ITESO, Universidad Jesuita de Guadalajara

Departamento de Matemáticas y Física

Ingeniería de características

Extracción de datos de diferentes fuentes

Dr. Gaddiel Desirena López

Primeavera 2026

Muchos datos son de carácter confidencial, es por eso que no se pueden hacer públicos o no se puede tener acceso a ellos desde un servidor, por ello se verán algunas formas de obtener los datos a través de archivos.

1. Archivos de texto

1.1. Texto a *DataFrame*

El formato más simple para almacenar datos, generalmente en formato ANSI aunque se puede presentar en UTF–8.

Para leer el archivo únicamente se debe conocer el separador, es decir, ¿los datos están separados por comas?, ¿por espacios?, ¿están alineados por un tabular?. Esto se especifica con el argumento 'sep' o 'delimiter' en uno de las funciones de *pandas* [1]

- read_table: Lee un archivo genérico delimitado y lo muetra en un DataFrame. El separador por defecto es el tabular '\t'.
- read_fwf: Lee una tabla de líneas con formato de ancho fijo (fixed–width formatted) en DataFrame. Debe ser una tabla alineada. Solo se puede especificar el ancho de la separación con 'widths'.
- read_csv: Lee un archivo de valores separados por coma (coma–separated values) a un DataFrame, sin embargo se puede especificar un separador diferente.

1.2. Texto a *regular expression*

Si lo que se desea es obtener un texto regular para procesamiento del lenguaje, se necesitan los datos como cadena de texto [2]

- Abrir el archivo:
  - file=open('path')
    texto=file.read()
    file.close()
  - with open('path') as file:
      texto=file.read()
  - Además del método 'read()' también es útil 'readline()' o 'readlines()', el primero es un iterador, el segundo regresa una lista de las líneas del archivo.

- Una librería para el tratamiento de cadenas de texto es 're' (regular expression). Algúnas funciones son
    - search(pattern,string): Busca en *string* y regresa el primer lugar de *pattern*. Regresa el objeto que coincide.
    - match(pattern,string): Si cero o más caracteres al principio de *string* coinciden con *pattern*, regresa el objeto que coincide.
    - split(pattern,string): Divide la *string* por el número de ocurrencias de *pattern*. Regresa una lista con el texto dividido.
    - findall(pattern,string): Retorna todas las coincidencias no superpuestas del *pattern* en la *string*, como una lista de cadenas. La cadena es examinada de izquierda a derecha, y las coincidencias son retornadas en el orden en que fueron encontradas.
    - sub(pattern,repl,string): Retorna la cadena obtenida reemplazando las ocurrencias no superpuestas de *pattern* en *string* por el reemplazo de *repl*. Si el patrón no se encuentra, se retorna string sin cambios. *repl* puede ser una cadena o una función.

2. Archivos CSV

Archivos separados por comas, éstos se leen, como ya se vio con la función 'read_csv' del paquete *pandas* y regresa un objeto *DataFrame*.

3. Archivos XLS o XLSX

Archivos generados por el software Excel o algún otro creador de hojas de cálculo. El paquete *pandas* ofrece la función 'read_excel('str')', donde 'str' es el nombre del archivo escrito entre comillas. La función regresa un *DataFrame* y soporta archivos XLS, XLSX, XLSM, XLSB, ODF, ODS y ODT.

Otra forma de leer hojas de cálculo es creando un objeto de clase *pandas.ExcelFile*, éste cuenta con el método 'parse()'.

En ambos casos se puede especificar si cuentan o no con encabezado, el nombre o nombres de las hojas a leer, el nombre o nombres de las columnas, entre otras cosas.

4. Archivos JOSN

Archivos para almacenar y compartir información de *Java Script*. Se pueden leer con la función de *pandas* 'read_json('str')', donde 'str' puede ser el nombre del archivo entre comillas, una dirección web u objetos de tipo *file*.

5. Archivos XML

Archivos con lenguaje *MarkUp*, *pandas* tiene la función 'read_html('str')', donde 'str', al igual que en el caso de los archivos JSON, puede ser el nombre del archivo entre comillas, una dirección url o un texto que contenga HTML.

Los archivos XML se leen a través del método 'parse' del objeto *xml.etree.ElementTree()*, del elemento resultante se extrae la "raíz" con el método 'getroot()'. Otra opción es a través del método 'fromstring('str')' del mismo objeto, la diferencia ahora es que 'str' es el objeto de tipo *file* con formato XML.

Los elementos raíz son iterables, donde cada iteración, igual que un diccionario cuenta con *tag* y *attrib*, este último cuenta con la estructura de un diccionario.

6. Archivos SHP

Un shapefile (SHP) es un formato no topológico que se utiliza para almacenar la ubicación geométrica y la información de atributos de las entidades geográficas.

Es un formato multiarchivo. El número mínimo requerido es de tres y tienen las extensiones siguientes [4]:

- SHP: es el archivo que almacena las entidades geométricas de los objetos.
- SHX: es el archivo que almacena el índice de las entidades geométricas.
- DBF: es la base de datos, en formato dBASE, donde se almacena la información de los atributos de los objetos.

Opcionalmente se pueden utilizar otros para mejorar el funcionamiento en las operaciones de consulta a la base de datos, información sobre la proyección cartográfica, o almacenamiento de metadatos. Estos archivos son:

- PRJ: Es el archivo que guarda la información referida al sistema de coordenadas en formato WKT.
- SBN y SBX: Almacena el índice espacial de las entidades.
- FBN y FBX: Almacena el índice espacial de las entidades para los shapefiles que son inalterables (solo lectura).
- AIN y AIH: Almacena el índice de atributo de los campos activos en una tabla o el tema de la tabla de atributos. XML: Almacena los metadatos del shapefile.

Para importar estos datos en un *DataFrame* de *pandas*, instalamos e importamos el paquete 'geopandas', en él la función 'read_file('archivo')' lee los archivos disponibles con las extinciones descritas anteriormente [5].

7. De imágenes

Si se desea hacer un procesamiento de imágenes, como identificación de números, reconocimiento de patrones en firmas o un análisis del estado de salud de una planta con IA; la obtención de datos a partir de una imagen, ya sea en escala de de grises o a color, es imperativo. Estos datos se obtienen con la función 'imread' de *matplotlib.pyplot* y regresa una matriz en el caso de imágenes en escala de grises o un arreglo de tres o cuatro matrices para imágenes a color.

Referencias

[1] `https://pandas.pydata.org/docs/reference/io.html`

[2] `https://docs.python.org/3/library/re.html`

[3] `https://docs.python.org/3/library/xml.etree.elementtree.html`

[4] `https://es.wikipedia.org/wiki/Shapefile`

[5] `https://geopandas.org/`