Transformando CSV a Grafo de Conocimiento: El Poder de la Estructura de Datos

De Tablas a Tripletes: Convirtiendo datos tabulados en conocimiento conectado.

Integrante: Diana Rocio Bermeo Cabrera

OBJETIVO

- Diseñar e implementar un algoritmo que permita convertir datos en formato tabular (CSV) a un modelo RDF (Resource Description Framework), aplicando principios de la Web Semántica y buenas prácticas de modelado de datos enlazados (Linked Data).

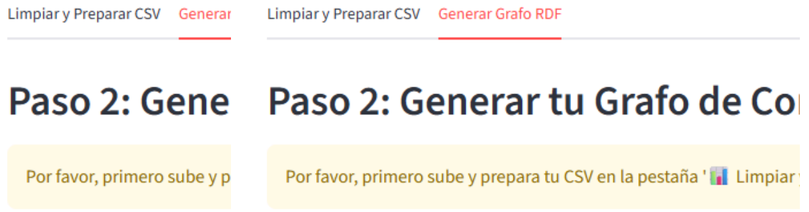
NECESIDAD

La necesidad clave de este proyecto es transformar datos tabulares (CSV) en conocimiento estructurado RDF, superando el desafío de las columnas multivaluadas para crear un grafo rico y conectado. Se busca automatizar y simplificar este proceso para usuarios, permitiendo que datos "planos" se conviertan en entidades y relaciones semánticamente significativas, esenciales para la web de datos conectados.

¿Cómo lo logramos?

Arquitectura y Diseño

La arquitectura se centra en un sistema modular de tres componentes lógicos que trabajan en conjunto para convertir datos CSV a RDF, con especial énfasis en el manejo de columnas multivaluadas



Códigos

limpiar_csv.py

Se encarga de la fase inicial de preprocesamiento y limpieza de un DataFrame de Pandas. Su objetivo es preparar los datos brutos del CSV para que sean consistentes y estén en el formato correcto antes de ser transformados en un grafo RDF.
En concreto, sus funciones principales son:

1. Manejo de Valores Nulos (NaN):
    - Identifica heurísticamente columnas que probablemente son identificadores únicos (como IDs, DOIs, etc) y elimina las filas completas si tienen valores nulos en estas columnas. Esto asegura que los registros clave estén completos.
    - Relleno de Nulos: Para otras columnas, rellena los valores nulos de forma inteligente:
        - Con cadenas vacías ('') para columnas de texto.
        - Con ceros (0) para columnas numéricas (como volumen, número de artículo, conteo de citas) que son candidatas a ser enteros.
2. Inferencia y Conversión de Tipos de Datos:
    - Analiza el contenido de cada columna para inferir su tipo de dato más apropiado (entero, flotante, booleano, cadena, o fecha/hora).
    - Aplica las conversiones necesarias para asegurar que los datos estén en el tipo correcto (ej., convertir una columna de números que Pandas leyó como texto a enteros o flotantes, o cadenas que parecen fechas a objetos datetime).

Códigos

convertir_a_rdf.py.

Es la base de la transformación de datos tabulares a un grafo de conocimiento RDF. Su rol principal es tomar un DataFrame de Pandas ya limpio y, basándose en un mapeo dinámico y configurable por el usuario, construir un grafo RDF estructurado.
Sus responsabilidades clave son:

- Crea un grafo RDF vacío (rdflib.Graph).
- Asocia prefijos comunes (como bibo, schema, dct, foaf) y un namespace personalizado (drber) con sus URIs completas. Esto hace que el grafo sea más legible y estandarizado.
- Incluye una función auxiliar (clean_uri_segment) que sanitiza cadenas de texto (eliminando espacios, caracteres especiales, etc.) para asegurar que los identificadores generados para las entidades RDF sean URIs válidas y consistentes.
- Por cada fila del DataFrame limpio, crea una instancia de la "entidad principal" (definida por el usuario, por ejemplo, un drber:Record o schema:Article).
- Genera una URI única para esta entidad principal, utilizando un valor de una columna específica del CSV (o el índice de la fila si no se especifica una columna de ID).
- Añade triples básicos al grafo para definir el tipo de la entidad y una etiqueta legible (rdfs:label).

Códigos

convertir_a_rdf.py.

- Itera sobre cada columna del DataFrame y aplica las reglas de mapeo definidas por el usuario:
  - Propiedades Literales: Convierte los valores de las celdas en literales RDF (texto, números, fechas, etc.) con el tipo de dato XSD especificado.
  - Propiedades de Objeto (Entidades Relacionadas): Si una columna está configurada para generar una entidad relacionada (ej., un autor, una institución, una revista):
    - Crea una nueva instancia de entidad RDF para cada valor único en esa columna (o sus sub-valores si es multivaluada).
    - Asigna una URI única y un tipo (clase) a esta nueva entidad relacionada.
    - Deduplica estas entidades relacionadas a nivel global: si un autor o institución ya ha sido creado en una fila anterior, reutiliza su URI existente en lugar de crear un duplicado.
    - Establece una relación (propiedad de objeto) entre la entidad principal de la fila y esta nueva entidad relacionada (ej., dct:creator vinculando un artículo a un autor).
    - Manejo de Columnas Multivaluadas: Divide los valores de las celdas que contienen múltiples elementos (separados por un delimitador como ;) y procesa cada sub-valor individualmente, ya sea como un literal o como una entidad relacionada.
    - Asignación de Propiedades a Entidades Relacionadas: Permite que otras columnas del CSV (que son literales) se mapeen como propiedades de las entidades relacionadas recién creadas en la misma fila (ej., una columna "NombreCompletoAutor" se asigna como foaf:name al nodo del autor, no al artículo).
    - **Añade todos los triples generados (sujeto-predicado-objeto) al grafo RDF.**

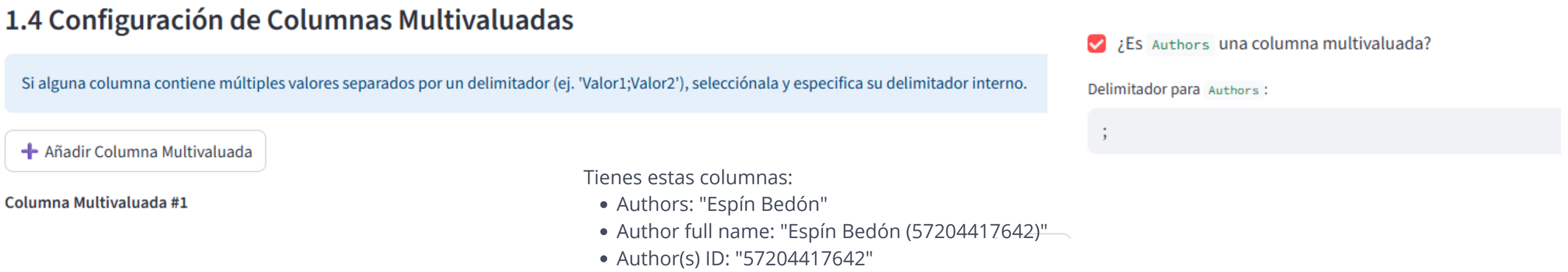
Códigos

app.py

Funciona como la interfaz de usuario principal y el controlador central de la aplicación. Su objetivo es guiar al usuario a través de un proceso interactivo de dos fases para transformar datos de un archivo CSV en un grafo de conocimiento RDF.
Sus funciones esenciales incluyen:

1. Presenta una interfaz basada en pestañas (Limpiar y Preparar CSV y Generar Grafo RDF) para estructurar el proceso paso a paso.
2. Gestión de la Interfaz de Usuario: Utiliza Streamlit para crear y gestionar todos los elementos interactivos, como botones para subir archivos, campos de texto para configuraciones (delimitadores, URIs), selectores para columnas y casillas de verificación para opciones de mapeo.
3. Manejo de Datos del Usuario: Permite la carga de archivos CSV, muestra previsualizaciones de los datos y gestiona el estado de las configuraciones del usuario a lo largo de las diferentes interacciones (por ejemplo, los nombres de columnas renombradas o las configuraciones de mapeo RDF).
4. Coordinación de Procesos: Actúa como un "director de orquesta", llamando a las funciones especializadas de los otros módulos:
    - Invoca a limpiar_dataframe_generico (de limpiar_csv.py) para preprocesar y limpiar el CSV.
    - Invoca a convertir_dataframe_a_rdf (de convertir_a_rdf.py) para construir el grafo RDF basándose en las configuraciones del usuario.
5. Realiza validaciones básicas de las entradas del usuario para asegurar la coherencia de los mapeos y proporciona mensajes claros de éxito, advertencia o error.
6. Entrega del Resultado: Una vez generado el grafo RDF, ofrece opciones para descargar el resultado en formatos Turtle y RDF/XML, y muestra una vista previa del grafo directamente en la aplicación.

Columnas Multivaluadas en la Generación de Grafos RDF



- Escenario 1 (Pestaña 1 - Preparación): Le dice al sistema qué columnas pueden tener múltiples valores y qué carácter (el delimitador, como ;) los separa. Es como enseñarle al sistema a "cortar" cadenas largas.
- Escenario 2 (Pestaña 2 - Mapeo a RDF): Le dice al sistema cómo usar esos valores ya "cortados" para crear entidades (como personas) en el grafo RDF, asignarles una propiedad (ej. dct:creator), darles un tipo (ej. foaf:Person) y vincularlos a su ID único. Es como darle el "sentido" a cada parte cortada para construir el grafo.

Demostración de App:

Explora y Conecta