Práctica 1.1: Extracción de Texto

UNR - TUIA - Procesamiento de Lenguaje Natural

Parte 1: Extracción de Texto desde Diferentes Fuentes

Ejercicio 1.1: Trabajando con archivos TXT y encodings

```
# a) Crear un archivo de texto con diferentes encodings
texto_ejemplo = "¡Hola! ¿Cómo estás? Ñandú, café, Zürich"

# Guardar en UTF-8
with open('texto_utf8.txt', 'w', encoding='utf-8') as f:
    f.write(texto_ejemplo)

# Guardar en ISO-8859-1
with open('texto_iso.txt', 'w', encoding='iso-8859-1') as f:
    f.write(texto_ejemplo)

# b) Leer ambos archivos y comparar
# TODO: Implementar la lectura de ambos archivos
# ¿Qué pasa si lees el archivo ISO con encoding UTF-8?
```

Ejercicio 1.2: Extracción desde HTML

```
from bs4 import BeautifulSoup

html_ejemplo = """
<!DOCTYPE html>
<html>
<head><title>Ejercicio de prueba 1.2</title></head>
<body>
    <h1>Título Principal</h1>
    <p class="contenido">Este es el <strong>primer</strong> párrafo.</p>
    <p class="contenido">Este es el <em>segundo</em> párrafo.</p>
    <div class="nota">
        <p>Esta es una nota especial</p>
    </div>
</body>
</html>
```

```
"""

# TODO: Usar BeautifulSoup para:
# a) Extraer solo el texto de los párrafos con class="contenido"
# b) Extraer el título de la página
# c) Obtener todo el texto sin etiquetas HTML
```

Ejercicio 1.3: Trabajando con PDFs

Para ello descargamos este archivo

```
# Instalar: pip install PyPDF2 pdfplumber

# a) Leer un PDF simple
import PyPDF2

def extraer_texto_pdf(archivo_pdf):
    """
    Extraer texto de todas las páginas de un PDF
    """
    # TODO: Implementar
    pass

# b) Comparar PyPDF2 vs pdfplumber
import pdfplumber

def comparar_extractores(archivo_pdf):
    """
    Comparar la calidad de extracción entre PyPDF2 y pdfplumber
    """
    # TODO: Extraer con ambas librerías y comparar resultados
    pass
```

Ejercicio 1.4: Web Scraping básico

```
import requests
from bs4 import BeautifulSoup

# Extraer noticias de un sitio web
def extraer_noticias_simple():
    """
    Extraer títulos de noticias de https://news.ycombinator.com/
    """
    url = "https://news.ycombinator.com/"
```

```
    # TODO: Completar la función para extraer:
    # - Título de cada noticia
    # - URL de cada noticia
    # - Número de puntos (si está disponible)

    # Retornar una lista de diccionarios
    return []

noticias = extraer_noticias_simple()
print(f"Se encontraron {len(noticias)} noticias")
```

Ejercicio 1.5: Trabajando con JSON

```
import json
import requests

# a) Leer JSON desde archivo local
datos_json = {
    "nombre": "Juan",
    "edad": 25,
    "cursos": ["Python", "NLP", "Machine Learning"],
    "activo": True
}

# Guardar JSON
with open('datos.json', 'w') as f:
    json.dump(datos_json, f, indent=2)

# TODO: Leer el archivo y extraer información específica

# b) Leer JSON desde una API
def obtener_datos_api():
    """
    Obtener datos desde una API pública (ejemplo: JSONPlaceholder)
    """
    url = "https://jsonplaceholder.typicode.com/posts/1"
    # TODO: Hacer request y parsear JSON
    pass
```

Ejercicio 1.6: Extracción desde CSV y Excel

```
import pandas as pd

# Crear un CSV de ejemplo
```

```
data = {
    'nombre': ['Ana', 'Luis', 'María'],
    'edad': [25, 30, 28],
    'ciudad': ['Rosario', 'Buenos Aires', 'Córdoba'],
    'descripcion': ['Estudiante de ingeniería', 'Desarrollador web',
'Diseñadora gráfica']
}

df = pd.DataFrame(data)
df.to_csv('personas.csv', index=False)

# TODO:
# a) Leer el CSV
# b) Extraer solo la columna 'descripcion'
# c) Combinar nombre y descripción en un texto
```

Parte 2: Procesamiento y Limpieza de Texto

Ejercicio 2.1: Limpieza básica de texto

```
import re
import string

texto_sucio = """
    ¡¡¡OFERTA ESPECIAL!!!   Compra AHORA...
    Visit@ nuestra página: www.ejemplo.com
    Contacto: info@ejemplo.com Tel: +54-341-123-4567
    #PromoVerano2024 #Descuentos
"""

def limpiar_texto_basico(texto):
    """
    Aplicar limpieza básica al texto:
    - Eliminar espacios extras
    - Convertir a minúsculas
    - Eliminar puntuación excesiva
    """
    # TODO: Implementar
    pass

def extraer_informacion(texto):
    """
    Extraer emails, URLs y teléfonos del texto
```

```
    """
    # TODO: Usar regex para extraer:
    # - Emails
    # - URLs
    # - Números de teléfono
    pass
```

Ejercicio 2.2: Parseo con expresiones regulares

```
import re

# Log de servidor web
log_entries = """
2024-01-15 10:30:45 INFO Usuario 'juan123' inició sesión desde
192.168.1.100
2024-01-15 10:32:10 ERROR Fallo en la conexión a la base de datos
2024-01-15 10:33:22 WARNING Intento de acceso no autorizado desde
10.0.0.5
2024-01-15 10:35:00 INFO Usuario 'maria456' cerró sesión
"""

def parsear_logs(log_text):
    """
    Extraer información estructurada de los logs
    Retornar lista de diccionarios con: fecha, hora, nivel, mensaje
    """
    # TODO: Implementar usando regex
    pass

# Parsear datos estructurados
factura = "FACT-2024-001542 | Cliente: Juan Pérez | Total: $1,234.56 |
Fecha: 15/01/2024"

def parsear_factura(texto_factura):
    """
    Extraer número de factura, cliente, total y fecha
    """
    # TODO: Usar regex o parse library
    pass
```

Ejercicio 2.3: Uso de la librería parse

```
from parse import parse
```

```
# Ejemplos de texto con formato consistente
ejemplos = [
    "El usuario juan@email.com se registró el 2024-01-15",
    "Producto: Laptop | Precio: $45000 | Stock: 15 unidades",
    "Error en línea 42 del archivo main.py: variable no definida"
]

# TODO: Usar parse para extraer información de cada ejemplo
# Definir patrones apropiados para cada caso
```

Ejercicio 2.4: Tokenización y análisis básico

```
def analizar_texto(texto):
    """
    Realizar análisis básico del texto:
    - Contar palabras
    - Contar oraciones
    - Palabras más frecuentes
    - Longitud promedio de palabras
    """
    # TODO: Implementar sin usar NLTK primero
    # Luego comparar con herramientas de NLTK si está disponible
    pass

texto_analisis = """
El procesamiento del lenguaje natural es una rama de la inteligencia
artificial.
Permite a las computadoras entender y generar lenguaje humano.
Es fundamental para aplicaciones como chatbots y traducción automática.
"""

estadisticas = analizar_texto(texto_analisis)
```

Parte 3: Proyecto Integrador Mini

Ejercicio 3.1: Pipeline completo de extracción y procesamiento

Crear un script que:

1. Extraiga texto de al menos 3 fuentes diferentes (web, PDF, CSV)
2. Combine todo el texto extraído
3. Aplique limpieza y procesamiento
4. Genere un reporte con estadísticas

```python
def proyecto_integrador():
    """
    Pipeline completo de extracción y procesamiento
    """

    # 1. Extracción desde web
    texto_web = ""  # TODO: Extraer de una página web

    # 2. Extracción desde PDF
    texto_pdf = ""  # TODO: Extraer de un PDF

    # 3. Extracción desde CSV
    texto_csv = ""  # TODO: Extraer columna de texto de CSV

    # 4. Combinar textos
    texto_completo = texto_web + "\n" + texto_pdf + "\n" + texto_csv

    # 5. Limpieza
    texto_limpio = ""  # TODO: Aplicar limpieza

    # 6. Análisis
    estadisticas = {}  # TODO: Generar estadísticas

    # 7. Guardar resultados
    # TODO: Guardar en JSON y/o CSV

    return estadisticas
```