Alumno:

Oscar Ivan Valenzuela Diaz

Doctorado:

Sistemas Computaciones

Materia:

Seminario de Programación de Computadoras

Actividad de Aprendizaje:

Actividad 1.- Obtención de Datos

Docente de la materia:

Dr. Gandhi Samuel Hernández Chan

Desarrollo

Introducción

En el marco del Seminario de Programación de Computadoras, se nos ha encomendado desarrollar una aplicación basada en web scraping que permita conectarse a una fuente de datos en Internet y extraer información en formato texto. Este proyecto no solo pone a prueba nuestras habilidades de programación, sino que también nos permite entender mejor las características de la arquitectura de una computadora y los métodos de comunicación entre el ser humano y la máquina mediante la programación.

Objetivo

El objetivo de este proyecto es crear una herramienta de extracción de datos (web scraper) que recopile información textual de la sección "locurioso" del sitio web de noticias "El Imparcial". La información extraída será procesada y almacenada en un archivo CSV para su posterior análisis.

Herramientas y Tecnologías Utilizadas

- Python: Lenguaje de programación principal para desarrollar el script de web scraping.
- BeautifulSoup: Biblioteca de Python para analizar documentos HTML y XML.
- urllib: Biblioteca de Python para manejar URL y realizar solicitudes HTTP.
- pandas: Biblioteca de Python para manipulación y análisis de datos.

Proceso de Desarrollo

Paso 1: Importación de Librerías

El primer paso fue importar las librerías necesarias para realizar el web scraping y el procesamiento de datos:

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import re
import string
```

Paso 2: Obtención del Contenido HTML

Se utilizó urllib.request para abrir y leer el contenido HTML de la página principal de "El Imparcial":

```
try:
    ImparcialFile =
urllib.request.urlopen('https://www.elimparcial.com/')
    ImparcialHtml = ImparcialFile.read()
    ImparcialFile.close()
except urllib.error.HTTPError as e:
    print("Error HTTP:", e.code, e.reason)
    raise
```

Paso 3: Análisis del HTML

Con BeautifulSoup, se analizó el contenido HTML para facilitar la búsqueda y extracción de elementos específicos:

```
soup = BeautifulSoup(ImparcialHtml, 'html.parser')
ImparcialAll = soup.find_all('a')
```

Paso 4: Filtrado de Enlaces

Se filtraron los enlaces que contenían la palabra 'locurioso':

```
fuente = []
enlace = []

for link in ImparcialAll:
    href = link.get('href')
    if href and 'locurioso' in href:
        fuente.append('el imparcial')
        enlace.append(href)

df = pd.DataFrame({'fuente': fuente, 'enlace': enlace})
```

Paso 5: Función para Preprocesar Texto

Se definió una función para limpiar y normalizar el texto extraído:

```
def preprocess(texto):
    texto = texto.lower()
    texto = texto.strip()
    texto = re.compile('<.*?>').sub('', texto)
    texto = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ',
texto)
    texto = re.sub(r'\d', ' ', texto)
    texto = re.sub('\s+', ' ', texto)
    return texto
```

Paso 6: Función para Extraer Texto

Se creó una función para extraer el texto de cada enlace:

```
def extraer_texto(enlace):
    nota = 'https://www.elimparcial.com' + enlace
    texto = ''
    try:
        ImparcialFile = urllib.request.urlopen(nota)
        ImparcialHtml = ImparcialFile.read()
        ImparcialFile.close()
        soup = BeautifulSoup(ImparcialHtml, 'html.parser')
        ImparcialAll = soup.find_all('p')
        for etiqueta in ImparcialAll:
            texto += str(etiqueta)
    except Exception as e:
        print(f"Error al extraer texto: {e}")
    return texto
```

Paso 7: Extracción y Preprocesamiento de Texto

Se iteró sobre cada enlace para extraer y preprocesar el texto:

```
noticias = []
for index, row in df.iterrows():
    enlace = row['enlace']
    try:
        texto = extraer_texto(enlace)
        texto = preprocess(texto)
        noticias.append(texto)
    except Exception as e:
        print(f"Error en el enlace {enlace}: {e}")
        noticias.append(None)

df['noticias'] = noticias
```

Paso 8: Limpieza del DataFrame

Se eliminaron filas con valores nulos y duplicados del DataFrame:

```
df = df.dropna()
df = df.drop_duplicates()
```

Paso 9: Guardado del DataFrame en un Archivo CSV

Finalmente, se guardó el DataFrame en un archivo CSV:

```
df.to_csv('tendencias_elimparcial.csv', index=False)
```

Conclusiones

El desarrollo de esta herramienta de extracción de datos nos permitió aplicar y consolidar conocimientos esenciales en programación, manipulación de datos y técnicas de web scraping. La aplicación desarrollada es capaz de conectarse a una fuente de datos en Internet, específicamente a la sección "locurioso" del sitio web de noticias "El Imparcial", extraer información textual relevante y almacenarla en un formato adecuado para su análisis posterior, en este caso, un archivo CSV.

Este proyecto no solo cumple con los objetivos planteados en el módulo de programación de computadoras, sino que también nos brinda una comprensión práctica de cómo interactuar con la arquitectura de una computadora. Además, nos permitió explorar y utilizar diversos métodos para establecer comunicación entre la máquina y el ser humano mediante la programación.

El proceso de elaboración de la herramienta incluyó varios pasos críticos:

Obtención del contenido HTML: Se realizó la conexión a la página web y se descargó el contenido HTML.

Análisis y filtrado de datos: Se utilizó BeautifulSoup para analizar el HTML y filtrar los enlaces que contenían la palabra 'locurioso'.

Preprocesamiento de texto: Se desarrolló una función para limpiar y normalizar el texto extraído.

Extracción de texto: Se implementó una función para extraer el contenido textual de cada enlace relevante.

Limpieza y almacenamiento de datos: Se procesó el texto extraído, eliminando duplicados y valores nulos, y finalmente se almacenó la información en un archivo CSV.

Cada uno de estos pasos fue crucial para asegurar que la información extraída fuera precisa y útil para futuros análisis. La implementación de esta herramienta nos permitió ver en acción cómo las computadoras pueden ser programadas para realizar tareas complejas de manera eficiente y efectiva, destacando la importancia de la programación en la recolección y procesamiento de datos.

En conclusión, este proyecto ha demostrado la utilidad y potencia del web scraping como herramienta para la extracción de datos, y cómo su aplicación práctica puede proporcionar valiosos insights a partir de información disponible en la web. Además, refuerza la importancia de una buena planificación y ejecución en cada etapa del desarrollo de software para lograr resultados exitosos.

Referencias:

Programación de Computadoras

- Sipser, M. (2012). Introduction to the Theory of Computation (3rd ed.). Cengage Learning.
- Hunt, A., & Thomas, D. (2019). The Pragmatic Programmer: Your Journey to Mastery (20th Anniversary Edition). Addison-Wesley.

Web Scraping

- Mitchell, R. (2018). Web Scraping with Python: Collecting More Data from the Modern Web (2nd ed.). O'Reilly Media.
- Hajba, G. L. (2018). Web Scraping with Python: A Step-by-Step Guide to Scraping the Web. Apress.

Manipulación de Datos

- McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.
- Grus, J. (2019). Data Science from Scratch: First Principles with Python (2nd ed.). O'Reilly Media.
- Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming for Total Beginners. No Starch Press.

Arquitectura de Computadoras

- Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design: The Hardware/Software Interface (5th ed.). Morgan Kaufmann.
- Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson.

ANEXO-Código - Script Python

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import re
import string

# Paso 1: Obtener el contenido HTML desde la URL
try:
    ImparcialFile =
urllib.request.urlopen('https://www.elimparcial.com/')
    ImparcialHtml = ImparcialFile.read()
    ImparcialFile.close()
except urllib.error.HTTPError as e:
    print("Error HTTP:", e.code, e.reason)
    raise

# Paso 2: Analizar el HTML utilizando BeautifulSoup
soup = BeautifulSoup(ImparcialHtml, 'html.parser')
ImparcialAll = soup.find_all('a')

# Paso 3: Filtrar los enlaces que contienen 'locurioso'
fuente = []
enlace = []

for link in ImparcialAll:
    href = link.get('href')
    if href and 'locurioso' in href:
        print(href)
        fuente.append('el imparcial')
        enlace.append(href)

# Crear un DataFrame usando los datos recolectados
df = pd.DataFrame({'fuente': fuente, 'enlace': enlace})

# Paso 4: Función para preprocesar el texto
def preprocess(texto):
    texto = texto.lower()  # Convertir a minúsculas
    texto = texto.strip()  # Eliminar espacios en blanco al inicio y
final
    texto = re.compile('<.*?>').sub('', texto)  # Eliminar etiquetas
HTML
    texto = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ',
texto)  # Eliminar puntuación
    texto = re.sub(r'\d', ' ', texto)  # Eliminar números
    texto = re.sub('\s+', ' ', texto)  # Reemplazar múltiples espacios
con uno solo
    return texto
```

```python
# Paso 5: Función para extraer texto de un enlace dado
def extraer_texto(enlace):
    nota = 'https://www.elimparcial.com' + enlace
    print(nota)
    texto = ''
    try:
        ImparcialFile = urllib.request.urlopen(nota)
        ImparcialHtml = ImparcialFile.read()
        ImparcialFile.close()
        soup = BeautifulSoup(ImparcialHtml, 'html.parser')
        ImparcialAll = soup.find_all('p')
        for etiqueta in ImparcialAll:
            texto += str(etiqueta)
    except Exception as e:
        print(f"Error al extraer texto: {e}")
    return texto

# Paso 6: Extraer y preprocesar el texto para cada enlace
noticias = []
for index, row in df.iterrows():
    enlace = row['enlace']
    try:
        texto = extraer_texto(enlace)
        texto = preprocess(texto)
        noticias.append(texto)
    except Exception as e:
        print(f"Error en el enlace {enlace}: {e}")
        noticias.append(None)

df['noticias'] = noticias

# Paso 7: Limpiar el DataFrame
df = df.dropna()  # Eliminar filas con valores nulos
df = df.drop_duplicates()  # Eliminar filas duplicadas

# Paso 8: Guardar el DataFrame final en un archivo CSV
df.to_csv('tendencias_elimparcial.csv', index=False)

# Paso 9: Imprimir el DataFrame para verificar
print(df.head())
```