```
!git clone https://github.com/pbloem/kgbench-loader.git
%cd kgbench-loader
!pip install .
```

```
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.8 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 81.7 MB/s eta 0:00:00
Building wheels for collected packages: kgbench, fire, wget
  Building wheel for kgbench (setup.py) ... done
  Created wheel for kgbench: filename=kgbench-0.2-py3-none-any.whl size=14130 sha256=cce60d3d3376ae5a4ba64ce436c7e48ca7c9c315c93b5a
  Stored in directory: /root/.cache/pip/wheels/0a/19/59/262e8e140b9b0256a1c0d914f1d588d08ed7e7f9714a9d647a
  Building wheel for fire (setup.py) ... done
  Created wheel for fire: filename=fire-0.7.0-py3-none-any.whl size=114249 sha256=c0460ee9f772ed329ca4decd5e324f757b01336e33f0fd90c
  Stored in directory: /root/.cache/pip/wheels/46/54/24/1624fd5b8674eb1188623f7e8e17cdf7c0f6c24b609dfb8a89
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9656 sha256=e9fbe72c75ccc23e9dabe0bdf618ea975237ffd7e0c1ecc3819f2
  Stored in directory: /root/.cache/pip/wheels/40/b3/0f/a40dbd1c6861731779f62cc4babcb234387e11d697df70ee97
Successfully built kgbench fire wget
Installing collected packages: wget, rdflib, nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed fire-0.7.0 kgbench-0.2 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12
```

```
from kgbench import load
dataset = load('dmg777k',torch=True)
```

```
Downloading dmg777k dataset.
loaded data dmg777k (64.19s).
```

```
print(f"Dataset type: {type(dataset)}")
print(f"Available attributes: {dir(dataset)}")
```

```
Dataset type: <class 'kgbench.load.Data'>
Available attributes: ['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribut
```

```
# Import necessary libraries
import kgbench
import torch
import networkx as nx
import matplotlib.pyplot as plt
from PIL import Image

# Load dataset (Change 'dmg777k' to your dataset if needed)


# Print basic dataset info
print(f"Dataset Name: {dataset.name}")
print(f"Number of Entities (Nodes): {dataset.num_entities}")
print(f"Number of Relations (Edges): {dataset.num_relations}")
print(f"Number of Triples: {len(dataset.triples)}\n")

# Check Available Attributes
print("Dataset Attributes:")
print(dir(dataset))

# Explore Triples (Knowledge Graph structure)
print("\nSample Triples (subject, predicate, object):")
for triple in dataset.triples[:5]:
    print(triple)

# Entity-to-Index and Index-to-Entity mappings
print("\nEntity-to-Index Mapping (First 5):", list(dataset.e2i.items())[:5])
print("Index-to-Entity Mapping (First 5):", list(dataset.i2e)[:5])
print("Index-to-Relation Mapping (First 5):", list(dataset.i2r)[:5])

# Check available data types
print("\nAvailable Data Types:", dataset.datatypes())

# Check if there are images or text
if "image" in dataset.datatypes():
    print("Dataset contains image data.")
    img = dataset.get_images(0)
    if img is not None:
        plt.imshow(img)
        plt.axis('off')
        plt.title("Sample Image from Dataset")
        plt.show()
    else:
        print("No image available for this node.")

if "text" in dataset.datatypes():
    print("\nSample text from dataset:", dataset.get_strings(0))

# Convert dataset to PyG format (for Graph Neural Networks)
try:
    graph_pyg = dataset.pyg()
    print("\nPyG Graph Structure:")
    print(graph_pyg)
except Exception as e:
    print("\nPyG Conversion Error:", e)

# Convert dataset to DGL format (for DGL-based GNNs)
try:
    graph_dgl = dataset.dgl()
    print("\nDGL Graph Structure:")
    print(graph_dgl)
except Exception as e:
    print("\nDGL Conversion Error:", e)
```

```
# Visualize a small subgraph using Network
```

```
Dataset Name: dmg777k
Number of Entities (Nodes): 341270
Number of Relations (Edges): 60
Number of Triples: 777124

Dataset Attributes:
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getstate__',

Sample Triples (subject, predicate, object):
tensor([130685,     28,  54795])
tensor([130685,     31, 201822])
tensor([130690,     28,  58948])
tensor([130690,     31, 201822])
tensor([130691,     28,  63024])

Entity-to-Index Mapping (First 5): [((0, ('', 'none')), 341269), ((1, ('"\'t Hof Alle Morgen Nieuwe Zorgen". Bakstenen woonhuis met p
Index-to-Entity Mapping (First 5): [('', 'none'), ('"\'t Hof Alle Morgen Nieuwe Zorgen". Bakstenen woonhuis met pannen zadeldak. Tuit
Index-to-Relation Mapping (First 5): ['http://data.pdok.nl/def/pdok#asWKT-RD', 'http://dbpedia.org/ontology/city', 'http://dbpedia.or

Available Data Types: ['iri', 'none', '@es', '@fy', '@nl', '@nl-nl', '@pt', '@ru', 'http://kgbench.info/dt#base64Image', 'http://www.

PyG Conversion Error: Pytorch geometric does not appear to be installed. Try `pip install pyg` to install it.

DGL Conversion Error: DGL does not appear to be installed. Try `pip install dgl` to install it. See http://dgl.ai for more informatic
```

```
len(dataset.triples)
```

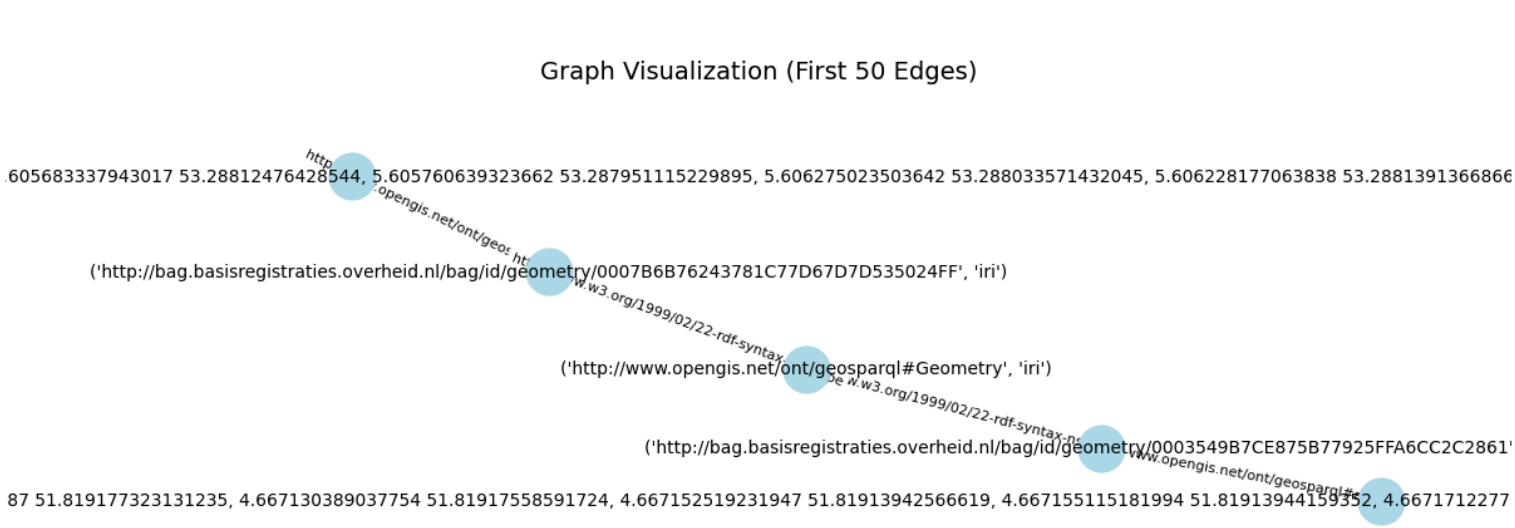
```
777124
```

```
import matplotlib.pyplot as plt
import random

# Get the images (you may adjust this based on how your dataset returns them)
images = dataset.get_images()

# Randomly select 20 indices (ensure there are enough images)
num_images_to_show = 10
random_indices = random.sample(range(len(images)), min(num_images_to_show, len(images)))

# Display the selected random images
plt.figure(figsize=(15, 15))  # Increase the figure size to make room for multiple images
for i, idx in enumerate(random_indices):
    plt.subplot(4, 5, i+1)  # Create a 4x5 grid (4 rows, 5 columns)
    plt.imshow(images[idx])
    plt.axis('off')  # Hide axes for better visualization
    plt.title(f"Image {idx+1}")

plt.show()
```


```
print(len(images))
```

```
46061
```

```
import networkx as nx
import matplotlib.pyplot as plt

# Create a graph
G = nx.Graph()
n=5
# Add edges from dataset (taking a subset for visualization)
for (s, p, o) in dataset.triples[:n]:  # First 50 triples
    G.add_edge(dataset.i2e[s], dataset.i2e[o], label=dataset.i2r[p])

# Use a better layout
pos = nx.spring_layout(G, seed=42)  # For better spacing

# Plot the graph with enhancements
plt.figure(figsize=(12, 8))
nx.draw(G, pos, with_labels=True, node_size=700, font_size=10, edge_color="gray", node_color="lightblue")

# Draw edge labels
edge_labels = {(dataset.i2e[s], dataset.i2e[o]): dataset.i2r[p] for (s, p, o) in dataset.triples[:n]}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8)

plt.title("Graph Visualization", fontsize=14)
plt.show()
```




```
unique_relations = set()  # Store unique predicates

for (_, p, _) in dataset.triples:  # Loop through triples
    unique_relations.add(dataset.i2r[p])  # Convert index to relation string

# Print all unique relationships
print("Unique Relationships (Predicates) in the Graph:")
for rel in sorted(unique_relations):  # Sorting for easier readability
    print(rel)

# Count of unique relations
print("\nTotal Unique Relationships:", len(unique_relations))
```

```
Unique Relationships (Predicates) in the Graph:
http://data.pdok.nl/def/pdok#asWKT-RD
http://dbpedia.org/ontology/city
http://dbpedia.org/ontology/codeNationalMonument
http://dbpedia.org/ontology/location
http://dbpedia.org/ontology/name
http://dbpedia.org/ontology/neighbourhood
http://dbpedia.org/ontology/thumbnail
http://purl.org/dc/terms/created
http://purl.org/dc/terms/creator
http://purl.org/dc/terms/description
http://purl.org/dc/terms/isPartOf
http://purl.org/dc/terms/spatial
http://schema.org/dateCreated
http://schema.org/roleName
http://www.geonames.org/ontology#alternateName
http://www.geonames.org/ontology#countryCode
http://www.geonames.org/ontology#featureClass
http://www.geonames.org/ontology#featureCode
http://www.geonames.org/ontology#locationMap
http://www.geonames.org/ontology#name
http://www.geonames.org/ontology#nearbyFeatures
http://www.geonames.org/ontology#officialName
http://www.geonames.org/ontology#parentADM1
http://www.geonames.org/ontology#parentADM2
http://www.geonames.org/ontology#parentCountry
http://www.geonames.org/ontology#parentFeature
http://www.geonames.org/ontology#population
http://www.geonames.org/ontology#wikipediaArticle
http://www.opengis.net/ont/geosparql#asWKT
```

http://www.opengis.net/ont/geosparql#hasGeometry
http://www.opengis.net/ont/geosparql#sfWithin
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2000/01/rdf-schema#isDefinedBy
http://www.w3.org/2000/01/rdf-schema#label
http://www.w3.org/2000/01/rdf-schema#seeAlso
http://www.w3.org/2002/07/owl#sameAs
http://www.w3.org/2003/01/geo/wgs84_pos#alt
http://www.w3.org/2006/vcard/ns#hasStreetAddress
http://www.w3.org/2006/vcard/ns#postal-code
http://www.w3.org/2006/vcard/ns#street-address
http://xmlns.com/foaf/0.1/depiction
http://xmlns.com/foaf/0.1/depicts
http://xmlns.com/foaf/0.1/name
https://data.labs.pdok.nl/rce/def/bouwjaar
https://data.labs.pdok.nl/rce/def/complexnummer
https://data.labs.pdok.nl/rce/def/fotograaf
https://data.labs.pdok.nl/rce/def/graveur
https://data.labs.pdok.nl/rce/def/huisnummer
https://data.labs.pdok.nl/rce/def/huisnummerCompleet
https://data.labs.pdok.nl/rce/def/huisnummerToevoeging
https://data.labs.pdok.nl/rce/def/internComplexNummers
https://data.labs.pdok.nl/rce/def/isFree
https://data.labs.pdok.nl/rce/def/locator
https://data.labs.pdok.nl/rce/def/ontwerper
https://data.labs.pdok.nl/rce/def/reprofotograaf
https://data.labs.pdok.nl/rce/def/rnaSubject
https://data.labs.pdok.nl/rce/def/schilder

Start coding or generate with AI.

```
Number of Unique Nodes: 341270
```

```
print(dataset.i2e[130685])  # Get subject name
print(dataset.i2r[28])      # Get relation name
print(dataset.i2e[54795])   # Get object name
```

```
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0003549B7CE875B77925FFA6CC2C2861', 'iri')
http://www.opengis.net/ont/geosparql#asWKT
('POLYGON ((4.667227191836703 51.81920171648348, 4.667236120685068 51.819203883584755, 4.6672263678506125 51.81920790458488, 4.667215
```

Start coding or generate with AI.

```
# Get the first two nodes
nodes_list = list(G.nodes())  # Convert nodes to a list

if len(nodes_list) >= 2:
    node1, node2 = nodes_list[:2]  # Extract first two nodes
    print("First node:", node1)
    print("Second node:", node2)
elif len(nodes_list) == 1:
    print("Only one node present:", nodes_list[0])
else:
    print("Graph has no nodes.")

# Get one example edge with label (if available)
if len(G.edges) > 0:
    example_edge = next(iter(G.edges(data=True)))
    u, v, data = example_edge
    relation_label = data.get("label", "Unknown")
    print(f"Example edge: {u} --({relation_label})--> {v}")
else:
    print("Graph has no edges.")
```

```
First node: ('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0003549B7CE875B77925FFA6CC2C2861', 'iri')
Second node: ('POLYGON ((4.667227191836703 51.81920171648348, 4.667236120685068 51.819203883584755, 4.6672263678506125 51.8192079045
Example edge: ('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0003549B7CE875B77925FFA6CC2C2861', 'iri') --(http://www.oper
```

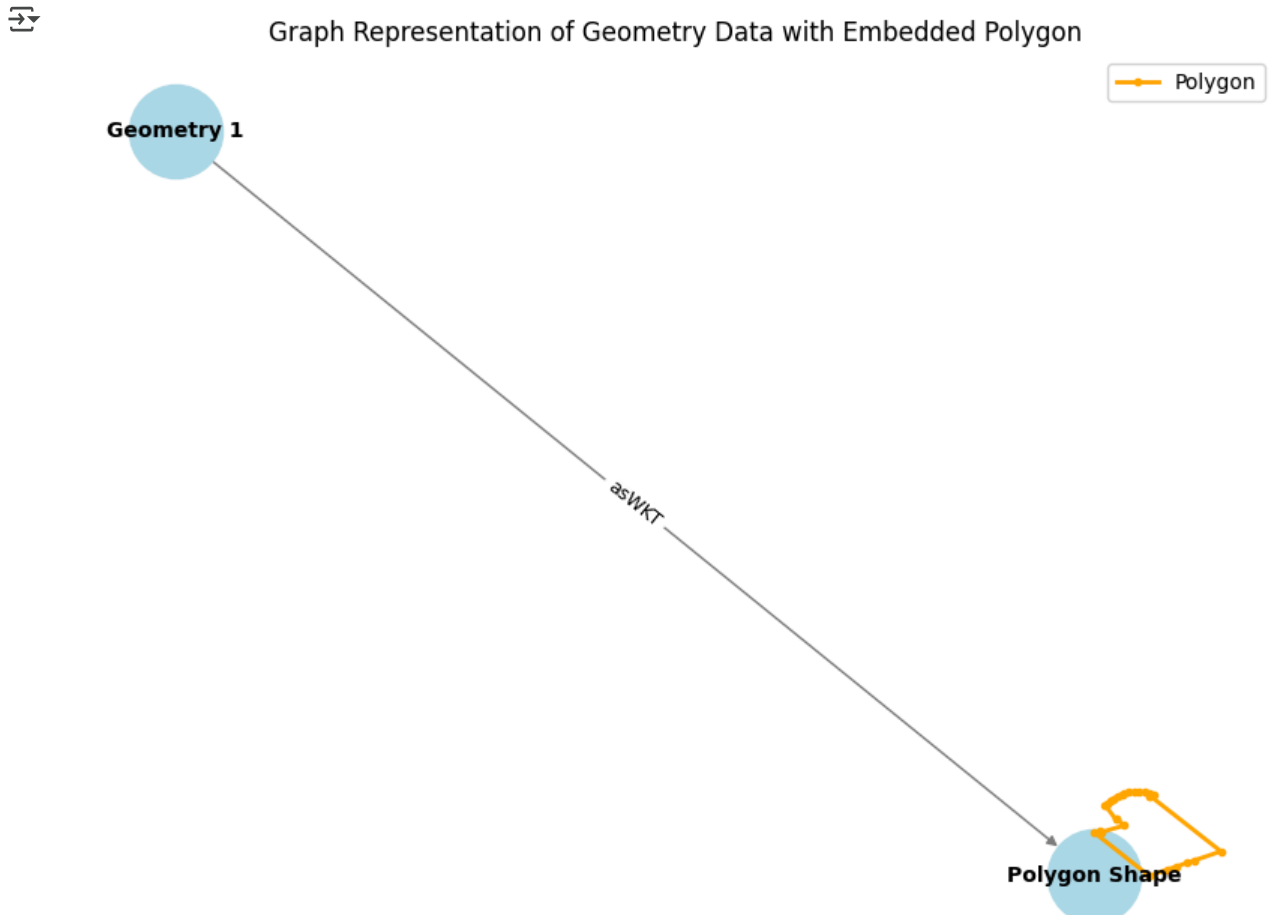
New visualisation

```
!pip install adjustText
```

```
Collecting adjustText
  Downloading adjustText-1.3.0-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from adjustText) (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from adjustText) (3.10.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from adjustText) (1.14.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->adjustText) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->adjustText
Downloading adjustText-1.3.0-py3-none-any.whl (13 kB)
Installing collected packages: adjustText
Successfully installed adjustText-1.3.0
```

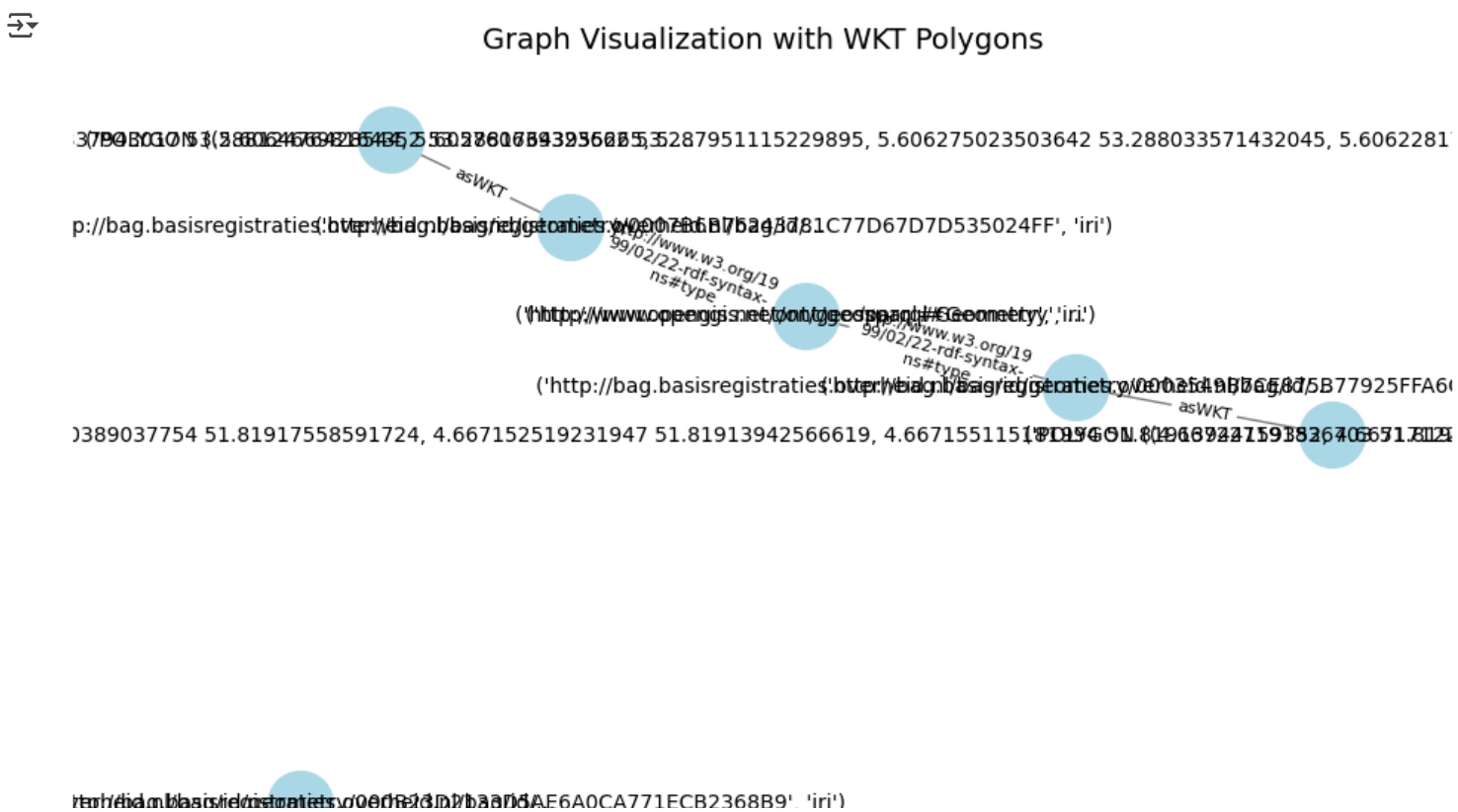
```
import networkx as nx
import matplotlib.pyplot as plt
from shapely.wkt import loads  # To parse WKT polygon

# Create a directed graph
G = nx.DiGraph()

# Example node identifiers
geometry_node = "Geometry 1"
polygon_node = "Polygon Shape"
edge_label = "asWKT"

# Add nodes with metadata
G.add_node(geometry_node, full_url='http://bag.basisregistraties.overheid.nl/bag/id/geometry/0003549B7CE875B77925FFA6CC2C2861')
G.add_node(polygon_node, full_url='http://www.opengis.net/ont/geosparql#wktLiteral')

# Add edge
G.add_edge(geometry_node, polygon_node, label=edge_label)

# Compute positions using a layout algorithm
pos = nx.spring_layout(G, seed=42)

# Draw graph nodes and edges
plt.figure(figsize=(8, 6))
nx.draw(G, pos, with_labels=True, node_color='lightblue', edge_color='gray', node_size=2000, font_size=10, font_weight="bold")

# Draw edge labels
edge_labels = {(geometry_node, polygon_node): edge_label}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=9)

# POLYGON DATA
wkt_polygon = "POLYGON ((4.667227191836703 51.81920171648348, 4.667236120685068 51.819203883584755, 4.6672263678506125 51.81920790458488,

# Parse the WKT polygon
polygon = loads(wkt_polygon)

# Extract polygon coordinates
polygon_coords = list(polygon.exterior.coords)
x_vals, y_vals = zip(*polygon_coords)

# Position the polygon inside the "Polygon Shape" node
polygon_center = pos[polygon_node]

# Adjust polygon coordinates to fit within the node
scale_factor = 1000  # Adjust this for better fit
x_scaled = [polygon_center[0] + scale_factor * (x - min(x_vals)) for x in x_vals]
y_scaled = [polygon_center[1] + scale_factor * (y - min(y_vals)) for y in y_vals]
```

```
# Plot the polygon INSIDE the node
plt.plot(x_scaled, y_scaled, color='orange', linewidth=2, marker="o", markersize=3, label="Polygon")

# Show title and legend
plt.title("Graph Representation of Geometry Data with Embedded Polygon", fontsize=12)
plt.legend()
plt.show()
```



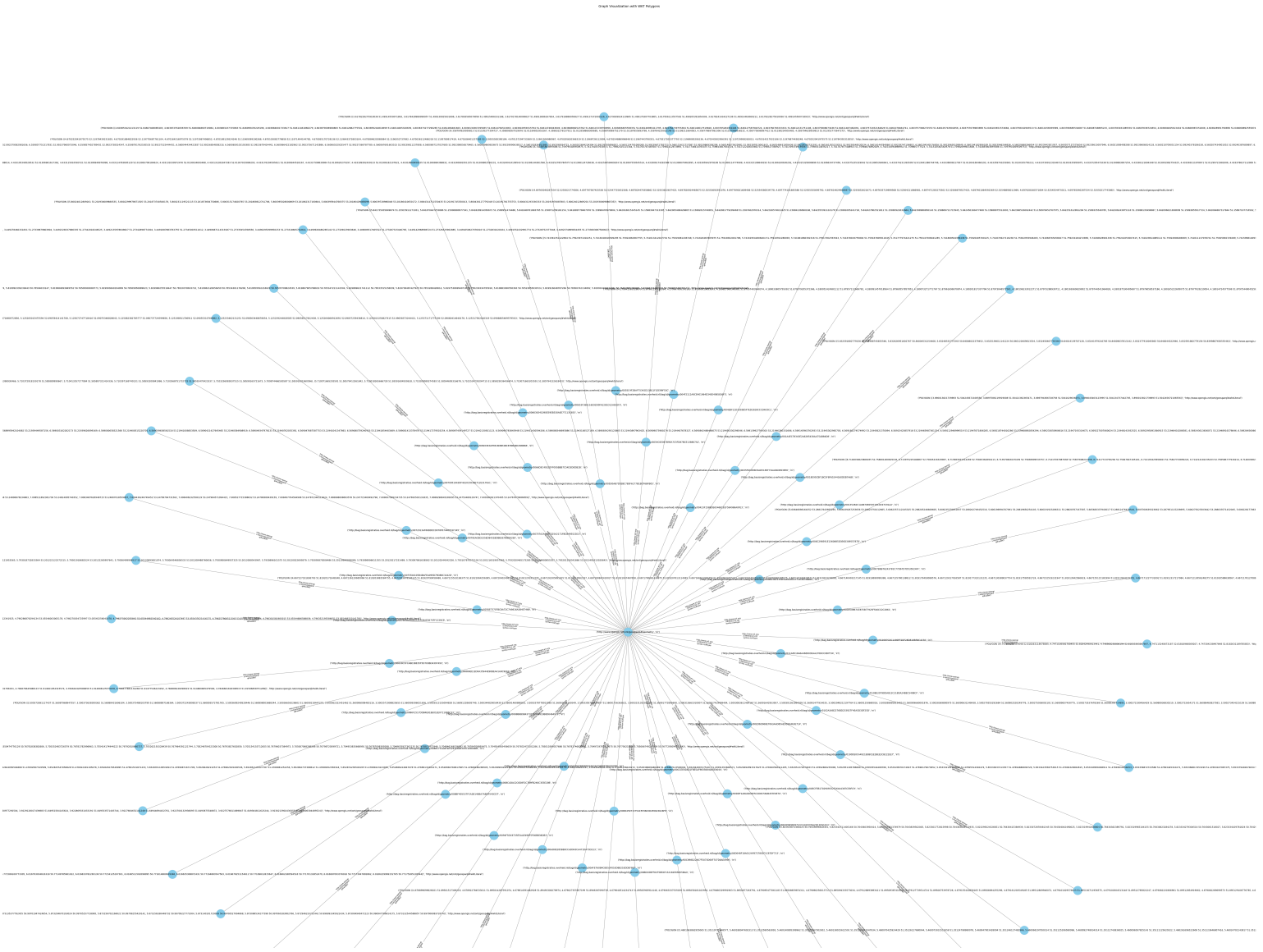

```
import networkx as nx
import matplotlib.pyplot as plt
from shapely.wkt import loads
from shapely.geometry import Polygon
import textwrap


def shorten_label(label, max_length=50):
    """Shorten long node labels for better visualization."""
    label = str(label)

    return label[:max_length] + "..." if len(label) > max_length else label

# Create a graph
G = nx.Graph()
n = 5  # Number of triples to visualize
polygons = []  # Store polygons separately for plotting

# Add edges and detect polygons
for (s, p, o) in dataset.triples[:n]:
    node1 = dataset.i2e[s]
    node2 = dataset.i2e[o]
    relation = dataset.i2r[p]

    # Handle WKT polygons separately
    if is_wkt_polygon(node2):
        polygons.append(loads(node2))  # Convert WKT to geometry for plotting
        node2 = "[POLYGON]"  # Replace with a simple label
```

```
    # Add edges with shorter relations (e.g., show "asWKT" instead of long URIs)
    relation = "asWKT" if relation == "http://www.opengis.net/ont/geosparql#asWKT" else relation
    G.add_edge(node1, node2, label=relation)

# Use a better layout
pos = nx.spring_layout(G, seed=42)

# Create figure
fig, ax = plt.subplots(figsize=(12, 8))

# Draw the graph
nx.draw(G, pos, with_labels=True, node_size=1000, font_size=10,
        edge_color="gray", node_color="lightblue", ax=ax)

# Format edge labels
edge_labels = {(u, v): textwrap.fill(data["label"], width=20) for u, v, data in G.edges(data=True)}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8, ax=ax)

# Shorten node labels for clarity
node_labels = {node: shorten_label(node) for node in G.nodes()}
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=10, ax=ax)

# Plot the polygons separately
for poly in polygons:
    x, y = poly.exterior.xy  # Extract polygon coordinates
    ax.fill(x, y, alpha=0.4, fc="lightgreen", edgecolor="green")

# Set title
plt.title("Graph Visualization with WKT Polygons", fontsize=14)

# Show the plot
plt.show()
```




```
import networkx as nx
import matplotlib.pyplot as plt
from shapely.wkt import loads
from shapely.geometry import Polygon
import textwrap

# Function to check if a node contains a WKT POLYGON
```

```
def is_wkt_polygon(node):
    return isinstance(node, str) and node.startswith("POLYGON")

# Function to shorten long URIs or WKT literals
def shorten_label(label, max_length=10):
    label = str(label)
    if is_wkt_polygon(label):
        return "[POLYGON]"  # Replace polygons with a generic label
    label=label[:max_length] + "..." if len(label) > max_length else label
    return textwrap.fill(label, width=10)  # Wrap text for better visualization

# Create a graph
G = nx.Graph()
n = 100  # Limit the number of triples to visualize
polygons = []  # Store polygons separately for plotting

# Add edges and detect polygons
for (s, p, o) in dataset.triples[:n]:
    node1 = dataset.i2e[s]
    node2 = dataset.i2e[o]
    relation = dataset.i2r[p]







    # Rename relation if it's "asWKT"
    G.add_edge(node1, node2, label=relation)

# Use a better layout
pos = nx.kamada_kawai_layout(G)

# Create figure
fig, ax = plt.subplots(figsize=(100, 100))

# Draw the graph
nx.draw(G, pos, with_labels=True, node_size=1500, font_size=10,
        edge_color="gray", node_color="skyblue", ax=ax)

# Format edge labels with better wrapping
edge_labels = {(u, v): textwrap.fill(data["label"], width=17) for u, v, data in G.edges(data=True)}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8, ax=ax)



# Plot the polygons separately
for poly in polygons:
    x, y = poly.exterior.xy  # Extract polygon coordinates
    ax.fill(x, y, alpha=0.4, fc="lightgreen", edgecolor="green", linewidth=1)

# Set title
plt.title("Graph Visualization with WKT Polygons", fontsize=14)
plt.savefig("graph_visualization.png", dpi=300, bbox_inches="tight")  # Saving with high DPI

# Show the plot
plt.show()
```



Start coding or generate with AI.

<Figure size 640x480 with 0 Axes>

```
for u, v, data in G.edges(data=True):
    print(v)
```

```
('POLYGON ((4.6672227191836703 51.819201716483 48, 4.667236120685068 51.819203883584755, 4.6672263678506125 51.81920790458488, 4.667215
('http://www.opengis.net/ont/geosparql#Geometry', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0007B6B76243781C77D67D7D535024FF', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/000B23D2133D5AE6A0CA771ECB2368B9', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/000C298591E1360885D3E6D30955787D', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/001352B2C1A6870E0597191D197C91A2', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0018E95CBF1BC5F8F65244160DEB7A6B', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/001A4EC78343E2A6285430AA75AB84D8', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0035FDC898C9AD51E6F73AA84D693EE6', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/00423F158BDE65468181F584986A0FE3', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/004E8F21D3303B5F7ED2928CCCD4C0C1', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/004F2112A9CE401964D34E04985836F3', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0054C0E067B9EA7C5FD87BCE138867A2', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0055C4F2BAF75342D13B11F1EE99F53C', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/005D0AB7D5EBC7B5F6177B1BD76BFB05', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/00653F366116E92D0FA2281D12AE92F3', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/006636341090E59EB530A0E77113C655', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0066D5C4932DFF095BBB7C54D3D0DB2B', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/006CD63AFE818DBE4BC87BBABC49BB66', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/00707A2A9E07D25A2171FBCB49552D15', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0070951943DF4E1015038E712E3170A1', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0070EACBD235ED8431BDBB3E7B8D02E0', 'iri')
```

```
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0072013AF66BDECD078F67440551F345', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0075FA0105B4BAFEA099D79D88C3242D', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0076E73707BC8A72C7A9816A384E7468', 'iri')
('http://bag.basisregistraties.overheid.nl/bag/id/geometry/0079E9A4488CE2EFD8AE5675FF1339C9', 'iri')
('POLYGON ((5.60646698164352 53.28817643955265, 5.606439267253656 53.28823750112665, 5.6062357112245325 53.288205246608605, 5.6062352
('POLYGON ((6.7140030623806455 52.758601460629194, 6.713975225446867 52.75845416329967, 6.713983031514269 52.75845362694113, 6.713970
('POLYGON ((3.890013821739893 51.504245672106556, 3.8897586119919008 51.50421362365471, 3.889794366728758 51.5041029036071, 3.8900454
('POLYGON ((5.832591882779156 50.839988745655546, 5.832626951602767 50.84004531254408, 5.83248533773343 50.84008022379452, 5.83251946
('POLYGON ((4.179941565528879 51.87961151728384, 4.179967650145183 51.87960019835143, 4.180027960349981 51.87957403948374, 4.18003380
('POLYGON ((5.741581351122694 51.75923971010251, 5.7416340016509295 51.75922862967755, 5.741613412027734 51.75919081436748, 5.7416204
('POLYGON ((5.841735035849874 51.15915911271931, 5.841870447253886 51.15900880977341, 5.841828914350672 51.1589951474488, 5.841848761
('POLYGON ((4.69765040287534 52.5255021774306, 4.697797587425536 52.52547725651568, 4.697824470358662 52.52553632637423, 4.6978209244
('POLYGON ((5.741685803897686 51.759795405950975, 5.741701057630571 51.75982325115344, 5.741763747487607 51.75981412673893, 5.7417748
('POLYGON ((4.603853421413119 51.64607408085403, 4.603872552836769 51.646084860719064, 4.603883237725863 51.646099476229196, 4.603884
('POLYGON ((4.358704629959813 52.01195277394727, 4.358806367919676 52.01184852953267, 4.358832278137911 52.011858868300685, 4.3588799
('POLYGON ((3.6178228275910036 51.49914550071893, 3.6178428980099957 51.499130925631896, 3.617893565079056 51.49915406161168, 3.61791
('POLYGON ((6.559792183564186 52.99258286532555, 6.559780426303763 52.992583205329574, 6.559780268809045 52.99258122972964, 6.5597462
('POLYGON ((4.670252041975575 52.11978439231835, 4.670191884822936 52.119770687761104, 4.670164316975579 52.11973067496852, 4.6701891
('POLYGON ((4.433489848294623 52.01330837965059, 4.433482299818166 52.01330850724876, 4.433446859788846 52.013309109956595, 4.4334181
('POLYGON ((5.80024012809201 53.201930009880535, 5.8002299978671505 53.201873740500176, 5.800231119522115 53.201873664750686, 5.80031
('POLYGON ((3.4492374899834455 51.273360308750064, 3.449278460151831 51.27339879803904, 3.4492328337686335 51.27342018246525, 3.44921
('POLYGON ((5.81994721067153 50.76496574877617, 5.819987140516807 50.765004134601, 5.819961797625884 50.76501470168929, 5.8199538714
('POLYGON ((5.125317562026319 52.090686546957954, 5.125292083269841 52.090729273644925, 5.125244882600087 52.09071880572856, 5.125205
('POLYGON ((5.709874466550587 51.58591001960596, 5.709979516494417 51.58568593921787, 5.710564326263988 51.58579187142961, 5.71053450
('POLYGON ((6.569146505908022 53.219783739495284, 6.568911825245705 53.219731637586555, 6.568902570682843 53.21972008239364, 6.568927
('POLYGON ((7.009166921370495 53.24767673686859, 7.009123955711112 53.24771362493977, 7.009130503227163 53.24771641233495, 7.00870626
('POLYGON ((5.793201150341866 53.201349191302064, 5.793166798904708 53.20135640631676, 5.793110284050815 53.20136995039768, 5.7930279
('POLYGON ((4.796352195586051 53.05546932014179, 4.79643401676529 53.055505605093046, 4.796367126760476 53.055560342785014, 4.7962355
```

Entity Frequency Analysis

```
from collections import Counter

# Extract entities (converting tensor to integer)
entities = [int(triple[0]) for triple in dataset.triples] + [int(triple[2]) for triple in dataset.triples]

# Count occurrences
entity_counts = Counter(entities)

# Display actual most frequent entities
print("Most common entities:", entity_counts.most_common(10))

# Check unique and rare entities again
rare_entities = [e for e, c in entity_counts.items() if c == 1]
print(f"Number of rare entities (appear once): {len(rare_entities)}")
```

```
Most common entities: [(201822, 65576), (211084, 60996), (341077, 23232), (28996, 18146), (23569, 16868), (25943, 13544), (79498, 939
Number of rare entities (appear once): 250658
```

Start coding or generate with AI.

```
import numpy as np
sorted_entities = entity_counts.most_common()

# Compute cumulative sum
frequencies = np.array([count for _, count in sorted_entities])
cumulative_freq = np.cumsum(frequencies) / sum(frequencies)  # Normalize to [0,1]

# Find 80% cutoff point
cutoff_index = np.searchsorted(cumulative_freq, 0.8)
top_entities = sorted_entities[:cutoff_index]

print(f"Number of entities covering 80% of occurrences: {len(top_entities)}")
print("Example top entities:", top_entities[:10])
```

```
Number of entities covering 80% of occurrences: 68865
Example top entities: [(201822, 65576), (211084, 60996), (341077, 23232), (28996, 18146), (23569, 16868), (25943, 13544), (79498, 939
```

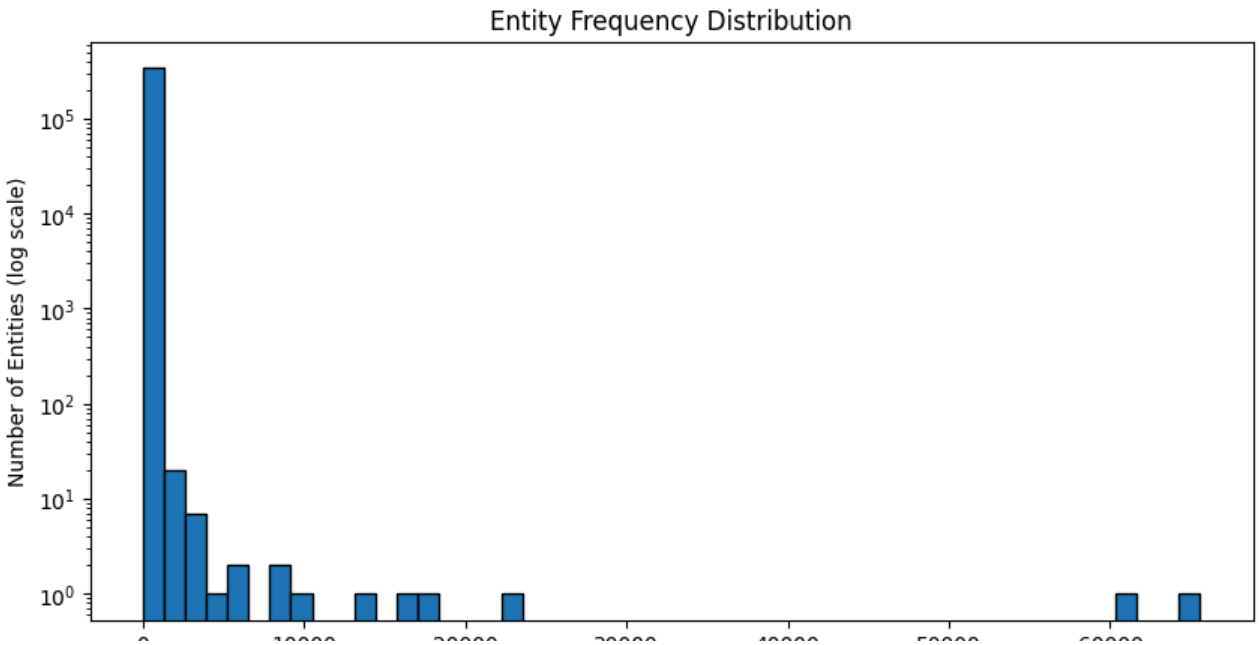
```
# Get the top 10 most common entity indices


# Print entity names using i2e mapping
for entity_id, count in top_entities[:10]:
    entity_name = dataset.i2e[entity_id]  # Convert ID to actual entity
    print(f"Entity: {entity_name} (ID: {entity_id}) appears {count} times.")
```

```
Entity: ('http://www.opengis.net/ont/geosparql#Geometry', 'iri') (ID: 201822) appears 65576 times.
Entity: ('https://data.labs.pdok.nl/rce/def/Afbeelding', 'iri') (ID: 211084) appears 60996 times.
Entity: ('zwart wit negatief', '@nl-nl') (ID: 341077) appears 23232 times.
Entity: ('Fotocollectie', '@nl-nl') (ID: 28996) appears 18146 times.
Entity: ('Collectie gebouwd', '@nl-nl') (ID: 23569) appears 16868 times.
Entity: ('Dukker, G.J.', 'none') (ID: 25943) appears 13544 times.
Entity: ('Wal, A.J. van der', 'none') (ID: 79498) appears 9396 times.
Entity: ('true', 'http://www.w3.org/2001/XMLSchema#boolean') (ID: 337400) appears 8230 times.
Entity: ('Agrarisch Erfgoed - Fotocollectie SHBO', '@nl-nl') (ID: 19920) appears 8090 times.
Entity: ('Digitale opname', '@nl-nl') (ID: 25196) appears 6023 times.
```

```
# Get statistics
total_entities = len(entity_counts)
rare_entity_count = len(rare_entities)
rare_entity_ratio = rare_entity_count / total_entities

print(f"Total Entities: {total_entities}")
print(f"Rare Entities (appear once): {rare_entity_count}")
print(f"Rare Entity Ratio: {rare_entity_ratio:.2%}")
```

```
Total Entities: 341270
Rare Entities (appear once): 250658
Rare Entity Ratio: 73.45%
```

```
import matplotlib.pyplot as plt

# Get frequencies of all entities
entity_frequencies = list(entity_counts.values())

# Plot histogram (log-scale for better visualization)
plt.figure(figsize=(10, 5))
plt.hist(entity_frequencies, bins=50, log=True, edgecolor="black")
plt.xlabel("Entity Frequency")
plt.ylabel("Number of Entities (log scale)")
plt.title("Entity Frequency Distribution")
plt.show()
```




```
import networkx as nx

# Create a graph where entities are nodes and relations are edges
G = nx.Graph()
G.add_edges_from([(t[0], t[2]) for t in dataset.triples])

# Compute connectivity statistics
num_components = nx.number_connected_components(G)
largest_component = max(nx.connected_components(G), key=len)
largest_component_ratio = len(largest_component) / total_entities

print(f"Number of Connected Components: {num_components}")

print(f"Largest Component Size: {len(largest_component)} ({largest_component_ratio:.2%} of all nodes)")
print(f"Average Node Degree: {sum(dict(G.degree()).values()) / len(G.nodes())}")
```

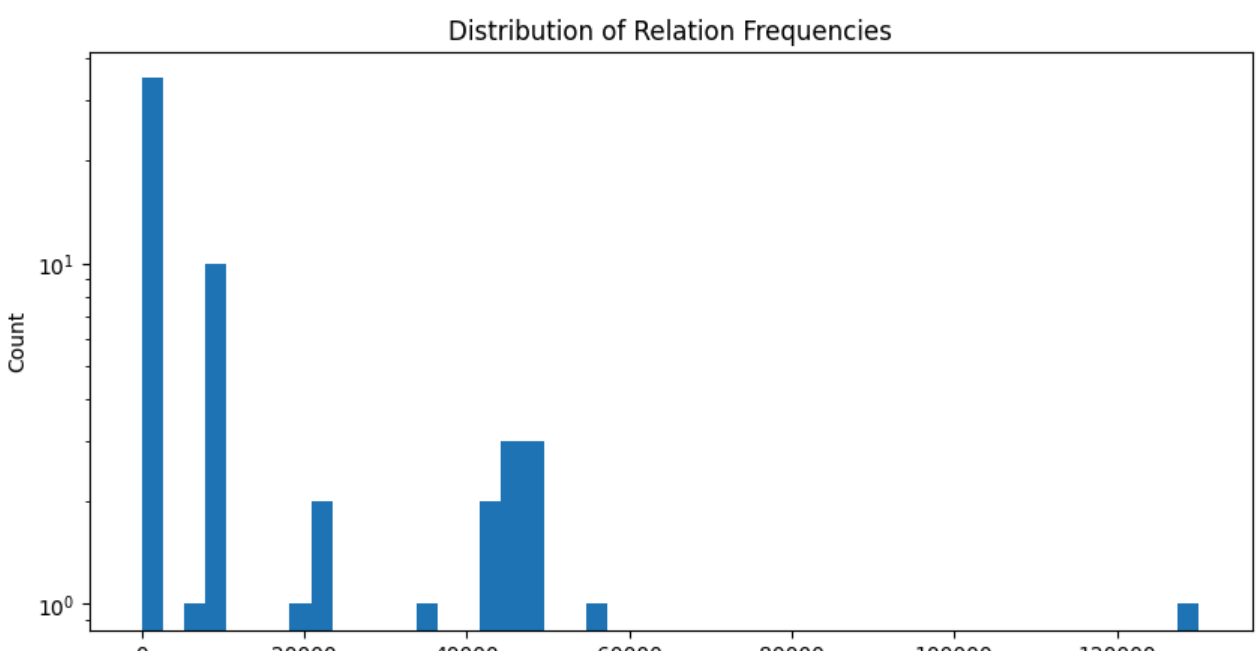
```
Number of Connected Components: 777124
Largest Component Size: 2 (0.00% of all nodes)
```

```
# Count relation occurrences
relation_counts = Counter([int(triple[1]) for triple in dataset.triples])

# Get most common relations
most_common_relations = relation_counts.most_common(10)

least_common_relations = sorted(relation_counts.items(), key=lambda x: x[1])[:10]

# Display results
print("Most common relations:")
for relation_id, count in most_common_relations:
    print(f"Relation: {dataset.i2r[relation_id]} (ID: {relation_id}) appears {count} times.")

print("\nLeast common relations:")
for relation_id, count in least_common_relations:
    print(f"Relation: {dataset.i2r[relation_id]} (ID: {relation_id}) appears {count} times.")
```

```
Most common relations:
Relation: http://www.w3.org/1999/02/22-rdf-syntax-ns#type (ID: 31) appears 130126 times.
Relation: http://purl.org/dc/terms/description (ID: 9) appears 57042 times.
Relation: http://xmlns.com/foaf/0.1/depicts (ID: 41) appears 47872 times.
Relation: http://xmlns.com/foaf/0.1/depiction (ID: 40) appears 47872 times.
Relation: http://purl.org/dc/terms/isPartOf (ID: 10) appears 47384 times.
Relation: https://data.labs.pdok.nl/rce/def/locator (ID: 52) appears 46123 times.
Relation: http://dbpedia.org/ontology/thumbnail (ID: 6) appears 46108 times.
Relation: http://purl.org/dc/terms/creator (ID: 8) appears 45111 times.
```

```
Relation: http://purl.org/dc/terms/created (ID: 7) appears 44216 times.
Relation: https://data.labs.pdok.nl/rce/def/fotograaf (ID: 45) appears 43964 times.

Least common relations:
Relation: https://data.labs.pdok.nl/rce/def/graveur (ID: 46) appears 1 times.
Relation: http://www.w3.org/2003/01/geo/wgs84_pos#alt (ID: 36) appears 3 times.
Relation: https://data.labs.pdok.nl/rce/def/ontwerper (ID: 53) appears 7 times.
Relation: https://data.labs.pdok.nl/rce/def/reprofotograaf (ID: 54) appears 8 times.
Relation: https://data.labs.pdok.nl/rce/def/schilder (ID: 56) appears 36 times.
Relation: http://www.geonames.org/ontology#officialName (ID: 21) appears 55 times.
Relation: http://dbpedia.org/ontology/location (ID: 3) appears 156 times.
Relation: https://data.labs.pdok.nl/rce/def/huisnummerToevoeging (ID: 49) appears 212 times.
Relation: http://dbpedia.org/ontology/neighbourhood (ID: 5) appears 258 times.
Relation: http://purl.org/dc/terms/spatial (ID: 11) appears 260 times.
```

```
# Get relation frequencies
relation_frequencies = list(relation_counts.values())

# Plot histogram
plt.figure(figsize=(10,5))
plt.hist(relation_frequencies, bins=50, log=True)  # Log scale for better visualization
plt.xlabel("Relation Frequency")
plt.ylabel("Count")
plt.title("Distribution of Relation Frequencies")
plt.show()
```



resembles a Zipfian distribution

```
relation_usage = {int(relation): set() for relation in relation_counts.keys()}  # Ensure keys are integers

for head, rel, tail in dataset.triples:
    rel = int(rel)  # Convert tensor to integer
    head = int(head)
    tail = int(tail)

    relation_usage[rel].add(head)
    relation_usage[rel].add(tail)

relation_connectivity = {rel: len(entities) for rel, entities in relation_usage.items()}
sorted_connectivity = sorted(relation_connectivity.items(), key=lambda x: x[1], reverse=True)

# Print top 10 most connected relations
print("Top 10 most connected relations:")
for rel, count in sorted_connectivity[:10]:
```

```
    print(f"Relation: {dataset.i2r[rel]} connects {count} unique entities")
```

```
Top 10 most connected relations:
Relation: http://www.w3.org/1999/02/22-rdf-syntax-ns#type connects 129374 unique entities
Relation: https://data.labs.pdok.nl/rce/def/locator connects 92214 unique entities
Relation: http://dbpedia.org/ontology/thumbnail connects 92169 unique entities
Relation: http://purl.org/dc/terms/description connects 89927 unique entities
Relation: http://xmlns.com/foaf/0.1/depicts connects 56268 unique entities
Relation: http://xmlns.com/foaf/0.1/depiction connects 56268 unique entities
Relation: http://purl.org/dc/terms/isPartOf connects 47422 unique entities
Relation: http://purl.org/dc/terms/creator connects 45738 unique entities
Relation: http://purl.org/dc/terms/created connects 45298 unique entities
Relation: https://data.labs.pdok.nl/rce/def/fotograaf connects 44428 unique entities
```

```
rare_relations = [rel for rel, count in relation_counts.items() if count == 1]
print(f"Number of relations that appear only once: {len(rare_relations)}")
print(f"Percentage of sparse relations: {len(rare_relations) / len(relation_counts) * 100:.2f}%")
```

```
Number of relations that appear only once: 1
Percentage of sparse relations: 1.67%
```

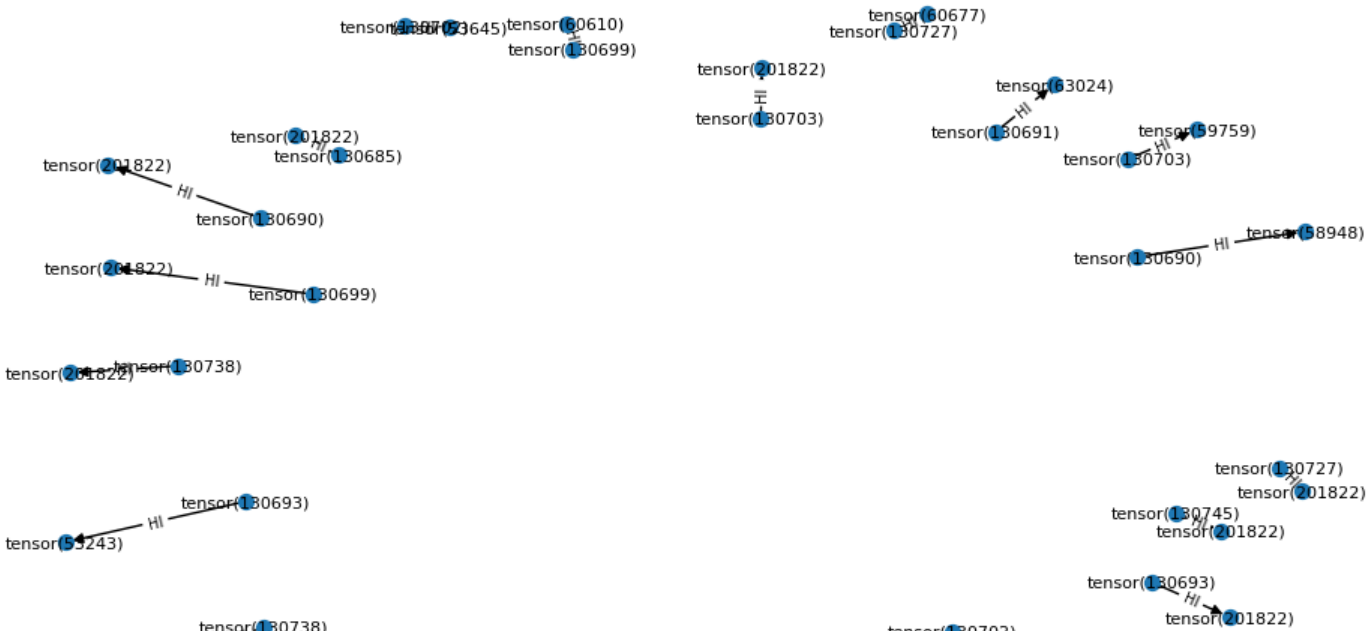
```
from collections import defaultdict
import itertools

entity_relation_map = defaultdict(set)

for head, rel, tail in dataset.triples:
    rel = int(rel)  # Convert tensor to integer
    head = int(head)
    tail = int(tail)
    entity_relation_map[head].add(rel)

co_occurrence_counts = Counter()

for relations in entity_relation_map.values():
    for rel1, rel2 in itertools.combinations(relations, 2):
        co_occurrence_counts[(rel1, rel2)] += 1

# Print most common relation pairs
print("Top 10 most co-occurring relation pairs:")
for (rel1, rel2), count in co_occurrence_counts.most_common(10):
    print(f"Relations: {dataset.i2r[rel1]} & {dataset.i2r[rel2]} appear together {count} times.")
```

```
Top 10 most co-occurring relation pairs:
Relations: http://purl.org/dc/terms/description & http://xmlns.com/foaf/0.1/depicts appear together 47872 times.
Relations: http://purl.org/dc/terms/description & http://www.w3.org/1999/02/22-rdf-syntax-ns#type appear together 47872 times.
Relations: http://xmlns.com/foaf/0.1/depicts & http://www.w3.org/1999/02/22-rdf-syntax-ns#type appear together 47872 times.
Relations: http://purl.org/dc/terms/description & http://purl.org/dc/terms/isPartOf appear together 47384 times.
Relations: http://purl.org/dc/terms/isPartOf & http://xmlns.com/foaf/0.1/depicts appear together 47384 times.
Relations: http://purl.org/dc/terms/isPartOf & http://www.w3.org/1999/02/22-rdf-syntax-ns#type appear together 47384 times.
Relations: http://purl.org/dc/terms/description & https://data.labs.pdok.nl/rce/def/locator appear together 46123 times.
Relations: http://xmlns.com/foaf/0.1/depicts & https://data.labs.pdok.nl/rce/def/locator appear together 46123 times.
Relations: https://data.labs.pdok.nl/rce/def/locator & http://www.w3.org/1999/02/22-rdf-syntax-ns#type appear together 46123 times.
Relations: http://dbpedia.org/ontology/thumbnail & http://purl.org/dc/terms/description appear together 46108 times.
```

```
# Create a directed graph
G = nx.DiGraph()
G.add_edges_from([(triple[0], triple[2]) for triple in dataset.triples])

# Compute graph properties
print(f"Number of Connected Components: {nx.number_weakly_connected_components(G)}")
print(f"Average Node Degree: {sum(dict(G.degree()).values()) / len(G.nodes())}")
```

```
Number of Connected Components: 777124
Average Node Degree: 1.0
```

```
# Draw a small subgraph for visualization
plt.figure(figsize=(10,6))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=50, font_size=8)
edge_labels = {(u, v):"HI" for u, v, d in G.edges(data=True)}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=7)
plt.show()
```




```
# Convert to a set to remove duplicates
unique_triples = set(dataset.triples)

# Check how many duplicates were present
num_duplicates = len(dataset.triples) - len(unique_triples)
print(f"Number of duplicate triples: {num_duplicates}")
```

```
Number of duplicate triples: 0
```

```
type(dataset.triples)
```

```
torch.Tensor
```

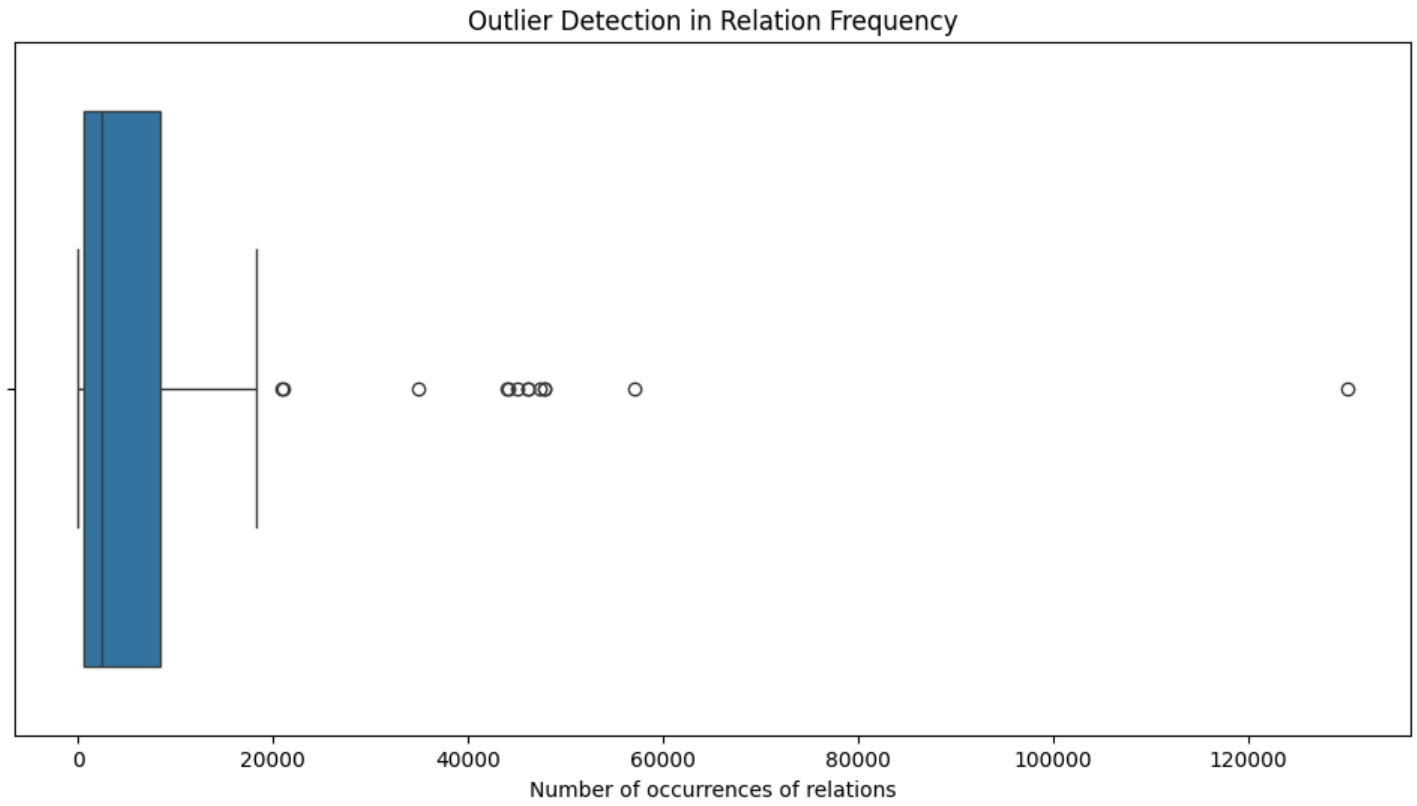
```
import torch


# Check for missing values in dataset.triples
missing_triples = [triple for triple in dataset.triples if torch.isnan(triple).any() or torch.isinf(triple).any()]
print(f"Number of missing triples: {len(missing_triples)}")
```

```
Number of missing triples: 0
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-55-fd59ec1a98de> in <cell line: 0>()
      4
      5 # If needed, remove them
----> 6 dataset.triples = [triple for triple in dataset.triples if None not in triple]
      7 print(f"Dataset size after removing missing triples: {len(dataset.triples)}")

 1 frames
/usr/local/lib/python3.11/dist-packages/torch/_tensor.py in __contains__(self, element)
   1223             return bool((element == self).any().item())  # type: ignore[union-attr]
   1224
-> 1225         raise RuntimeError(
   1226             f"Tensor.__contains__ only supports Tensor or scalar, but you passed in a {type(element)}."
   1227         )

RuntimeError: Tensor.__contains__ only supports Tensor or scalar, but you passed in a <class 'NoneType'>.
```

```
type(dataset.triples[1] )
```

```
torch.Tensor
```

```
# Count occurrences of each relation
relation_counts = Counter(int(triple[1].item()) for triple  in dataset.triples)

# Plot distribution
plt.figure(figsize=(12,6))
sns.boxplot(x=list(relation_counts.values()))
plt.xlabel("Number of occurrences of relations")
plt.title("Outlier Detection in Relation Frequency")
plt.show()

# Find relations appearing significantly more or less
q1 = np.percentile(list(relation_counts.values()), 25)
q3 = np.percentile(list(relation_counts.values()), 75)
iqr = q3 - q1

# Define outliers as relations occurring way outside normal range
outlier_threshold = q3 + 1.5 * iqr
outlier_relations = {rel: count for rel, count in relation_counts.items() if count > outlier_threshold}
```




```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-60-4fd82e9b9687> in <cell line: 0>()
     18 outlier_relations = {rel: count for rel, count in relation_counts.items() if count > outlier_threshold}
```

```
outlier_keys = list(outlier_relations.keys())  # Extract relation indices
outlier_names = [dataset.i2r[key] for key in outlier_keys]  # Convert indices to names

print(f"Outlier relations (very high frequency): {outlier_names}")
```

```
Outlier relations (very high frequency): ['http://www.opengis.net/ont/geosparql#asWKT', 'http://www.w3.org/1999/02/22-rdf-syntax-ns#t
```

```
outlier_keys = list(outlier_relations.keys())  # Extract relation indices

# Extract only the relation keys from the least common ones
low_freq_relations = [relation for relation, _ in relation_counts.most_common()[-10:]]

# Convert indices to relation names
outlier_names = [dataset.i2r[key] for key in low_freq_relations]

print(f"Outlier relations (very low frequency): {outlier_names}")
```

```
Outlier relations (very low frequency): ['http://purl.org/dc/terms/spatial', 'http://dbpedia.org/ontology/neighbourhood', 'https://da
11: 260 occurrences
5: 258 occurrences
49: 212 occurrences
3: 156 occurrences
21: 55 occurrences
56: 36 occurrences
54: 8 occurrences
53: 7 occurrences
36: 3 occurrences
46: 1 occurrences
```

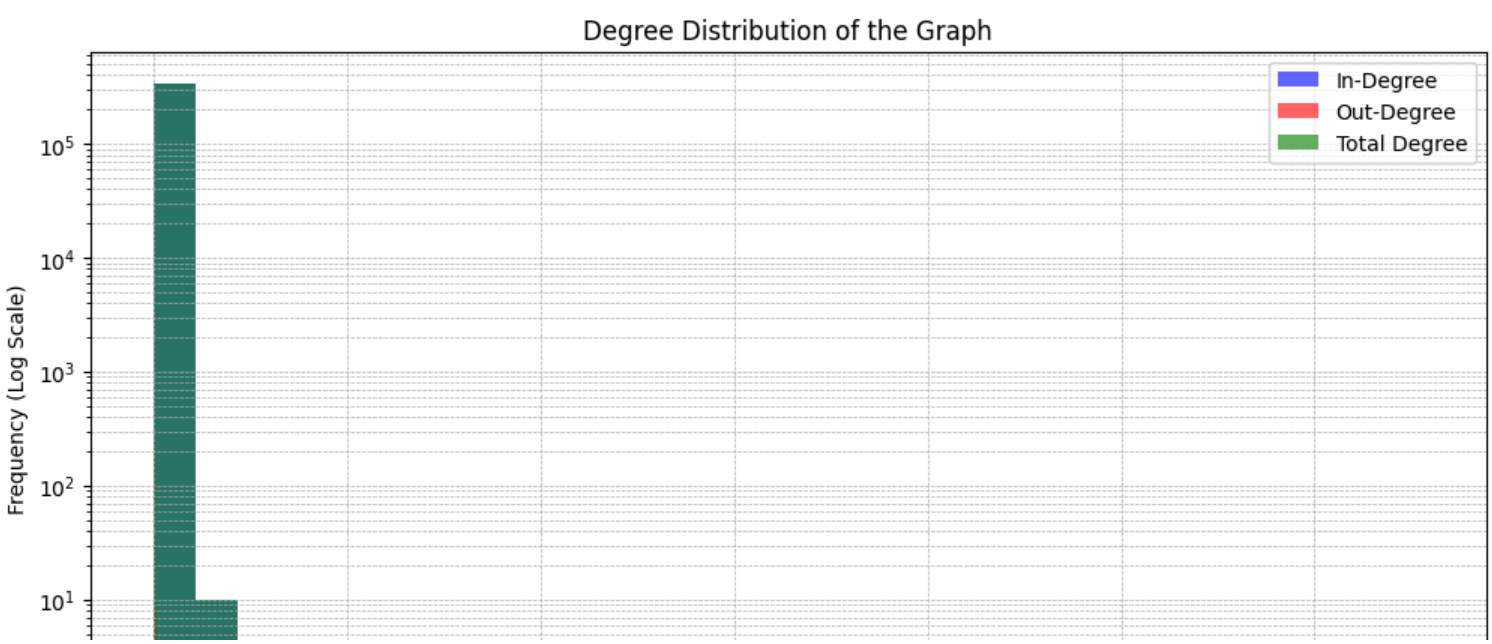
```
import networkx as nx

# Create a directed graph
```

```
G = nx.DiGraph()

# Add edges with relations as edge attributes
G.add_edges_from([(int(triple[0].item()), int(triple[2].item()), {"relation": triple[1]}) for triple in dataset.triples])

# Find all entities appearing in triples (convert tensors to integers)
all_entities = set(int(triple[0].item()) for triple in dataset.triples) | set(int(triple[2].item()) for triple in dataset.triples)

# Add all entities to the graph explicitly
G.add_nodes_from(all_entities)

# Now find isolated nodes
connected_entities = set(G.nodes())
isolated_nodes = all_entities - connected_entities

print(f"Number of isolated nodes: {len(isolated_nodes)}")  # Should now give correct result
if G.number_of_nodes() > 0:
    node_max_in = max(G.in_degree(), key=lambda x: x[1], default=(None, 0))   # Node with max in-degree
    node_max_out = max(G.out_degree(), key=lambda x: x[1], default=(None, 0)) # Node with max out-degree
    node_max_total = max(G.degree(), key=lambda x: x[1], default=(None, 0))  # Node with max total degree

    print(f"Node with max in-degree: {node_max_in[0]} (Degree: {node_max_in[1]})")
    print(f"Node with max out-degree: {node_max_out[0]} (Degree: {node_max_out[1]})")
    print(f"Node with max total degree: {node_max_total[0]} (Degree: {node_max_total[1]})")
else:
    print("Graph has no nodes.")
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-114-d894f41a09fc> in <cell line: 0>()
      5
      6 # Add edges with relations as edge attributes
----> 7 G.add_edges_from([(int(triple[0].item()), int(triple[2].item()), {"relation": triple[1]}) for triple in dataset.triples])
      8
      9 # Find all entities appearing in triples (convert tensors to integers)

<ipython-input-114-d894f41a09fc> in <listcomp>(.0)
      5
      6 # Add edges with relations as edge attributes
----> 7 G.add_edges_from([(int(triple[0].item()), int(triple[2].item()), {"relation": triple[1]}) for triple in dataset.triples])
      8
      9 # Find all entities appearing in triples (convert tensors to integers)

KeyboardInterrupt:
```

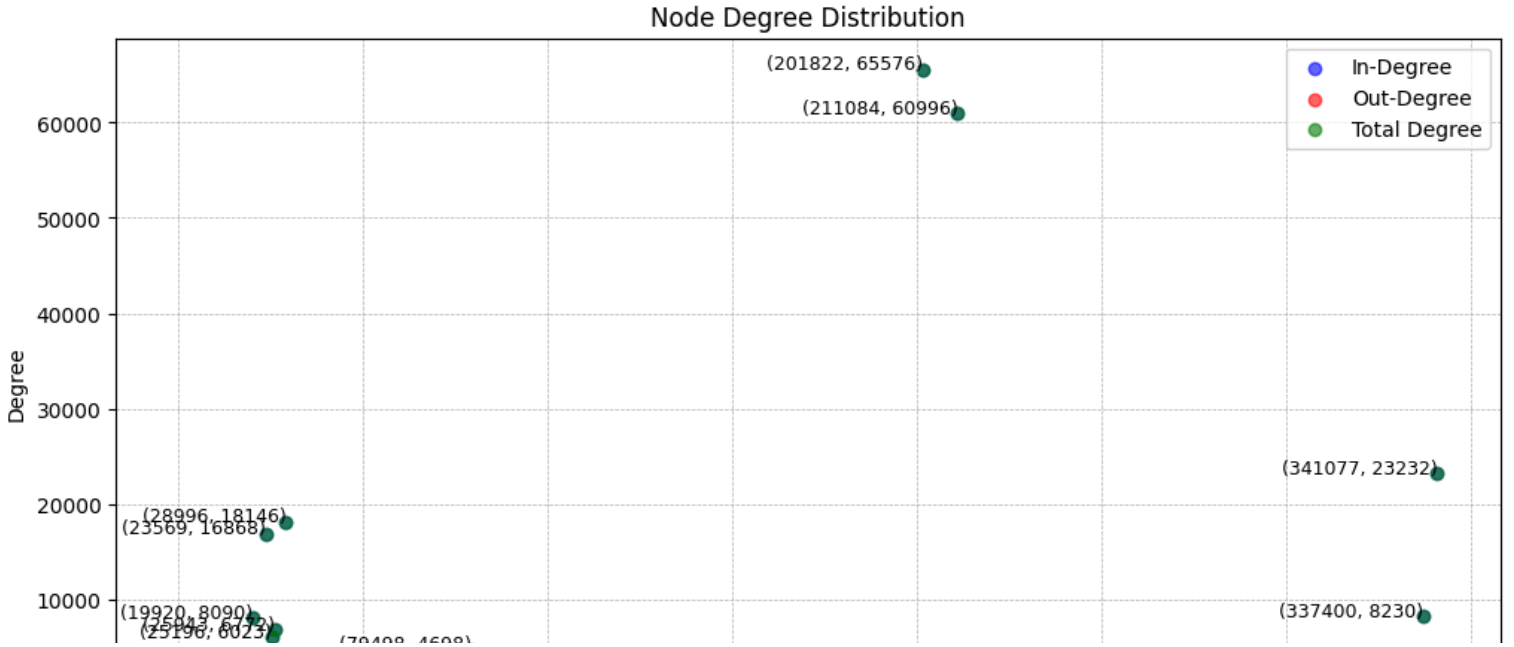
```
import networkx as nx
import matplotlib.pyplot as plt

# Create a directed graph
G = nx.DiGraph()

# Add edges with relations as edge attributes
G.add_edges_from([(int(triple[0].item()), int(triple[2].item()), {"relation": triple[1]}) for triple in dataset.triples])

# Compute degrees
in_degrees = [G.in_degree(n) for n in G.nodes()]
out_degrees = [G.out_degree(n) for n in G.nodes()]
total_degrees = [G.degree(n) for n in G.nodes()]

# Plot degree distributions
plt.figure(figsize=(12, 6))

plt.hist(in_degrees, bins=30, alpha=0.6, label="In-Degree", color="blue", log=True)
plt.hist(out_degrees, bins=30, alpha=0.6, label="Out-Degree", color="red", log=True)
plt.hist(total_degrees, bins=30, alpha=0.6, label="Total Degree", color="green", log=True)

plt.xlabel("Degree")
plt.ylabel("Frequency (Log Scale)")
```

```
plt.title("Degree Distribution of the Graph")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)

plt.show()
```




```
import networkx as nx
import matplotlib.pyplot as plt

# Compute degrees
node_indices = list(G.nodes())  # Get the node indices
in_degrees = [G.in_degree(n) for n in node_indices]
out_degrees = [G.out_degree(n) for n in node_indices]
total_degrees = [G.degree(n) for n in node_indices]

# Plot scatter graph
plt.figure(figsize=(12, 6))

scatter_in = plt.scatter(node_indices, in_degrees, label="In-Degree", color="blue", alpha=0.6)
scatter_out = plt.scatter(node_indices, out_degrees, label="Out-Degree", color="red", alpha=0.6)
scatter_total = plt.scatter(node_indices, total_degrees, label="Total Degree", color="green", alpha=0.6)

# Annotate points where the total degree > 10
for i, (node, degree) in enumerate(zip(node_indices, total_degrees)):
    if degree > 10:
        plt.text(node, degree, f"({node}, {degree})", fontsize=9, color="black", ha="right")

plt.xlabel("Node Index")
plt.ylabel("Degree")
plt.title("Node Degree Distribution")
plt.legend()
plt.grid(True, linestyle="--", linewidth=0.5)

plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Creating legend with loc="best" can be slow with
  fig.canvas.print_figure(bytes_io, **kw)
```




```
# Annotate points where the total degree > 10
for i, (node, degree) in enumerate(zip(node_indices, total_degrees)):
    if degree > 10000:
        plt.text(node, degree, f"({node}, {degree})", fontsize=9, color="black", ha="right")

plt.xlabel("Node Index")
plt.ylabel("Degree")
plt.title("Node Degree Distribution")
plt.legend()
plt.grid(True, linestyle="--", linewidth=0.5)

plt.show()
```

```
<ipython-input-7-74c48fd29b45>:9: UserWarning: No artists with labels found to put in legend.  Note that artists whose label start wi
  plt.legend()
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/IPython/core/formatters.py in __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

 7 frames
/usr/local/lib/python3.11/dist-packages/matplotlib/backends/backend_agg.py in __init__(self, width, height, dpi)
     67         self.width = width
     68         self.height = height
---> 69         self._renderer = _RendererAgg(int(width), int(height), dpi)
     70         self._filter_renderers = []
     71

ValueError: Image size of 169174263x24236962 pixels is too large. It must be less than 2^23 in each direction.

<Figure size 640x480 with 1 Axes>
```

```
print(dataset.i2e[201822 ],dataset.i2e[288128 ],dataset.i2e[201822  ])
```

```
('http://www.opengis.net/ont/geosparql#Geometry', 'iri') ('https://data.labs.pdok.nl/rce/id/monument/8336', 'iri') ('http://www.openg
```

```
relations = set(int(triple[1].item()) for triple in dataset.triples)

print(f"Number of unique relations: {len(relations)}")
```

```
Number of unique relations: 60
```

```
relation_names = [dataset.i2r[rel] for rel in relations]

print("Relations:", relation_names)
```

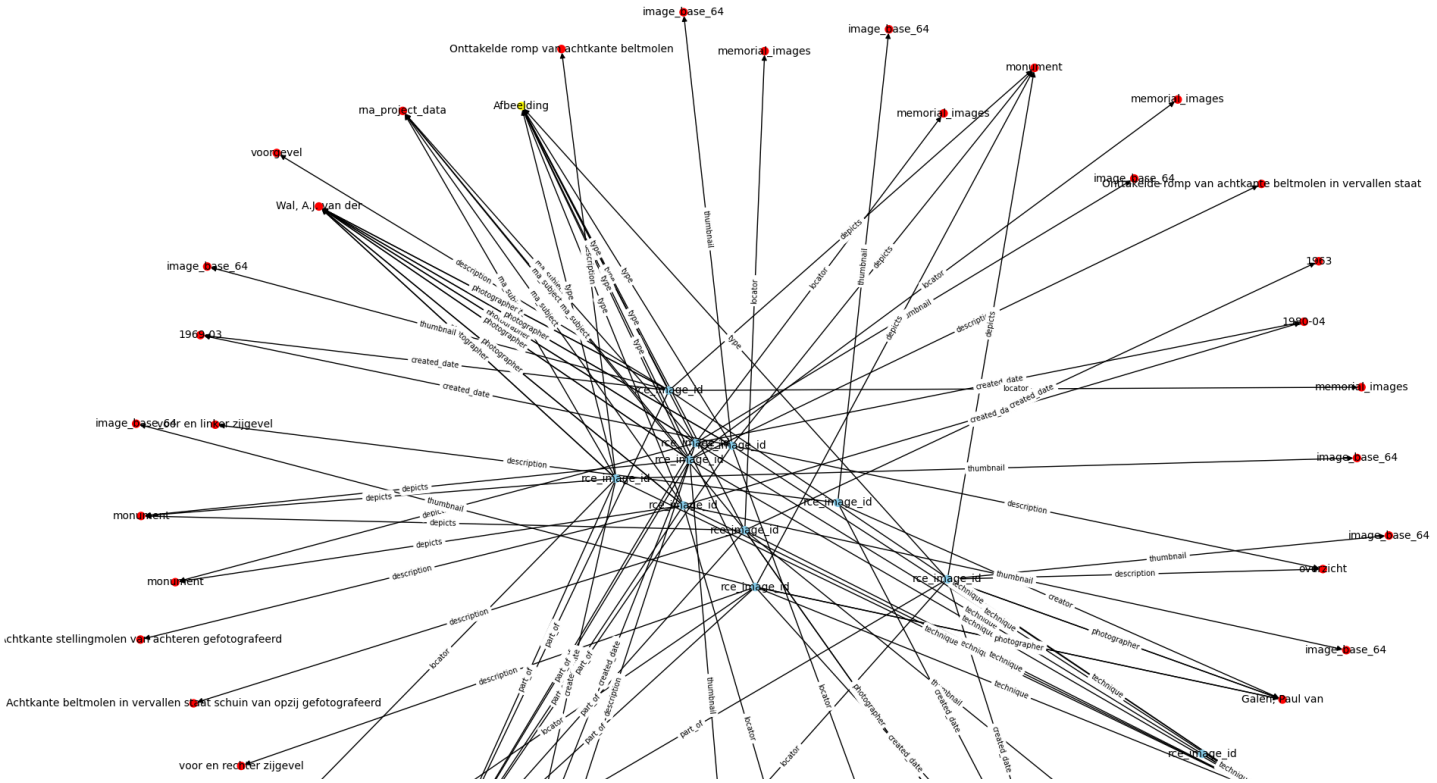
```
Relations: ['http://data.pdok.nl/def/pdok#asWKT-RD', 'http://dbpedia.org/ontology/city', 'http://dbpedia.org/ontology/codeNationalMor
```

```
relation_dict = {
    "http://data.pdok.nl/def/pdok#asWKT-RD": "geometry",
    "http://dbpedia.org/ontology/city": "city",
    "http://dbpedia.org/ontology/codeNationalMonument": "monument_code",
    "http://dbpedia.org/ontology/location": "location",
    "http://dbpedia.org/ontology/name": "name",
    "http://dbpedia.org/ontology/neighbourhood": "neighbourhood",
    "http://dbpedia.org/ontology/thumbnail": "thumbnail",
    "http://purl.org/dc/terms/created": "created_date",
    "http://purl.org/dc/terms/creator": "creator",
    "http://purl.org/dc/terms/description": "description",
    "http://purl.org/dc/terms/isPartOf": "part_of",
    "http://purl.org/dc/terms/spatial": "spatial",
    "http://schema.org/dateCreated": "date_created",
    "http://schema.org/roleName": "role_name",
    "http://www.geonames.org/ontology#alternateName": "alt_name",
    "http://www.geonames.org/ontology#countryCode": "country_code",
    "http://www.geonames.org/ontology#featureClass": "feature_class",
    "http://www.geonames.org/ontology#featureCode": "feature_code",
    "http://www.geonames.org/ontology#locationMap": "location_map",
    "http://www.geonames.org/ontology#name": "name",
    "http://www.geonames.org/ontology#nearbyFeatures": "nearby_features",
    "http://www.geonames.org/ontology#officialName": "official_name",
    "http://www.geonames.org/ontology#parentADM1": "admin1_parent",
    "http://www.geonames.org/ontology#parentADM2": "admin2_parent",
    "http://www.geonames.org/ontology#parentCountry": "parent_country",
    "http://www.geonames.org/ontology#parentFeature": "parent_feature",
    "http://www.geonames.org/ontology#population": "population",
    "http://www.geonames.org/ontology#wikipediaArticle": "wikipedia",
    "http://www.opengis.net/ont/geosparql#asWKT": "geometry",
    "http://www.opengis.net/ont/geosparql#hasGeometry": "has_geometry",
    "http://www.opengis.net/ont/geosparql#sfWithin": "within",
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#type": "type",
    "http://www.w3.org/2000/01/rdf-schema#isDefinedBy": "defined_by",
    "http://www.w3.org/2000/01/rdf-schema#label": "label",
    "http://www.w3.org/2000/01/rdf-schema#seeAlso": "see_also",
    "http://www.w3.org/2002/07/owl#sameAs": "same_as",
    "http://www.w3.org/2003/01/geo/wgs84_pos#alt": "altitude",
    "http://www.w3.org/2006/vcard/ns#hasStreetAddress": "street_address",
    "http://www.w3.org/2006/vcard/ns#postal-code": "postal_code",
    "http://www.w3.org/2006/vcard/ns#street-address": "street_address",
    "http://xmlns.com/foaf/0.1/depiction": "depiction",
    "http://xmlns.com/foaf/0.1/depicts": "depicts",
    "http://xmlns.com/foaf/0.1/name": "name",
    "https://data.labs.pdok.nl/rce/def/bouwjaar": "construction_year",
    "https://data.labs.pdok.nl/rce/def/complexnummer": "complex_number",
    "https://data.labs.pdok.nl/rce/def/fotograaf": "photographer",
    "https://data.labs.pdok.nl/rce/def/graveur": "engraver",
    "https://data.labs.pdok.nl/rce/def/huisnummer": "house_number",
    "https://data.labs.pdok.nl/rce/def/huisnummerCompleet": "full_house_number",
    "https://data.labs.pdok.nl/rce/def/huisnummerToevoeging": "house_number_suffix",
    "https://data.labs.pdok.nl/rce/def/internComplexNummers": "internal_complex_numbers",
    "https://data.labs.pdok.nl/rce/def/isFree": "is_free",
    "https://data.labs.pdok.nl/rce/def/locator": "locator",
    "https://data.labs.pdok.nl/rce/def/ontwerper": "designer",
    "https://data.labs.pdok.nl/rce/def/reprofotograaf": "reproduction_photographer",
```

```
    "https://data.labs.pdok.nl/rce/def/rnaSubject": "rna_subject",
    "https://data.labs.pdok.nl/rce/def/schilder": "painter",
    "https://data.labs.pdok.nl/rce/def/techniek": "technique",
    "https://data.labs.pdok.nl/rce/def/tekenaar": "drafter",
    "https://data.pldn.nl/cbs/wijken-buurten/def/cbs#regiocode": "region_code"
}
head_dict = {
    "https://data.labs.pdok.nl/.well-known/genid/": "genid",
    "https://data.pldn.nl/cbs/wijken-buurten/regios/2016/id/land-geografisch/": "geo_region",
    "http://sws.geonames.org/": "geonames",
    "https://data.labs.pdok.nl/cbs/id/gemeente/": "municipality",
    "https://data.pldn.nl/cbs/wijken-buurten/regios/2016/id/geometry/": "geometry",
    "https://data.labs.pdok.nl/rce/id/image/": "image_by_rce",
    "http://bag.basisregistraties.overheid.nl/bag/id/geometry/": "bag_geometry",
    "https://data.labs.pdok.nl/rce/id/monument/": "monument",
    "http://www.opengis.net/ont/geosparql": "GeoSparql",
    "https://data.labs.pdok.nl/rce/def/Afbeelding": "Afbeelding",
    "https://images.memorix.nl/rce/download/fullsize/": "fullsize_image_download",
    "http://data.cultureelerfgoed.nl/semnet/": "cultureel_erfgoed",
    "http://nl.wikipedia.org/wiki/": "wikipedia",
    "http://www.geonames.org/ontology#Feature": "OntologyFeature",
    "https://data.pldn.nl/cbs/wijken-buurten/def/": "wijken_buurten_def",
    "https://images.memorix.nl/rce/download/fullsize/": "memorial_images",
    "http://www.opengis.net/ont/": "open_gis_ontology",
    "http://sws.geonames.org/": "geonames_sws",
    "http://data.cultureelerfgoed.nl/semnet/": "cultureel_erfgoed_semnet",
    "http://www.geonames.org/": "geonames",
    "http://www.rnaproject.org/data/": "rna_project_data",
    "https://data.labs.pdok.nl/cbs/id/gemeente/": "gemeente",
    "https://data.pldn.nl/cbs/wijken-buurten/regios/2016/id/geometry/": "geometry",
    "http://dbpedia.org/resource/": "dbpedia_resource",
    "https://data.labs.pdok.nl/rce/id/image/": "rce_image_id",
    "://": "unknown_protocol",
    "https://data.labs.pdok.nl/rce/def/": "rce_def",
    "http://wikipedia.org/": "wikipedia",
    "http://bag.basisregistraties.overheid.nl/bag/id/geometry/": "bag_geometry",
    "https://data.labs.pdok.nl/rce/id/monument/": "monument"
}



import networkx as nx
import matplotlib.pyplot as plt
import torch
import numpy as np
n = 1000  # Number of random triples you want to select
indices = torch.randperm(dataset.triples.size(0))[:n]

# Sample random triples from the tensor using these indices
random_triples = dataset.triples[211000:211100]
# Randomly select n triples from dataset

def shorten(name, length=70):
    """Ensure name is a string and shorten it intelligently."""
    if isinstance(name, tuple):

        url=name[0]
        for base_url, short_label in head_dict.items():
          if url.startswith(base_url):  # Check if URL starts with a known base URL
              return short_label  # Return the corresponding short label



        # If tuple contains 'POLYGON', extract and shorten the polygon description
        for part in name:
            if "POLYGON" in part:
                return "POLYGON(…)"
            if "POINT" in part:
              return "Point()" # Shortened for readability
            if 'http://kgbench.info/dt#base64Image' in  part:
              return "image_base_64"
            if 'wiki' in  part:
```

```
                .
            return "image_base_64"

        # Default: Convert entire tuple to string if no specific pattern found
        name = name[0]
    if "http" in name :
      print(name)
    # Truncate if necessary
    return name[:length] + "…" if len(name) > length else name


# Create a MultiDiGraph where edges have labels (relations)
G = nx.MultiDiGraph()
for head, rel, tail in random_triples:
       # G.add_edge(dataset.i2e[head], dataset.i2e[tail], label=dataset.i2r[rel])
        G.add_edge(int(head),int(tail), label=dataset.i2r[rel])
# Set different colors for Head and Tail nodes
head_nodes = {int(head) for head, _, _ in random_triples}
tail_nodes = {int(tail) for _, _, tail in random_triples}

# Assign colors
node_colors = []
for node in G.nodes():
    if int(node)==211084:#to fidn degree for some reason its shown seperately
        print("Found")
        node_colors.append("yellow")
    elif  int(node) in head_nodes:
        node_colors.append("skyblue")  # Head nodes - Blue
    elif int(node) in tail_nodes:
        node_colors.append("red")  # Tail nodes - Red
    else:
        node_colors.append("red")  # Other nodes (if any)

# Draw a small subgraph for visualization
plt.figure(figsize=(20,15))
pos = nx.spring_layout(G, k=3/np.sqrt(n), scale=20)  # Increased k and scale for better spacing
nx.draw(G, pos, with_labels=False, node_size=50, font_size=8, node_color=node_colors)
edge_labels = {(u, v): relation_dict.get(d['label'],"unkown") for u, v, d in G.edges(data=True)}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=7)
node_labels = {node: shorten(dataset.i2e[node]) for node in G.nodes()}
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=10)
plt.show()
```

Found




```
# Print nodes with degree > 1 along with their mapped value from dataset.i2e
print("Nodes with Degree > 1:")

for node in G.nodes():
    degree = G.degree(node)
    if degree > 1:
        node_value = dataset.i2e[node ] # Handle missing mappings
        print(f"Node: {node}, Degree: {degree}, Mapped Value: {node_value}")
```

```
Nodes with Degree > 1:
Node: 341077, Degree: 10, Mapped Value: ('zwart wit negatief', '@nl-nl')
Node: 228653, Degree: 11, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215140', 'iri')
Node: 4328, Degree: 2, Mapped Value: ('1980-04', 'none')
Node: 79498, Degree: 12, Mapped Value: ('Wal, A.J. van der', 'none')
Node: 23569, Degree: 9, Mapped Value: ('Collectie gebouwd', '@nl-nl')
Node: 211084, Degree: 9, Mapped Value: ('https://data.labs.pdok.nl/rce/def/Afbeelding', 'iri')
Node: 283366, Degree: 2, Mapped Value: ('https://data.labs.pdok.nl/rce/id/monument/39032', 'iri')
Node: 201839, Degree: 5, Mapped Value: ('http://www.rnaproject.org/data/0d4bd09f-e7e6-44b1-8939-e6f65d3cf47a', 'http://www.w3.org/200
Node: 228654, Degree: 11, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215142', 'iri')
Node: 228656, Degree: 11, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215428', 'iri')
Node: 25943, Degree: 2, Mapped Value: ('Dukker, G.J.', 'none')
Node: 286852, Degree: 3, Mapped Value: ('https://data.labs.pdok.nl/rce/id/monument/526018', 'iri')
Node: 228657, Degree: 11, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215429', 'iri')
Node: 4160, Degree: 2, Mapped Value: ('1968-09', 'none')
Node: 228658, Degree: 11, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215430', 'iri')
Node: 228659, Degree: 10, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215498', 'iri')
Node: 4167, Degree: 2, Mapped Value: ('1969-03', 'none')
Node: 287106, Degree: 4, Mapped Value: ('https://data.labs.pdok.nl/rce/id/monument/529104', 'iri')
Node: 228660, Degree: 10, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215499', 'iri')
Node: 336504, Degree: 2, Mapped Value: ('overzicht', '@nl-nl')
Node: 228661, Degree: 10, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215507', 'iri')
Node: 4284, Degree: 3, Mapped Value: ('1977-02', 'none')
Node: 29570, Degree: 5, Mapped Value: ('Galen, Paul van', 'none')
Node: 228662, Degree: 10, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215508', 'iri')
Node: 228663, Degree: 4, Mapped Value: ('https://data.labs.pdok.nl/rce/id/image/20215509', 'iri')
```

```
for head, _, tail in random_triples:
    if int(head.item) == 211084 or int(tail) == 211084:
```

```
        print(f" head {head.item()} {(type(head))} and tail  {tail} ")
```

```
   head 228653 <class 'torch.Tensor'> and tail  211084
   head 228654 <class 'torch.Tensor'> and tail  211084
   head 228656 <class 'torch.Tensor'> and tail  211084
   head 228657 <class 'torch.Tensor'> and tail  211084
   head 228658 <class 'torch.Tensor'> and tail  211084
   head 228659 <class 'torch.Tensor'> and tail  211084
   head 228660 <class 'torch.Tensor'> and tail  211084
   head 228661 <class 'torch.Tensor'> and tail  211084
   head 228662 <class 'torch.Tensor'> and tail  211084
```

```
node_degrees = dict(G.degree())  # Total degree (in-degree + out-degree)
filtered_nodes = {node for node, degree in node_degrees.items() if degree >= 2}  # Keep nodes with degree ≥ 2

# Create a subgraph with only the filtered nodes
G_filtered = G.subgraph(filtered_nodes).copy()

# Assign colors
node_colors = ["skyblue" if node in filtered_nodes else "red" for node in G_filtered.nodes()]

# Draw the filtered graph
plt.figure(figsize=(20, 15))
pos = nx.spring_layout(G_filtered, k=3/np.sqrt(n), scale=20)
nx.draw(G_filtered, pos, with_labels=False, node_size=50, font_size=8, node_color=node_colors)
edge_labels = {(u, v): relation_dict.get(d['label'], "unknown") for u, v, d in G_filtered.edges(data=True)}
nx.draw_networkx_edge_labels(G_filtered, pos, edge_labels=edge_labels, font_size=7)
node_labels = {node: shorten(dataset.i2e[node]) for node in G_filtered.nodes()}
nx.draw_networkx_labels(G_filtered, pos, labels=node_labels, font_size=10)
plt.show()

print(f"Original graph nodes: {G.number_of_nodes()}, edges: {G.number_of_edges()}")
print(f"Filtered graph nodes: {G_filtered.number_of_nodes()}, edges: {G_filtered.number_of_edges()}")
```

```
import random

random_triples = random.sample(dataset.triples.tolist(), 70)  # Select 120 random triples
```

```
urls=[]
for head, rel, tail in random_triples:
    head_value = dataset.i2e[head]  # Get the actual head value
    tail_value = dataset.i2e[tail]
    urls.append(head_value)
    if "@nl-nl" == tail_value[1] or "none" == tail_value[1] :
      print(tail_value)
```

```
('Nieuwstraat', 'none')
('Achtkante stellingmolen in de winter schuin van opzij gefotografeerd achter loods', '@nl-nl')
('48', 'none')
('Dukker, G.J.', 'none')
('POLYGON ((156818.473816 462205.330627, 156820.317017 462208.863037, 156828.575989 462224.725037, 156829.343018 462226.197998, 15683
('2002-12', 'none')
('Exterieur overzicht', '@nl-nl')
('gevel boenhok/bakhuis repro fotonr. 3060, J.Verheul', '@nl-nl')
('Wal, A.J. van der', 'none')
('Agrarisch Erfgoed - Fotocollectie SHBO', '@nl-nl')
('zwart wit negatief', '@nl-nl')
('Zuid Haffel', 'none')
('zwart wit negatief', '@nl-nl')
('Inleiding Eclectisch LANDHUIS genaamd Petite Suisse annex Villa Kanjel annex Dr.Poelsoord, 1880, gesitueerd in een historische park
('2011', 'none')
('Evers, F.P.C.', 'none')
('Wal, A.J. van der', 'none')
('6', 'none')
('1987-12', 'none')
('1996-05', 'none')
('2007', 'none')
('Dorpsweg 82 te Twisk', 'none')
("Oude Foto's", '@nl-nl')
('Agrarisch Erfgoed - Fotocollectie SHBO', '@nl-nl')
('Opfergeltstraat 3 te Wijnandsrade, e.a.', 'none')
('1965', 'none')
('Gevelsteen', '@nl-nl')
('Papenvoort 2A te Nuenen', 'none')
('Overzicht van de rechter zijgevel en de achtergevel', '@nl-nl')
('Collectie gebouwd', '@nl-nl')
```

```
from urllib.parse import urlparse
filtered_urls = []
uniqueurl=set()
for head,_, _ in dataset.triples:
    url=dataset.i2e[head][0]
    parsed_url = urlparse(url)
    base_path = "/".join(parsed_url.path.strip('/').split('/')[:-1])  # Remove last segment
    new_url = f"{parsed_url.scheme}://{parsed_url.netloc}/{base_path}/"  # Reconstruct the URL
    uniqueurl.add(new_url)
    filtered_urls.append(new_url)
print(uniqueurl,len(uniqueurl))
# Print results
for url in filtered_urls:
    print(len(url))
```

Start coding or generate with AI.

```
{'http://sws.geonames.org/2744199/', 'http://sws.geonames.org/2756359/', 'http://sws.geonames.org/2755243/', 'http://sws.geonames.org
```

finding unqire heads

```
import random
from urllib.parse import urlparse
from tqdm import tqdm

unique_urls = set()

for head, rel, tail in tqdm(dataset.triples, desc="Processing triples"):
        head_value = dataset.i2e[head]  # Get the actual head value
```

```
        if not isinstance(head_value, (list, tuple)):  # Ensure it's iterable
            print(f"Unexpected head_value: {head_value}")
            continue

        parsed_url = urlparse(head_value[0])
        base_path = "/".join(parsed_url.path.strip('/').split('/')[:-1])  # Remove last segment
        new_url = f"{parsed_url.scheme}://{parsed_url.netloc}/{base_path}/"  # Reconstruct the URL
        unique_urls.add(new_url)




# Save unique URLs to a text file
output_file = "unique_urls.txt"
with open(output_file, "w") as f:
    for url in unique_urls:
        f.write(url + "\n")

print(f"Total unique base URLs: {len(unique_urls)}")
```

```
Processing triples: 100%|██████████| 777124/777124 [00:14<00:00, 54190.27it/s]Total unique base URLs: 3200
```

```
unique_urls2 = {url for url in unique_urls if ".geonames.org" not in url and "wiki" not in url }
unique_urls2.add("http://sws.geonames.org/")
unique_urls2.add("http://www.geonames.org/")
unique_urls2.add("http://wikipedia.org/")

# Print the cleaned set
print(len(unique_urls2),(unique_urls2))
```

```
18 {'https://data.labs.pdok.nl/.well-known/genid/', 'https://data.pldn.nl/cbs/wijken-buurten/regios/2016/id/land-geografisch/', 'http
```

```
import requests
from rdflib import Graph

geonames_id = "2755348"
rdf_url = f"http://sws.geonames.org/{geonames_id}/about.rdf"

g = Graph()
g.parse(rdf_url)

for s, p, o in g:
    print(s, p, o)
```

```
https://sws.geonames.org/2755348/ http://www.w3.org/2003/01/geo/wgs84_pos#long 6.12083
https://sws.geonames.org/2755348/about.rdf http://creativecommons.org/ns#attributionName GeoNames
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#parentADM2 https://sws.geonames.org/6544256/
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#nearbyFeatures https://sws.geonames.org/2755348/nearby.rdf
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#locationMap https://www.geonames.org/2755348/greffelkamp.html
https://sws.geonames.org/2755348/about.rdf http://purl.org/dc/terms/modified 2007-06-03
https://sws.geonames.org/2755348/ http://www.w3.org/2003/01/geo/wgs84_pos#lat 51.9525
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#countryCode NL
https://sws.geonames.org/2755348/about.rdf http://creativecommons.org/ns#license https://creativecommons.org/licenses/by/4.0/
https://sws.geonames.org/2755348/about.rdf http://www.w3.org/1999/02/22-rdf-syntax-ns#type http://xmlns.com/foaf/0.1/Document
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#featureClass https://www.geonames.org/ontology#P
https://sws.geonames.org/2755348/about.rdf http://creativecommons.org/ns#attributionURL https://www.geonames.org
https://sws.geonames.org/2755348/about.rdf http://xmlns.com/foaf/0.1/primaryTopic https://sws.geonames.org/2755348/
https://sws.geonames.org/2755348/ http://www.w3.org/1999/02/22-rdf-syntax-ns#type http://www.geonames.org/ontology#Feature
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#parentCountry https://sws.geonames.org/2750405/
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#featureCode https://www.geonames.org/ontology#P.PPL
https://sws.geonames.org/2755348/about.rdf http://purl.org/dc/terms/created 2006-01-15
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#parentFeature https://sws.geonames.org/6544256/
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#name Greffelkamp
https://sws.geonames.org/2755348/ http://www.geonames.org/ontology#parentADM1 https://sws.geonames.org/2755634/
https://sws.geonames.org/2755348/ http://www.w3.org/2000/01/rdf-schema#isDefinedBy https://sws.geonames.org/2755348/about.rdf
```

unique tails

```
import random
from urllib.parse import urlparse
from tqdm import tqdm

tails=[]
for head, rel, tail in tqdm(dataset.triples, desc="Processing triples"):
        tail_value = dataset.i2e[tail]  # Get the actual head value
        tails.append(tail_value)




# Save unique URLs to a text file
output_file = "tails .txt"
with open(output_file, "w") as f:
    for tail in tails:
        f.write(tail[0] + "\n")

print(f"Total tails: {len(tails)}")
```

```
# Save unique URLs to a text file
output_file = "tails .txt"
with open(output_file, "w") as f:
    for tail in tails:
        f.write(tail[0] + "\n")

print(f"Total tails: {len(tails)}")
```

```
Total tails: 777124
```

```
torch.version.cuda
```

```
'12.4'
```

```
!pip install -q torch-geometric
!pip install open_clip_torch
!pip install shapely # For handling geometry
!pip install PILLOW
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.1/63.1 kB 3.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 33.8 MB/s eta 0:00:00
  Collecting open_clip_torch
    Downloading open_clip_torch-2.31.0-py3-none-any.whl.metadata (31 kB)
  Requirement already satisfied: torch>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (2.6.0+cu124)
  Requirement already satisfied: torchvision in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (0.21.0+cu124)
  Requirement already satisfied: regex in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (2024.11.6)
  Collecting ftfy (from open_clip_torch)
    Downloading ftfy-6.3.1-py3-none-any.whl.metadata (7.3 kB)
  Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (4.67.1)
  Requirement already satisfied: huggingface-hub in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (0.29.3)
  Requirement already satisfied: safetensors in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (0.5.3)
  Requirement already satisfied: timm in /usr/local/lib/python3.11/dist-packages (from open_clip_torch) (1.0.15)
  Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (3.18.0)
  Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_tor
  Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (3.4.2)
  Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (3.1.6)
  Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (2025.3.0)
  Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_c
  Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open
  Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_c
  Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_t
  Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_
  Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_t
  Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_cli
```

```
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_cli
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_c
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torc
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_to
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_cl
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.9.0->open_clip_torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.9.0->open_
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from ftfy->open_clip_torch) (0.2.13)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub->open_clip_torch) (24
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub->open_clip_torch) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub->open_clip_torch) (2.32.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from torchvision->open_clip_torch) (2.0.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from torchvision->open_clip_torch) (
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.9.0->open_clip_torch
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub->o
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub->open_clip_tor
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub->open_cl
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub->open_cl
Downloading open_clip_torch-2.31.0-py3-none-any.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 28.3 MB/s eta 0:00:00
Downloading ftfy-6.3.1-py3-none-any.whl (44 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.8/44.8 kB 1.7 MB/s eta 0:00:00
Installing collected packages: ftfy, open_clip_torch
Successfully installed ftfy-6.3.1 open_clip_torch-2.31.0
Requirement already satisfied: shapely in /usr/local/lib/python3.11/dist-packages (2.0.7)
Requirement already satisfied: numpy<3,>=1.14 in /usr/local/lib/python3.11/dist-packages (from shapely) (2.0.2)
Requirement already satisfied: PILLOW in /usr/local/lib/python3.11/dist-packages (11.1.0)
```

```
import torch
import open_clip
from shapely.wkt import loads as load_wkt
from PIL import Image
import io
import base64
import numpy as np

# Load CLIP model
model ,preprocess_train,preprocess_val = open_clip.create_model_and_transforms("ViT-B-32", pretrained="openai")
tokenizer = open_clip.get_tokenizer("ViT-B-32")

device = "cuda" if torch.cuda.is_available() else "cpu"
model=model.to(device)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
open_clip_model.safetensors: 100%  605M/605M [00:03<00:00, 178MB/s]
/usr/local/lib/python3.11/dist-packages/open_clip/factory.py:388: UserWarning: These pretrained weights were trained with QuickGELU a
  warnings.warn(
```

```
 correct  for new vesio nimport torch
import open_clip
from shapely.wkt import loads as load_wkt
from PIL import Image
import io
import base64
import numpy as np

# Load CLIP model
model, preprocess = open_clip.create_model_and_transforms("ViT-B-32", pretrained="openai"
tokenizer = open_clip.get_tokenizer("ViT-B-32")

device = "cuda" if torch.cuda.is_available() else "cpu"
```

```
  File "<ipython-input-5-20dcd323879b>", line 1
    correct  for new vesio nimport torch
             ^
SyntaxError: invalid syntax
```

```
import torch
import open_clip
import base64  # Import base64 for decoding
from shapely.wkt import loads as load_wkt
from PIL import Image
import io
import numpy as np
import os
import psutil
import gc

def get_clip_embedding(triple, dataset, model, tokenizer, preprocess, device):
    head, relation, tail = triple  # Extract entity, relation, and value

    # Convert head and tail using i2e (entity list)
    head_text = dataset.i2e[head][0]
    relation_text = dataset.i2r[relation][0]
    tail_text = dataset.i2e[tail][0]

    # Handle different types of values
    if isinstance(tail_text, str):
        # Check if it's a URL, spatial data (WKT), or plain text
        if tail_text.startswith("http") or tail_text.startswith("www"):
            tail_text = f"URL: {tail_text}"
        elif tail_text.startswith("POLYGON") or tail_text.startswith("POINT"):
            geom = load_wkt(tail_text)  # Convert WKT to geometry
            tail_text = f"Geometry: {geom.wkt}"  # Convert back to text
        elif tail_text.startswith("jqwbwihbafohusbfnq"):  # Image handling
            print(len(tail_text))  # Debugging output

            try:
                if isinstance(tail_text, str):
                    # Check if it's a Base64-encoded string and decode it
                    image_bytes = base64.b64decode(tail_text)
```