hasta 3 personas.

Presentación

Se deberá subir informe en Jupyter con diagrama de clases, decisiones de diseño, conclusiones, código fuente (documentado), casos de prueba y cuerpo de noticias a un drive virtual (por ejemplo google drive, dropbox, etc.) y compartir el enlace con los permisos suficientes para poder descargar y probar el código.
El mail con el enlace debe indicar en el cuerpo todos los integrantes del grupo y el asunto debe ser EDD-TP2-(martes/jueves)

Fechas de Entrega:

Turno de los martes: 05/11/2019 Turno de los jueves: 07/11/2019