Práctica 1: Adquisición y Preprocesamiento de Texto

Equipo: 1
Miembros: Yijun Wang, Gonzalo García

Índice de la memoria

1. **Introducción**
    - Nombres y apellidos de los integrantes del equipo.
    - Contextualización del procesamiento de lenguaje natural
    - Objetivos de la práctica
    - Descripción del caso de uso (reseñas de BoardGameGeek)
2. **Comparativa entre crawling-scraping y API**
    - Eficiencia de descarga.
    - Cobertura de datos obtenidos.
    - Ventajas e inconvenientes de cada método.
3. **Operaciones de preprocesamiento de texto implementadas**
    - Listado y breve descripción de cada operación aplicada sobre las reseñas.
4. **Estructura y funciones del corpus desarrollado**
    - Estructura interna de los datos.
    - Funciones o clases principales para acceso y manipulación del corpus.
5. **Estadísticas del corpus**
    - Número total de reseñas.
    - Número de reseñas con *rating*.
    - Número de reseñas con texto.
    - Número de reseñas con *rating* y texto.
    - Distribución de *ratings*.
    - Número de juegos y de usuarios distintos.

1. Introducción

El Procesamiento de Lenguaje Natural (PLN) es una rama de la inteligencia artificial que se centra en la interacción entre el lenguaje humano y los sistemas informáticos. Una de sus etapas iniciales más relevantes consiste en la **adquisición y preparación de datos textuales**, ya que la calidad y organización del corpus determinan en gran medida los resultados de los análisis posteriores, como clasificación, extracción de información o generación automática de texto.

Objetivos de la práctica

La práctica tiene como objetivos principales:

- Adquirir reseñas textuales públicas disponibles en la web por dos vías distintas: **crawling-scraping** y **API**.

- Implementar un proceso de **preprocesamiento básico** que permita limpiar, normalizar y segmentar los textos para su uso en futuras tareas de análisis.

- Construir un **corpus organizado y accesible** que recoja tanto las reseñas originales como sus versiones preprocesadas, y que sirva de base para las siguientes prácticas de la asignatura.

Descripción general del caso de uso (reseñas de BoardGameGeek)

El caso de uso elegido se centra en la recopilación de **reseñas de juegos de mesa publicadas en la plataforma BoardGameGeek (BGG)**, una de las bases de datos más completas a nivel mundial sobre este dominio. De cada reseña es posible extraer información como el usuario que la generó, el texto de la opinión y la puntuación numérica asociada. Estos datos resultan especialmente útiles para construir un corpus de valoraciones subjetivas, que posteriormente será empleado para el desarrollo de modelos de clasificación automática según la polaridad de las opiniones (positivas, negativas o neutras).

2. Comparativa entre crawling-scraping y API

Eficiencia de descarga

En cuanto a **eficiencia de descarga**, utilizar la API es más eficiente ya que los datos están almacenados y no se necesita parsing o esperar a que los datos terminen de cargar en el HTML como haciendo scraping. Este tiempo de carga incluye además contenido extra que puede no ser necesario, como scripts de CSS o elementos multimedia, lo que ralentiza más la tarea. Esto se puede observar claramente en la línea del crawler: *WebDriverWait(driver,3).until(EC.presence_of_all_elements_located((By.CSS_SELECTOR,"li.summary-rating-item")))*, en la que para poder empezar a extraer toda la información del HTML se necesita que haya terminado de cargar primero, lo que hace que este método sea más lento que a acceder a los datos guardados en la API de manera directa.

Cobertura de datos obtenidos

Por otro lado, se debe tener en cuenta la **cobertura de los datos obtenidos** con cada método. Una ventaja de utilizar crawling-scraping es la posibilidad de acceder y descargar datos que aparecen en las webs y que no se guardan en las APIs como reseñas ocultas o vacías. En el caso de BGG, las reseñas que estaban guardadas en la API solo eran reseñas con comentarios, por lo que se perdían muchas reseñas que sí tenían puntuación pero no comentario, perdiendo así mucha información sobre la puntuación del juego.
Sin embargo, para poder hacer scraping de las reseñas en la web no podíamos acceder a tanta información como con la API, ya que datos del juego como mecánica, número de jugadores o descripción estaban disponibles para la API y no para el crawler (estaban en otra página distinta).

Ventajas e inconvenientes

El uso de crawling-scraping también sufre de una falta de robustez. Mientras que las APIs suelen mantener una estructura estable y documentada, las páginas web pueden cambiar su HTML sin previo aviso, lo que obliga a modificar constantemente el código del crawler. Esto no solo aumenta los costes de mantenimiento sino que también puede hacer que el sistema deje de funcionar de un día para otro si el diseño de la web cambia.

Además, los datos obtenidos mediante la API suelen venir ya estructurados y normalizados, lo que facilita su procesamiento posterior. En cambio, los datos extraídos mediante scraping pueden requerir **procesos adicionales de parsing**, ya que el texto puede incluir etiquetas

HTML, saltos de línea o elementos irrelevantes. En nuestro caso, hemos tenido que procesar las fechas del HTML para convertirlas al formato *timestamp*, ya que en la web estaban escritas como: *"12 Dic 2024"*.

Por último, para resumir lo antes mencionado en **ventajas e inconvenientes de cada método**, cabe destacar que aunque utilizar crawling-scraping sea más inestable en cuanto a cambios en el HTML de BGG o a alta latencia, depende de la información que necesitemos elegiremos una opción u otra. El método que mayor información ofrece sobre los juegos de mesa es la API, aunque no incluye reseñas sin comentarios.
En nuestro caso, vamos a crear un corpus basado en reseñas de usuarios y sus puntuaciones. Como necesitaremos una gran cantidad de reseñas y puntuaciones **vamos a usar el método de crawling-scraping.**

3. Operaciones de preprocesamiento de texto implementadas

El preprocesamiento de texto constituye una fase esencial en la construcción del corpus, ya que transforma las reseñas obtenidas en datos limpios, homogéneos y adecuados para su posterior análisis.

Las principales operaciones implementadas en esta práctica fueron:

1. **Limpieza de etiquetas y símbolos no informativos**
   Se eliminaron etiquetas HTML, entidades de marcado y caracteres especiales que aparecían en las reseñas. También se normalizaron signos de puntuación, se eliminaron secuencias de caracteres repetidos (por ejemplo, "goooood" → "good") y se eliminaron espacios innecesarios.

2. **Normalización del texto**
   Se convirtió todo el contenido a minúsculas y se homogenizó el formato, reduciendo variaciones causadas por diferencias tipográficas.

3. **Sustitución de elementos no textuales**
   Se reemplazaron URLs, emojis y otros símbolos no lingüísticos por *tokens* genéricos (<URL>, <EMOJI>, etc.), lo que permite conservar la información contextual sin alterar la limpieza del texto.

4. **Detección y filtrado de idioma**
   Con la librería langdetect se identificó el idioma de cada reseña. Dado que el caso de uso se centra en el inglés, únicamente se mantuvieron las reseñas en este idioma.

5. **Eliminación de *stopwords***
   Se eliminaron las palabras vacías más comunes del inglés utilizando el conjunto de *stopwords* de NLTK, con el fin de reducir ruido léxico. Se guardó también una versión paralela del texto sin filtrar, por si fuera necesaria en etapas posteriores.

6. **Segmentación en oraciones**
   Finalmente, se aplicó una segmentación en oraciones con nltk.tokenize.sent_tokenize. Esta operación permite acceder fácilmente a las oraciones individuales de cada reseña, lo que resulta útil para futuros análisis gramaticales o semánticos.

7. **Tokenización en palabras**

   Cada oración segmentada se dividió en palabras utilizando nltk.tokenize.word_tokenize. Durante esta tokenización se eliminaron signos no alfabéticos y, si se activó el filtrado de *stopwords*, solo se conservaron los términos léxicos más relevantes.
   El resultado se almacena en el atributo tokens_by_sentence, que contiene una lista de listas con los tokens de cada oración. Esta estructura facilita el acceso directo a las unidades mínimas del texto para cálculos estadísticos, extracción de vocabulario o modelado lingüístico.

   Ejemplo del proceso de preprocesamiento

   Texto original (raw):

   "WOW!! This game is soooo good 😍😍 Check my full review here: https://boardgamegeek.com/review/12345
 10/10!!"

   Texto limpio y normalizado:

   "wow!! this game is soo good check my full review here 10 10"

   Segmentación en oraciones:

   1. "wow!! this game is soo good"
   2. "check my full review here 10 10"

      Tokenización por oración:

```
[
  ["wow", "game", "soo", "good"],
  ["check", "full", "review", "10", "10"]
]
```

4. Estructura y funciones del corpus desarrollado

El corpus desarrollado permite almacenar, acceder y procesar de forma unificada las reseñas obtenidas desde *BoardGameGeek*. Se implementó en Python con un diseño orientado a clases que facilita su reutilización y ampliación en futuras fases del proyecto.

Estructura interna de los datos

El sistema está compuesto por dos clases principales: **ReviewBGG** y **CorpusBGG**.

- **Clase ReviewBGG**
  Representa una reseña individual e incluye tanto la información original como los campos derivados del preprocesamiento.
  Los atributos principales son:

Campo	Descripción
game_id	Identificador del juego en BGG
user	Nombre de usuario que publicó la reseña
rating	Valoración numérica (1–10)
timestamp	Fecha en formato ISO-8601
text_raw	Texto original de la reseña
text_clean	Texto limpio tras el preprocesamiento
sentences	Lista de oraciones segmentadas
tokens_by_sentence	Palabras tokenizadas por oración
lang	Idioma detectado

- **Clase CorpusBGG**

  Gestiona el conjunto de reseñas y ofrece funciones para carga, preprocesamiento, filtrado y exportación de resultados.
  Sus métodos más relevantes son:

- **load_json_single_game(path):** carga reseñas de un solo juego.

- **load_json_from_ids(game_ids)**: combina varios juegos en un único corpus.
- **preprocess_all()**: aplica detección de idioma, limpieza, tokenización y filtrado de *stopwords* a todas las reseñas.
- **reviews_by_game(id)**: filtra las reseñas pertenecientes a un juego concreto.
- **save_json(path):** guarda el corpus en formato JSON.
- **stats()**: calcula estadísticas generales (reseñas, usuarios, juegos, distribución de ratings, etc.).

Ejemplo de uso

El siguiente ejemplo muestra el flujo básico de trabajo con el corpus:

```python
from CorpusBGG import CorpusBGG

# Cargar reseñas de varios juegos
corpus = CorpusBGG.load_json_from_ids([13, 174430, 187645])

# Aplicar preprocesamiento (limpieza, idioma, segmentación, stopwords)
corpus.preprocess_all()

# Consultar estadísticas
print(corpus.stats())
```

Ejemplo de salida (una reseña)


```
{
  "game_id": 142527,
  "rating": 8.0,
  "text_raw": "The least nimble ship around. No koiogran really makes life difficult. But the new antipursuit lasers in this pack bring a great boost to large
  "user": "Ultimario",
  "timestamp": "2013-10-22T22:00:00Z",
  "lang": "en",
  "text_clean": "The least nimble ship around. No koiogran really makes life difficult. But the new antipursuit lasers in this pack bring a great boost to lar
  "sentences": [
    "The least nimble ship around.",
    "No koiogran really makes life difficult.",
    "But the new antipursuit lasers in this pack bring a great boost to large ships."
  ],
  "tokens_by_sentence": [
    [
      "least",
      "nimble",
      "ship",
      "around"
    ],
    [
      "koiogran",
      "really",
```


Funciones principales de acceso y manipulación

El diseño modular del corpus permite realizar consultas específicas y mantener la trazabilidad de cada reseña. Algunos ejemplos de uso son:

- corpus.reviews_with_rating() → obtiene todas las reseñas con valoración.
- corpus.reviews_with_text() → devuelve las reseñas que contienen texto.
- corpus.reviews_by_game(13) → filtra todas las reseñas del juego *Catán*.
- corpus.stats() → devuelve un resumen con el número de reseñas, juegos, usuarios y distribución de *ratings*.

Este diseño facilita tanto la exploración inicial de los datos como su integración en futuras fases del proyecto dedicadas al análisis y modelado de texto.

5. Estadísticas del corpus

El corpus se genera utilizando las reseñas de 15 juegos distintos de BGG. Para formar un corpus significativo, se han usado 5 juegos con una puntuación alta (7-10), 5 juegos con una puntuación media (5-7) y 5 juegos con una puntuación baja (0-5). Los juegos se han seleccionado a mano y rondan las 1500-1800 reseñas cada uno.

- Número total de reseñas.

  Hay en total 27013 reseñas de 15 juegos distintas.

- Número de reseñas con *rating*.

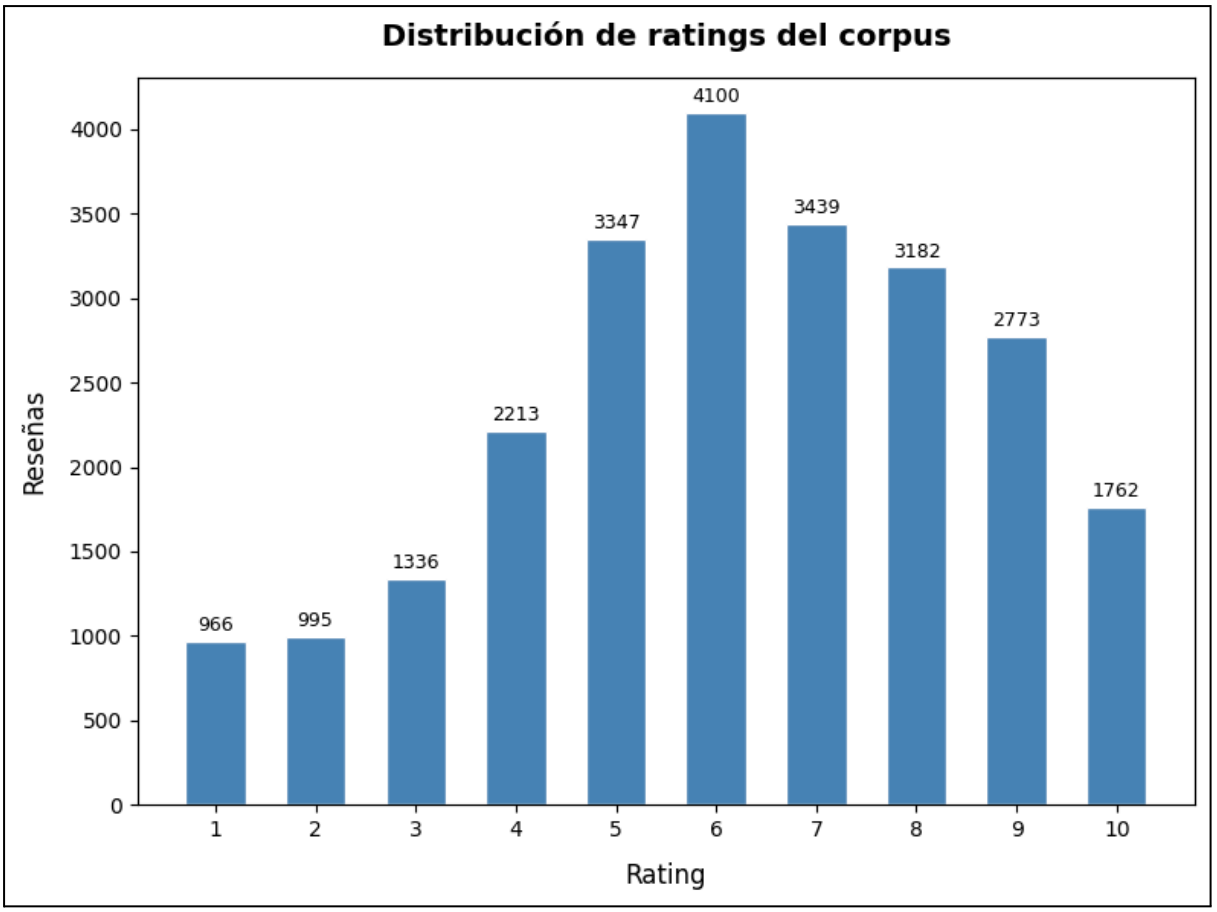
  Hay 24113 reseñas con rating.

- Número de reseñas con texto.

  En el corpus hay 6148 con texto.

- Número de reseñas con *rating* y texto.
  En total hay 3993 reseñas con *rating* y texto.

- Distribución de *ratings*.



Para realizar esta gráfica se ha tenido en cuenta que aquellas reseñas con menos de un .4 corresponden con el entero anterior (5.4 es un 5), las distribuciones completas se encuentran en las estadísticas del corpus. Por otro lado, cabe destacar que esta puede no ser la distribución final del corpus, ya que a medida que vayamos realizando prácticas siguientes estos resultados pueden cambiar.

- Número de juegos y de usuarios distintos.

Como ya se ha mencionado antes, se han recogido reseñas de 15 juegos distintos, donde hay reseñas de 22712 usuarios distintos.