Laboratorio 2:

Programación Orientada a Objetos

Paradigmas 2024 - FAMAF

En este laboratorio vamos a implementar en Java un lector automático de *feeds*, como aplicación de consola. Para ello, utilizaremos las herramientas del paradigma orientado a objetos.

Instalación

1. Instalar Java Development Kit (JDK) y Java Run Environment (JRE). Para el laboratorio siguiente van a necesitar la versión 17, con lo que recomendamos utilizar esta versión.
   En un Ubuntu 22 basta con

   ```
   sudo apt install openjdk-17-jdk openjdk-17-jre
   ```

   Pueden usar un IDE pero cuidado, nosotros correremos el código con el Makefile provisto, por lo que no se acostumbren a las mieles de la IDE. Es decir, deben entender el proceso de compilación de Java.

   El repositorio contiene un archivo `.jar` que es una dependencia del proyecto que debería permanecer en el repo, tengan cuidado de no agregalo al `.gitignore`.

2. Clonar de Bitbucket el skeleton provisto por la cátedra: grupo00_lab02_2024

3. Push del skeleton al repositorio del grupo: `grupo??_lab02_2024`

Funcionalidad

El software debe traer los feeds de noticias en XML y parsearlos en una estructura de datos adecuada. Cada feed puede ser interpretado como una colección de artículos. De ser solicitado por el usuario, se imprimirán en pantalla los atributos (título, descripción y fecha) de cada uno de los artículos que se hayan procesado. Los

feeds de noticias que seleccionamos corresponden a diarios digitales argentinos pero se pueden incorporar otros.

Adicionalmente el programa debe poder computar entidades nombradas de los artículos. Estas son entes del mundo real que pueden ser denotados con un nombre propio, tales como personas, lugares, organizaciones, productos, etc. Por ejemplo, en el artículo "El presidente de los Estados Unidos, Joe Biden, visitó la ciudad de Nueva York" las entidades nombradas son “Estados Unidos”, "Joe Biden" y "Nueva York". (Pueden leer más en la wiki, o preguntarle a Laura, experta en el tema).

En nuestra aplicación, el cómputo de entidades nombradas consistirá en las siguientes tareas:

1. **Identificar** las entidades nombradas en un artículo. La selección de las entidades nombradas se hará a partir de heurísticas y el usuario podrá elegir entre diferentes heurísticas. Por ejemplo, una heurística podría ser "todas las palabras que empiezan con mayúscula son entidades nombradas", aunque tendría muchos *falsos positivos*. Sería esperable de una buena heurística que detecte las menciones a “Caputo” en la sección *economía* de Página12, pero no incluya palabras como “Impulso”. Desde la cátedra les proveeremos una heurística sencilla y ustedes deberán programar otras.
2. **Clasificar** las entidades nombradas en categorías predefinidas y etiquetarlas ("taggearlas") con los distintos tópicos que representan. Por ejemplo, "Joe Biden" se clasificaría como "persona" y se etiquetaría con "política" (pueden etiquetar con más de un tópico pero las categorías son una sola por entidad).
3. **Computar estadísticas** muy simples sobre las entidades nombradas en los artículos: cuántas ocurrencias de cada entidad nombrada se encontraron por categoría y por tópico.

Especificación y posible ruta de trabajo

Interfaz de usuario

La CLI (command line interface) podrá tomar varios parámetros y los deberá procesar correctamente. La siguiente tabla muestra los parámetros con la descripción de su comportamiento y sus posibles valores.

Parámetro	**Descripción**	**Valores posibles**
`-h, --help`	Muestra la ayuda y sale del programa	

`-f, --feed`	Procesa el feed con la clave especificada	`p12pais`
		`p12eco`
		`lmgral`
		`lmnoti`

`-ne, --named-entity`	Usa la heurística especificada para extraer entidades nombradas	Keys de las heurísticas que implementen y la provista por la cátedra
`-pf, --print-feed`	Imprime el feed procesado	
`-sf, --stats-format`	Imprime las estadísticas en el formato especificado (por categoría o por tópico)	`cat` `topic`

- Si no se especifica `-f, --feed` se buscarán todos los feeds disponibles. De lo contrario se buscará solamente el especificado. La información de los feeds, incluyendo las keys que toma la CLI están en el archivo `.data/feeds.json` (pueden usar más feeds y en ese caso aceptar más valores para este parámetro).
- Si no se especifica el valor de `-ne o --named-entity`, no se computarán entidades nombradas, sólo se buscarán los feeds. En ese caso el flag `-pf o --print-feed`, se considerará activado.
- En caso de especificar el valor de `-ne o --named-entity` pero no el de `-sf, --stats-format` el comportamiento por defecto para el cómputo de estadísticas será por categoría (`cat`).

Feed Parser

Desde la cátedra les proveemos un método en la clase `FeedParser` que se encarga de hacer un fetch a la URL de un feed y retornar el contenido de este como un string. Ustedes tienen que escribir una clase `Article` que tenga como atributos `title`, `description`, `pubDate` y `link`. Además deberá contar con un método `print` que imprima los atributos de manera "pretty" (fácil de leer en la consola). El formato de la impresión debe ser el siguiente:

```
Title: Mes del Astroturismo en Córdoba: dónde y cuándo son las actividades en las distintas localidades
Description: Durante todo abril habrá decenas de actividades en más de 20 localidades. El astroturismo invita a disfrutar de la experiencia de observar atentamente el cielo estrellado, donde convive el pasado, el presente y el futuro.
Publication Date: Wed, 10 Apr 2024 12:50:00 GMT
Link:
https://lmdiario.com.ar/contenido/448748/mes-del-astroturismo-en-cordoba-donde-y-cuando-son-las-actividades-en-las-distin
********************************************************************************
```

Adicionalmente deben implementar la funcionalidad que parsea el string con el XML en una colección de instancias de la clase `Article`. Para ello completen el método `parseXML` de la clase `FeedParser`.

Entidades nombradas

Estructura de los datos

Deben pensar en una manera de estructurar los datos para representar entidades nombradas. Recuerden que cada entidad nombrada pertenece a una categoría distinta y está etiquetada con uno o más tópicos. Las categorías son: `PERSON, LOCATION, ORGANIZATION, OTHER`. Los tópicos incluyen pero no se restringen a: `POLITICS, SPORTS, ECONOMY, HEALTH, TECHNOLOGY, CULTURE, OTHER`.
Las entidades nombradas de diferentes categorías deben tener al menos una característica propia de la categoría (excepto por la categoría OTHER). Por ejemplo, una entidad nombrada de la categoría `LOCATION` puede tener atributos de longitud y

latitud. No se preocupen por como obtener los valores de estas características, simplemente piensen que la estructura las debe contemplar.
Una vez hayan pensado una manera de estructurar los datos valídenla con su docente a cargo para después implementarla.

Extracción y clasificación

La extracción de entidades nombradas del texto y título de cada artículo la realizaremos mediante heurísticas. Nosotros proveemos una heurística sencilla pero ustedes deberán crear dos más. Recuerden que el usuario podrá elegir entre ellas, por lo tanto deben pensar en como programar esto de manera consistente con el paradigma de programación orientada a objetos.

La clasificación de las entidades nombradas en categorías y tópicos la haremos mediante un diccionario. Este está provisto en el código base. El criterio es que si una entidad nombrada está en el diccionario, entonces se clasifica en la categoría y tópico correspondiente. Si no está, se clasifica en la categoría `OTHER` y tópico `OTHER`. Estas tareas son habitualmente realizadas por un modelo de machine learning y sería muy lindo hacerlo mediante una API (como la de openAI), sin embargo suelen ser pagas o por lo menos requieren el uso de API_KEYs. Para no complicar la implementación, decidimos hacerlo de esta manera. Pero si se copan y usan una librería, genial.

Cómputo de estadísticas

Una vez que hayan extraído las entidades nombradas de los artículos y las hayan clasificado deberán computar estadísticas muy simples. Deben contar cuántas veces aparece cada entidad nombrada por categoría y por tópico. La elección de computar por categoría o por tópico la hará el usuario mediante el flag `-sf` o `--stats_format` pero el comportamiento por defecto será por categoría.
El formato para imprimir las estadísticas será como en el siguiente ejemplo (y de manera análoga para el cómputo de estadísticas por tópico):

```
Category: ORGANIZATION
        instituto (2)
        cgt (1)
        belgrano (3)
Category: LOCATION
        china (1)
        argentina (2)
        cordoba (8)
        malvinas (13)
Category: OTHER
        dengue (1)
Category: PERSON
        fernandez (1)
        llaryora (6)
```

Estructura del proyecto

Esta es la estructura que les damos. Deben crear archivos y estructurar el código de manera consistente.

```
├── .gitignore
├── bin                                     # Archivos compilados .class
├── INFORME.md
├── lib                                     # 3rd party libraries .jar
│   └── json-20240303.jar                   # Librería para parsear JSON
├── Makefile
├── README.md
└── src                                     # Código fuente
    ├── App.java                            # Clase principal
    ├── data
    │   ├── dictionary.json                 # Diccionario de entidades nombradas
    │   └── feeds.json                      # Datos de feeds de noticias
    ├── feed                                # Feeds y su parseo desde XML
    │   ├── Article.java
    │   └── FeedParser.java
    ├── namedEntities                       # Entidades nombradas y sus
    │   ├── heuristics                      # Heurísticas para identificar entidades
nombradas
    │   │   └── CapitalizedWordHeuristic.java
    │   └── NamedEntity.java
    └── utils                               # Utilidades (sobretodo UI)
        ├── Config.java
        ├── FeedsData.java
        └── JSONParser.java
```

Comandos de compilación y ejecución

Les proveemos un `Makefile` que quizás pueda ser mejorado pero sus funcionalidades básicas son las siguientes:

- Para compilar el código ejecutamos

  `$ make`

  lo cual crea todos los archivos compilados en el directorio `./bin`
- Para correr el código ejecutamos

  `$ make run ARGS="<flags>"`

  donde `<flags>` son las flags que corresponden a los `args` toma la función principal del software.
- `$ make clean` borra los archivos que `.class` que se generan en la compilación

Recomendaciones

- Leer un tutorial o libro de Java antes de comenzar.
- Identificar en su aplicación cada mini-funcionalidad y mapearla a una clase que la abstraiga y la encapsule. “Mientras más clases, mejor”. No importa si una clase tiene un solo atributo y/o método, ya que, ésta inicialmente podría ser simple pero en el futuro podría crecer producto de nuevas funcionalidades.
- Testear cada mini-funcionalidad o clase por separada y a medida que las van implementado, es decir, construir una solución entera, conlleva construir pequeñas soluciones paso a paso y no pretender que todo funcione correctamente al último sin haber testeado cada parte individual de la aplicación.
- Identificar qué mini-funcionalidades son del mismo tipo o naturaleza para poder generalizarlas en una superclase utilizando herencia. En una superclase se modela aquellos atributos y métodos comunes a todas las subclases, quedando en cada subclase exclusivamente aquellos atributos y métodos propios o particulares de una subclase.
- A su vez, no exageren con la herencia: si dos funcionalidades son distintas, no fuercen una herencia solo por tener un nombre parecido.

- Hay muchas formas de resolver el laboratorio, pero queremos asegurarnos que entienden la POO. Deben hacer buenos usos de herencia, encapsulamiento, etc. Ante la duda pregunten.
- Hagan un buen manejo de los errores. No asuman que todo va a andar perfectamente, y ante un error tienen que informarlo adecuadamente y salir.
- No dar por sentado que el código del skeleton provisto por la cátedra está libre de bugs. Además pueden modificarlo libremente si ustedes lo consideran necesario.

Puntos Extras

- Utilizar una API de procesamiento de lenguaje natural para categorizar y etiquetar entidades nombradas (por ejemplo la API de OpenAI).
- Computar estadísticas más interesantes.
- Implementar lectura de feeds en formato json (ver por ejemplo https://www.reddit.com/subreddits/search.json?q=argentina).
- Completar los campos específicos de cada entidad nombrada de manera realista o semi-realista. Por ejemplo, pueden usar el string del nombre de una locación para buscar sus coordenadas mediante una API de geocoding.

Entrega y Defensa

Fecha límite de entrega: XXX a las 23:59:59 hs. Entrega del código fuente + informe mediante push en bitbucket en el repositorio asignado por la cátedra al grupo. Día y horario de la defensa a coordinar con el docente asignado a su grupo.

En la entrega

Además del código fuente, deben incluir lo siguiente:

- `INFORME.md`**:** completen el template que les proveemos.