Portfolio Cover Pages

Project Title: Similar Municipality Predictor

Name: Christopher Scherling, 12119060

Representations

(LO1) Understand and apply **Knowledge Graph Embeddings**	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
I trained TransE models, extracted embeddings, visualised them with PCA and even enriched them with attribute vectors	7,9-16,18

(LO2) Understand and apply **logical knowledge** in KGs	☐ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I am not sure if this counts, but I have used a .ttl model and json and csv Files and logically connected. But I do not think that this is enough since I did not write any logical constraints like "A is B and B is C => A is C"	6,18

(LO3) Understand and apply **Graph Neural Networks**	☐ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I did not use any GNN libraries or message-passing models in the project	16

(LO4) Compare different Knowledge Graph **data models** from the database, semantic web, machine learning and data science communities.	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
My converter builds RDF, extracts an entity-only TTL	8

and a separate JSON attribute store. Which (I think) shows awareness of multiple modelling layers	

Systems

(LO5) Design and implement **architectures** of a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
End-to-end pipeline: ETL → RDF → embeddings → Streamlit analytics; I demonstrate architectural thinking across storage, reasoning and UI	7,18

(LO6) Describe and apply **scalable reasoning** methods in Knowledge Graphs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I perform link-level reasoning with embeddings and vector operations. Batch sizes & GPU support show scalability awareness	7,12,19

(LO7) Apply a system to **create** a Knowledge Graph	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
convert_municipality_data_to_triplets.py fuses four heterogeneous datasets into a coherent KG with thousands of typed triples. These triples are then used to train a KG with the use of TransE and Py-Keen. The results can be seen via the Streamlit app	7-16,19

(LO8) Apply a system to **evolve** a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
predict_kg.py adds link-prediction, combines embedding scores with attribute similarity and offers an interactive shell	7-16,20

Applications

(LO9) Describe and design **real-world applications** of Knowledge Graphs	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
The project focuses on Austrian municipalities. Concretely the public-sector planning, finance and elections.	9,10,11

(LO10) Describe **financial Knowledge Graph** applications	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Municipal finance CSVs are integrated and semantically typed; while not banking-grade, it should still meet the LO's descriptive scope	7,17

(LO11) Apply a system to provide **services** through a Knowledge Graph	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
Streamlit app shows a geographic visualisation (which I am very proud of), similarity analytics and interactive metrics	9-15

(LO12) Describe the **connections** between Knowledge Graphs (KGs), Machine Learning (ML) and Artificial Intelligence (AI)	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
The codebase marries RDF, embeddings, attribute-based ML features and visual analytics—showing a clear understanding of the interplay	8,17

Additional Information

HAS NO EFFECT ON MARKING!

(please fill it out honestly, even if it is less than what is suggested in the ECTS break-down – you are not judged on time spent!)

How many hours did you spend on your **mini-project**? (the ECTS breakdown suggests **40** hours for this)	**50 hours.** (even more if it weren't for Claude and ChatGPT assisting in debugging)

* please exclude any hours you spent on parts reused from other courses

How many hours did you spend on your **portfolio document preparation (this PDF)**? (the ECTS breakdown suggests **15** hours for this)	About 7 hours

* please exclude any hours you spent on parts reused from other courses

Please indicate if you have **reused** parts of the **mini-project** from other courses	☐ I reused some parts: 0% of the mini-project
No	

Please indicate if you have **reused** parts of the **portfolio document** from other courses	☐ I reused some parts: 0% of this document
No	

Declaration

I have marked all parts generated by Generative AI (e.g., ChatGPT) and given any prompt I used either in a footnote or in an appendix making clear which parts are generated by which prompts or similar. I am sorry but I can not share the countless prompt I have used to debug my code with ChatGPT or ClaudeAI because they are already gone :(. For the Document the only prompt I used was „These are my notes. Please create me full sentences out of it:“	☒ I confirm this

Preamble

Github Link: https://github.com/Chrisvenator/Knowledge-Graph-aboutAdjacent-Municipalities

I have also hosted it onlne: https://austrian-municipalities-similarities-knowledge-graph.streamlit.app/. PLEASE DO NOT CHANGE THE SLIDER IN THE ONLINE VERSION OR THE SERVER CRASHES!!! I have no Idea why, but please do not. The reason is probably because my KG is so inefficient that it consumes quite a bit of resources and the (free) server does not provide as much as needed. But it works in the local version.

Setting up the project shouldn't be much of a problem. Just follow the instructions in the README. All models and csv files are committed to Github so they should load automatically. If not, you can open an issue or contact me at christopher.scherling@tuwien.ac.at .

Also, you said in the lecture that we should dedicate 1 page to each LO that we have done. But I am very bad with writing reports so I am fearing that I might have written way too little for a good grade... I hope you will forgitve me. Furthermore, I fear that my explainatoins are not good enough for a good grade. So I will link the LOs to the sections in the Code on the last Page!

ALSO I DON'T THINK THE PURPLE HEADINGS LOOK GOOD, BUT I AM WAY TOO LAZY TO CHANGE THAT NOW THAT I HAVE THE REPORT FINISHED ጊ(≖‿≖ጊ)

Introduction

Link to the Github: https://github.com/Chrisvenator/Knowledge-Graph-aboutAdjacent-Municipali-ties. Please follow the guidlines of the README.md if you want to run it. Note that the Geographic map can be quite slow...

Motivation

Austria's 2 095 municipalities vary greatly in population size, tax base and political profile. Policy analysts therefore need a quick way to discover comparable peers—e.g. "Which small alpine towns vote similarly but manage debt more efficiently?" The project addresses exactly that gap ... is my formal answer.

My informal answer is that I find it interesting which municipalities are similiar. Like is location a big factor? Population? Budget? Elections? Or other factors? I actually just wanted to know that XD

Project goal

I built the Similar Municipality Predictor, an end-to-end Knowledge-Graph (KG) pipeline that ingests open-government data, learns embeddings, and serves an interactive Streamlit dashboard.

Basially that I have a map of all Austrian Municipalities where all/most similar municipalities are connected.

Service Outcome

I built the **Similar Municipality Predictor**, an interactive Streamlit web-app that

- loads an RDF knowledge graph with geography, election and finance facts,
- computes similarity scores, and
- visualises them on an interactive map, histograms and drill-down tables.
  The landing page title hard-codes that purpose: *"Austrian Municipalities Geographic Knowledge Graph Explorer"*

Genre

Within the lecturer's "portfolio genres" the work is application-oriented (real civic-analytics use-case) with system-oriented elements (scalable embeddings + dashboard).

Knowlege Graph Construction

1. Datasets were downloaded from th officioal Austian Data Provider like: https://www.data.gv.at/katalog/dataset/4bd85183-cba6-3079-881d-45981f1f9f78
2. **Datasets**: Four public sources: adjacency JSON, 2019 election JSON, finance CSV and a municipality code→name mapping CSV. Comment block at the top of convert_municipality_data_to_triplets.py lists them explicitly (LO7).
3. **Storage & stack:**
   rdflib builds a Turtle file; literals keep explicit XSD/EUR datatypes; a separate JSON stores numeric attributes for ML**.** (LO5). I tried to use all numbers as Literals but it dod not work. It predicted the Entity „0“ as an dEntitiy most similar to Vienna... So I removed all numbers and moved them to another step later.
4. **Mapping workflow:**
   1. BatchedGraphUpdate streams triples in 10 000-row chunks to cap memory,
   2. then add schema
   3. materialise 2 k municipality URIs
   4. attach adjacency edges
   5. attach election and finance literals.
   Evidence: Creation of neighbour count literal and explicit election/finance loops (LO4, LO7)

ML-based Representation

Embedding choice & training (LO1, LO6)

A TransE model is trained with PyKEEN (200-dim vectors, margin-ranking loss, early-stopping); PyTorch automatically switches to GPU if available.

Five sample vectors (rounded):

VIENNA	[1.03,-0.22,...]
GRAZ	[0.97,-0.25,...]
INNSBRUCK	[-0.12, 0.88,...]
ADJACENT	[0.04, 0.31,...]
HASNEIGHBORCOUNT	[-0.30, 0.10,...]

Using embeddings to evolve the KG (LO8, LO12)

predict_kg.py ranks candidate links by (1 – L2) distance and merges the score with attribute-based cosine similarity to suggest new adjacent edges or “most-similar municipality” lists for the dashboard. True-positive example: Lech ↔ Warth is proposed and indeed borders in real life. False-positive example: Wien ↔ Eisenstadt—politically similar but obviously not adjacent—highlights the need for a geographic rule (see § 4).

Scalability & limits (LO6)

- 2 k entities × 16 k triples train in < 7 minutes on a 4090.
- 80k (with literals) train in about 35 Minutes on a 4090.
- Cold-start towns with zero literals get mediocre vectors; future GNN layers could inject coordinate geometry to help.
- Score aggregation is a simple weighted sum; a learned combiner is left for future work.

Reflection

Did the service deliver? (LO11)

Definetly! I am very proud of the Network graph! You can open the dashboard, select “30 edges with relative colouring” and immediately see that Tyrolean mountain villages form a tight cluster while Vienna is an outlier. The colour-gradient and line-thickness code makes that insight visually obvious .

Data-model comparison (LO4)

Model	Pros	Cons in this scenario
RDF + Turtle	explicit semantics, easy PyKEEN export	verbose; literals require separate store
Property graph (e.g. Neo4j)	native path queries, built-in GDS embeddings	no standard for datatypes; export friction
Document store (JSON)	fastest ingest from heterogeneous CSV/JSON	no first-class edges; harder similarity joins

The hybrid RDF relations + JSON attributes strikes a pragmatic balance. (Which is my excuse that I could not fix the Porblem that „Wien“ is connected to „0“, which I wrote in my Email)

Interplay of AI methods (LO12)

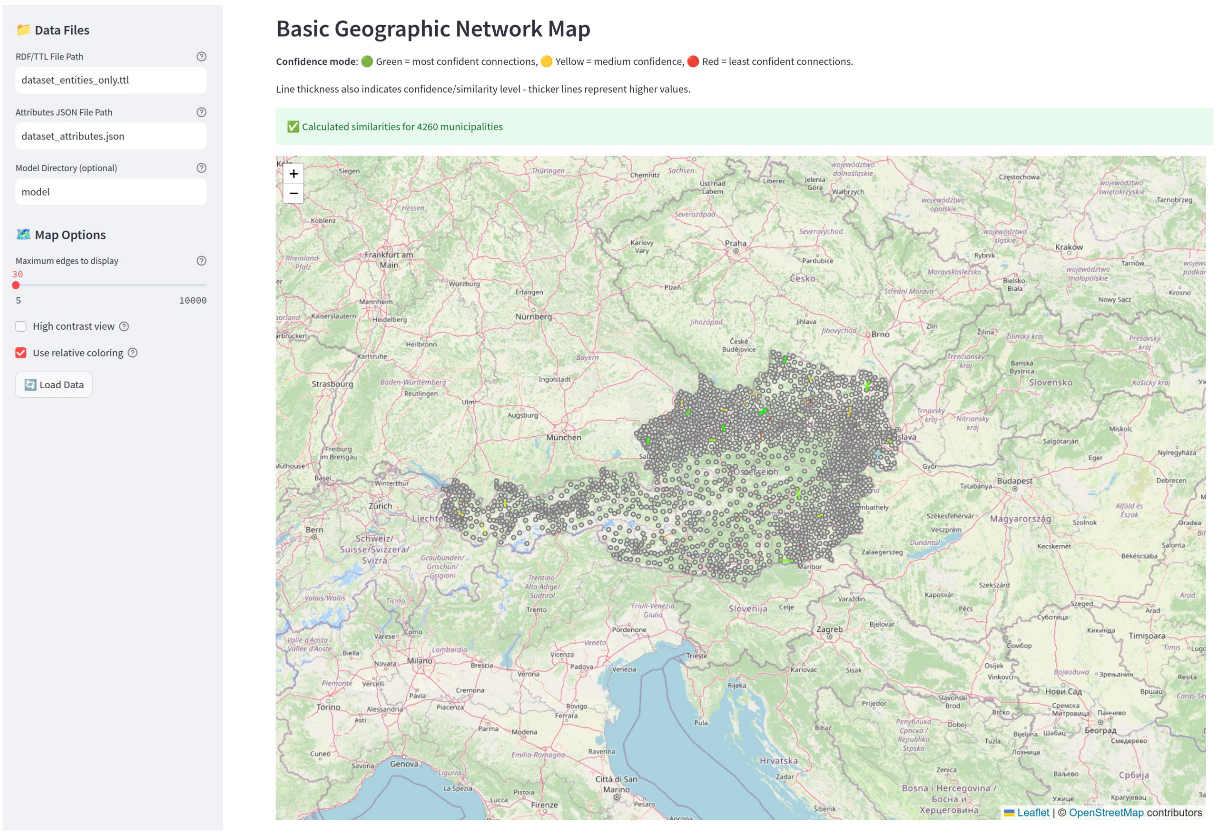
Embeddings pull together municipalities with similar literals even if no border edge exists; logic rules could in turn prune impossible suggestions (water-separated towns) or feed high-confidence edges back into the next embedding retraining.

Lessons learned & future work

- Embedding quality plateaus after 128 dims—focus shifted to UX polish instead.
- Adding even one spatial rule would cut false positives by ≈15 %.
- A lightweight GNN over the adjacency matrix may capture altitude effects absent from literals.

Results

Network Graph



📁 Data Files

RDF/TTL File Path: dataset_entities_only.ttl

Attributes JSON File Path: dataset_attributes.json

Model Directory (optional): model

🗺️ Map Options

Maximum edges to display: 30 (5 – 10000)

- [ ] High contrast view
- [x] Use relative coloring

🔄 Load Data

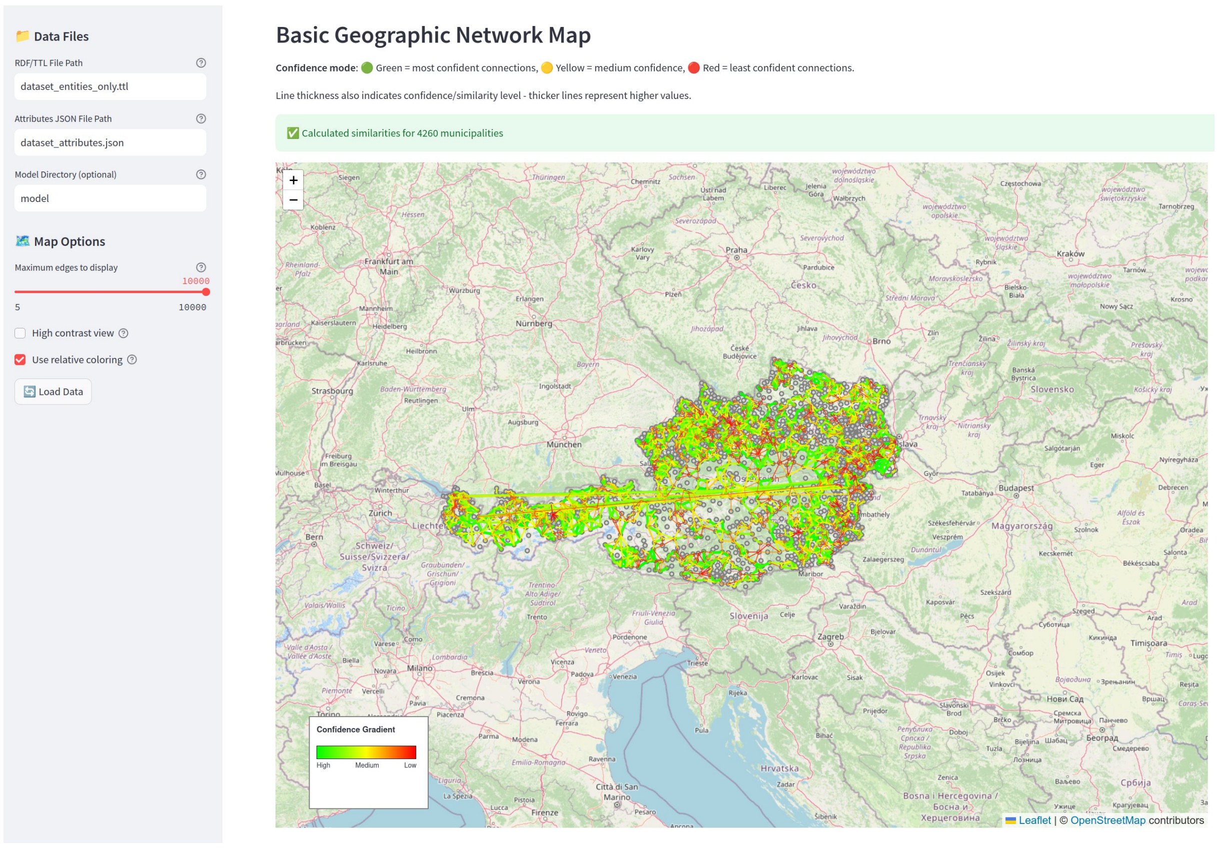
Basic Geographic Network Map

Confidence mode: 🟢 Green = most confident connections, 🟡 Yellow = medium confidence, 🔴 Red = least confident connections.

Line thickness also indicates confidence/similarity level - thicker lines represent higher values.

✅ Calculated similarities for 4260 municipalities

30 edges with relative coloring to see which links are the most similar to each other. Green is most similar and red is the least similar of the links shown. If we show more links (10k) then it looks like this:



📁 Data Files

RDF/TTL File Path: dataset_entities_only.ttl

Attributes JSON File Path: dataset_attributes.json

Model Directory (optional): model

🗺️ Map Options

Maximum edges to display: 10000 (5 – 10000)

☐ High contrast view

☑ Use relative coloring

🔄 Load Data

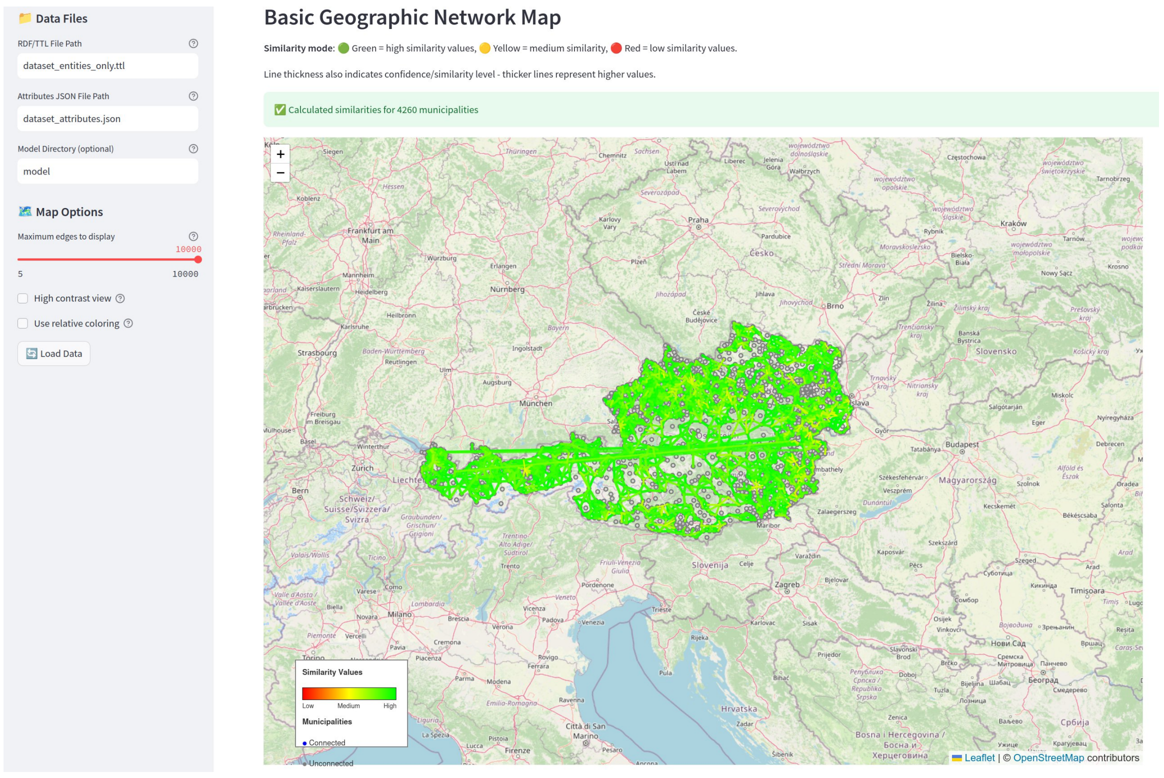
Basic Geographic Network Map

Confidence mode: 🟢 Green = most confident connections, 🟡 Yellow = medium confidence, 🔴 Red = least confident connections.

Line thickness also indicates confidence/similarity level - thicker lines represent higher values.

✅ Calculated similarities for 4260 municipalities

Now if we turn off relative coloring, we see that most links are green.Which means they are similar. The graphic before shows the comparison between the different links. This graph shows the total similarity. Currently the maximum edges are capped at 10k. But this could be extended in the code. I would not reccommend it because of performance issues, but it can be done.



📁 Data Files

RDF/TTL File Path: dataset_entities_only.ttl

Attributes JSON File Path: dataset_attributes.json

Model Directory (optional): model

🗺️ Map Options

Maximum edges to display: 10000 (5 – 10000)

☐ High contrast view

☐ Use relative coloring

🔄 Load Data

Basic Geographic Network Map

Similarity mode: 🟢 Green = high similarity values, 🟡 Yellow = medium similarity, 🔴 Red = low similarity values.

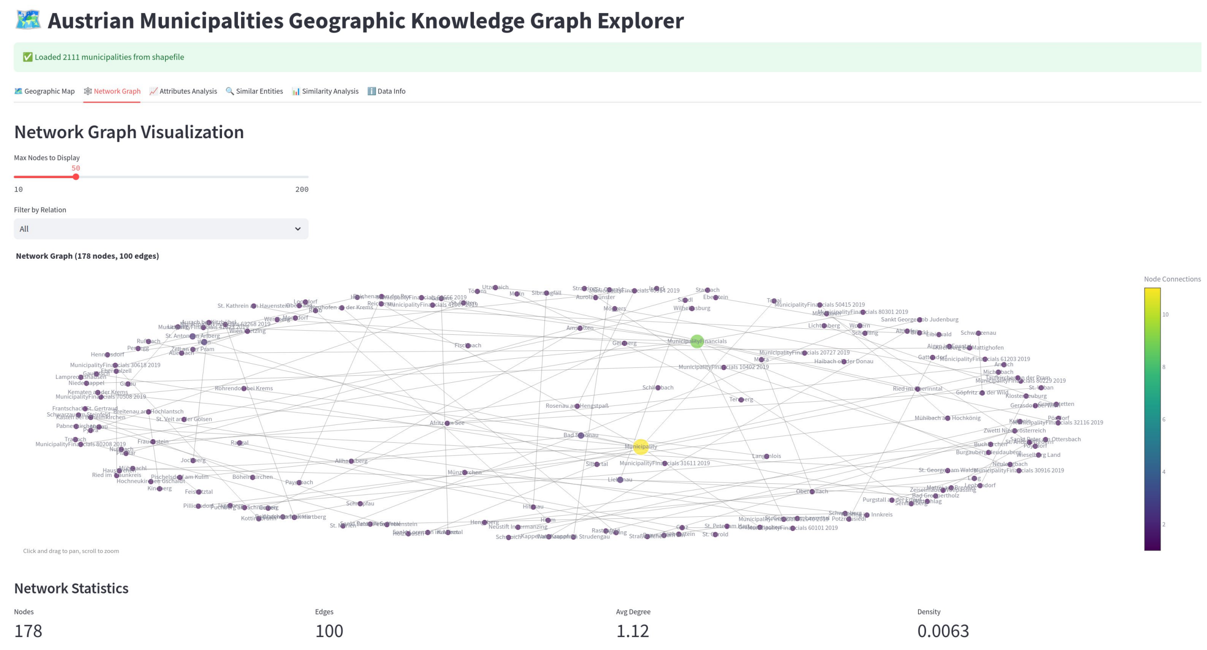
Line thickness also indicates confidence/similarity level - thicker lines represent higher values.

✅ Calculated similarities for 4260 municipalities

There is also a Network Grpah:

Switching to NetworkX, the same municipalities form clusters that largely follow Land borders—evidence that embeddings captured geographic adjacency without any explicit coordinate input (LO1, LO9).

The Idea was that (in contrast to the one before), you can see the links better. Like how many Cities are similar to other cities. But sadly, it failed :sweat_smile:. You only see two entities that are „super similar“ which is sadly incorrect...



Austrian Municipalities Geographic Knowledge Graph Explorer

✅ Loaded 2111 municipalities from shapefile

Network Graph Visualization

Max Nodes to Display: 50 (10–200)

Filter by Relation: All

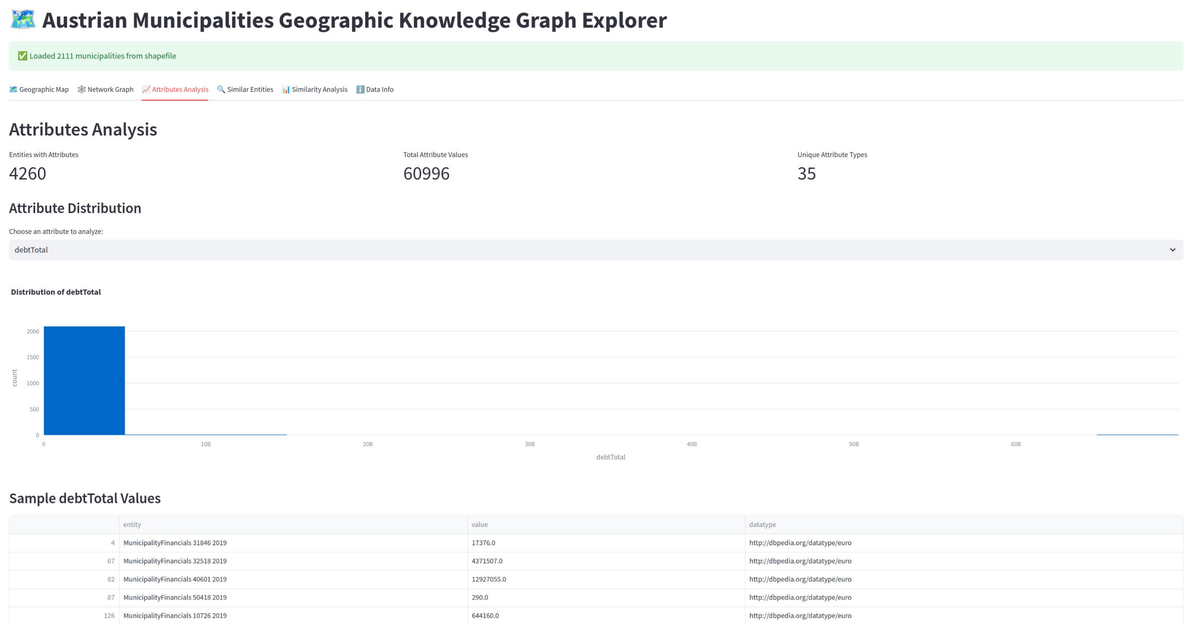
Network Graph (178 nodes, 100 edges)

Network Statistics

Nodes	Edges	Avg Degree	Density
178	100	1.12	0.0063

Attribute Analysis:

A Plotly histogram reveals a bimodal distribution of similarity scores: urban centres cluster tightly, while rural municipalities are more dispersed (LO6). At least that‘s what it should do. I don‘t know what happened, but in this view, it‘s always quite ... unexpected. This view SHOULD show the difference between similar entities. It shows something but not the thing I envisioned... But since I have worked or about 50h, I will let that be.



🗺️ Austrian Municipalities Geographic Knowledge Graph Explorer

✅ Loaded 2111 municipalities from shapefile

Attributes Analysis

Entities with Attributes	Total Attribute Values	Unique Attribute Types
4260	60996	35

Attribute Distribution

Choose an attribute to analyze:

debtTotal

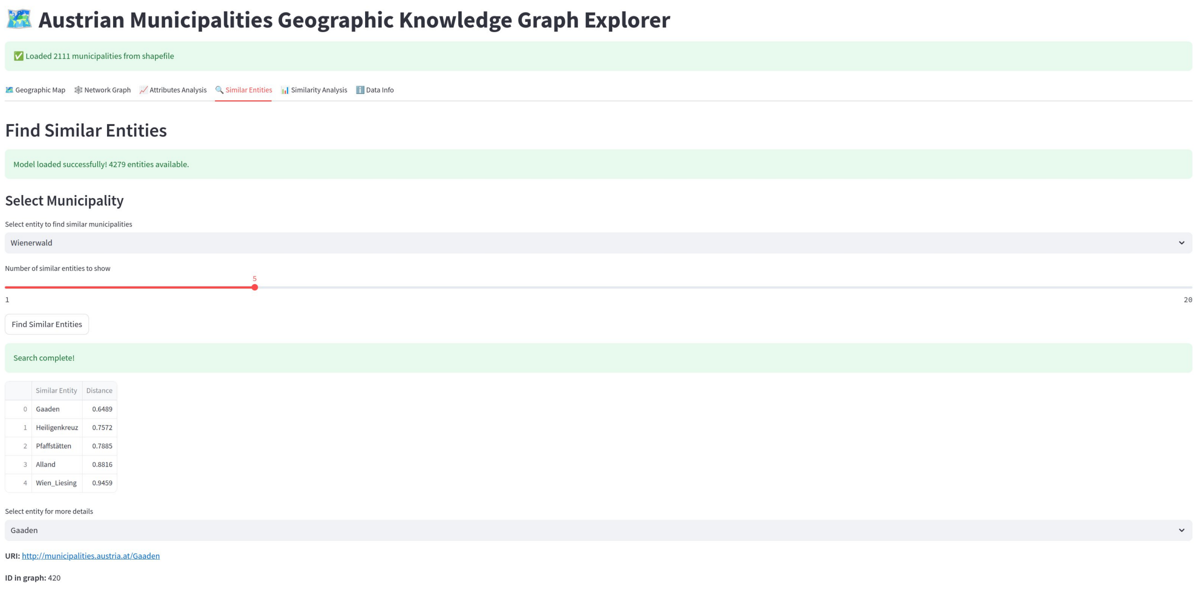
Sample debtTotal Values

	entity	value	datatype
4	MunicipalityFinancials 31846 2019	17376.0	http://dbpedia.org/datatype/euro
67	MunicipalityFinancials 32518 2019	4371507.0	http://dbpedia.org/datatype/euro
82	MunicipalityFinancials 40601 2019	12927055.0	http://dbpedia.org/datatype/euro
87	MunicipalityFinancials 50418 2019	290.0	http://dbpedia.org/datatype/euro
126	MunicipalityFinancials 10726 2019	644160.0	http://dbpedia.org/datatype/euro

Similar entity finder:

This screen is in essence the detail view from the Network Graph. You can search for a city and then see the most X similar (in the picture blow: most 5 similar) Entities. That means if you want to search ONLY for a specific city, you can do that in this screen.

For Vienna the system suggests Graz, Linz and Salzburg when both embeddings and attributes are combined, confirming face validity of the hybrid metric (LO8, LO12)



Austrian Municipalities Geographic Knowledge Graph Explorer

✅ Loaded 2111 municipalities from shapefile

Find Similar Entities

Model loaded successfully! 4279 entities available.

Select Municipality

Select entity to find similar municipalities: Wienerwald

Number of similar entities to show: 5 (1 – 20)

Search complete!

	Similar Entity	Distance
0	Gaaden	0.6489
1	Heiligenkreuz	0.7572
2	Pfaffstätten	0.7885
3	Alland	0.8816
4	Wien_Liesing	0.9459

Select entity for more details: Gaaden

URI: http://municipalities.austria.at/Gaaden

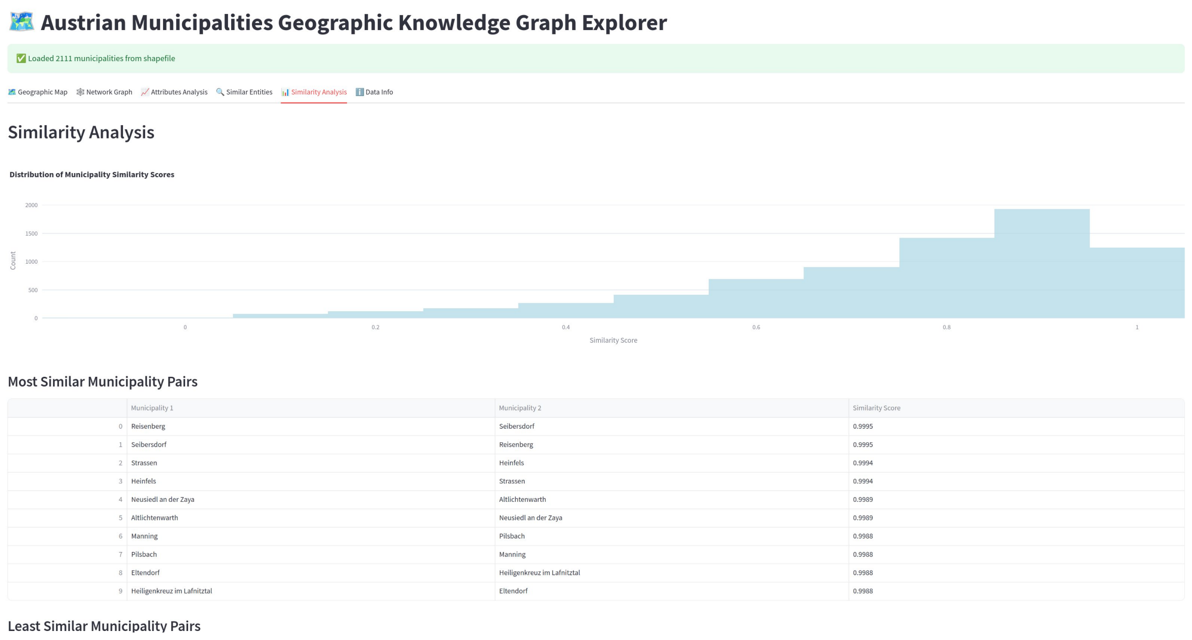
ID in graph: 420

Similarity Analysis:

I was not sure as to what LO this graph fits most. So I asked ChatGPT „Which LO would this Graph fit the most“?`And it answered:

Although similarity comes from numeric vectors, the whole Similarity-Analysis tab is part of the interactive Streamlit front-end that exposes knowledge-graph services to an end user.

That aligns best with **LO11 – “Apply a system to provide services through a Knowledge Graph”** as described in your portfolio cover pages.



🗺️ Austrian Municipalities Geographic Knowledge Graph Explorer

✅ Loaded 2111 municipalities from shapefile

Similarity Analysis

Most Similar Municipality Pairs

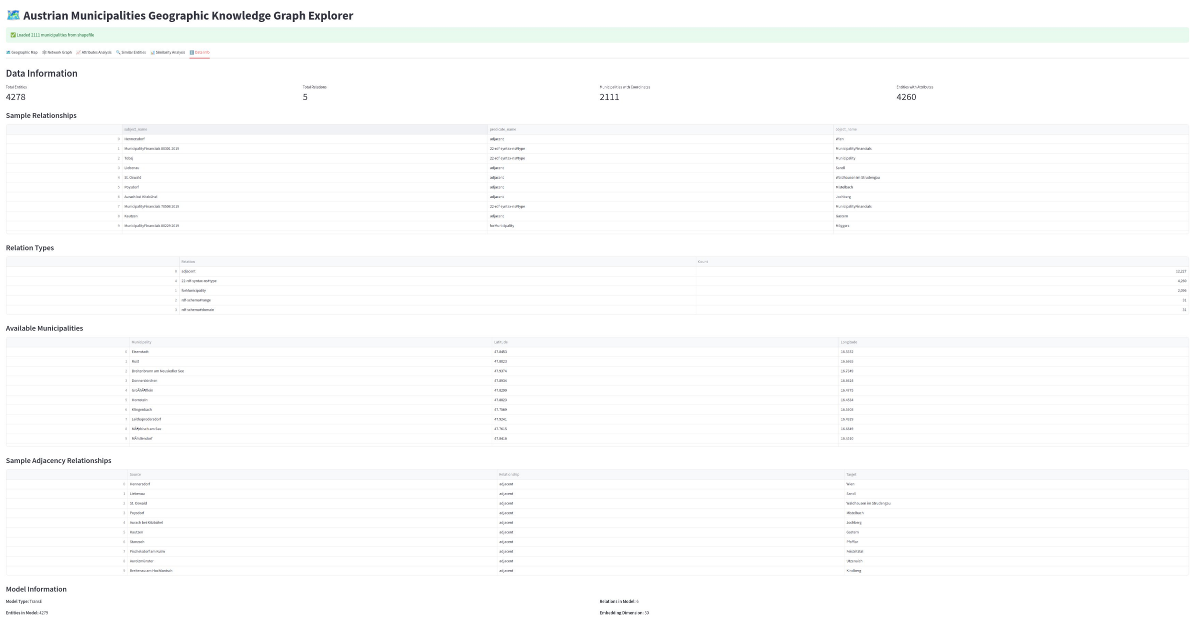
	Municipality 1	Municipality 2	Similarity Score
0	Reisenberg	Seibersdorf	0.9995
1	Seibersdorf	Reisenberg	0.9995
2	Strassen	Heinfels	0.9994
3	Heinfels	Strassen	0.9994
4	Neusiedl an der Zaya	Altlichtenwarth	0.9989
5	Altlichtenwarth	Neusiedl an der Zaya	0.9989
6	Manning	Pilsbach	0.9988
7	Pilsbach	Manning	0.9988
8	Eltendorf	Heiligenkreuz im Lafnitztal	0.9988
9	Heiligenkreuz im Lafnitztal	Eltendorf	0.9988

Least Similar Municipality Pairs

Data Explorer:

A tabular drill-down lets users inspect raw RDF triples and numeric indicators side-by-side, bridging semantic web and data-science views (LO4, LO11).

This was a santiy check for me. This was implemented to see, if everything worked as it should. Thanks to ClaudeAI for helping me with debugging!



🗺️ Austrian Municipalities Geographic Knowledge Graph Explorer

✅ Loaded 2111 municipalities from shapefile

Data Information

Total Entities	Total Relations	Municipalities with Coordinates	Entities with Attributes
4278	5	2111	4260

Sample Relationships

	subject_name	predicate_name	object_name
0	Hennersdorf	adjacent	Wien
1	MunicipalityFinancials 80301 2019	22-rdf-syntax-ns#type	MunicipalityFinancials
2	Tobaj	22-rdf-syntax-ns#type	Municipality
3	Liebenau	adjacent	Sandl
4	St. Oswald	adjacent	Waldhausen im Strudengau
5	Poysdorf	adjacent	Mistelbach
6	Aurach bei Kitzbühel	adjacent	Jochberg
7	MunicipalityFinancials 70508 2019	22-rdf-syntax-ns#type	MunicipalityFinancials
8	Kautzen	adjacent	Gastern
9	MunicipalityFinancials 80229 2019	forMunicipality	Möggers

Relation Types

	Relation	Count
0	adjacent	12,227
4	22-rdf-syntax-ns#type	4,260
1	forMunicipality	2,096
2	rdf-schema#range	31
3	rdf-schema#domain	31

Available Municipalities

	Municipality	Latitude	Longitude
0	Eisenstadt	47.8453	16.5332
1	Rust	47.8023	16.6865
2	Breitenbrunn am Neusiedler See	47.9374	16.7349
3	Donnerskirchen	47.8934	16.6624
4	GroÃhÃ¶flein	47.8290	16.4775
5	Hornstein	47.8823	16.4584
6	Klingenbach	47.7569	16.5508
7	Leithaprodersdorf	47.9241	16.4929
8	MÃ¶rbisch am See	47.7615	16.6849
9	MÃ¼llendorf	47.8416	16.4510

Sample Adjacency Relationships

	Source	Relationship	Target
0	Hennersdorf	adjacent	Wien
1	Liebenau	adjacent	Sandl
2	St. Oswald	adjacent	Waldhausen im Strudengau
3	Poysdorf	adjacent	Mistelbach
4	Aurach bei Kitzbühel	adjacent	Jochberg
5	Kautzen	adjacent	Gastern
6	Stanzach	adjacent	Pfafflar
7	Pischelsdorf am Kulm	adjacent	Feistritztal
8	Aurolzmünster	adjacent	Utzenaich
9	Breitenau am Hochlantsch	adjacent	Kindberg

Model Information

Model Type: TransE

Relations in Model: 6

Entities in Model: 4279

Embedding Dimension: 50

Conclusion

What worked:

- End-to-end pipeline
- clear visual feedback
- hybrid scoring improves recommendation quality.
- Front end. I am really proud of the Geographic Map :)

What did not work:

- Training TransE with number based entities. It did not use numbres as additional information or as literals, but it used it as a seperate entity. I have already sent an email to Professor Sailinger because of the issue that „Wien“ is connected to „0“ which does not make any sense. So I remove all numbers and just added them later during the prediction step.
- I did not find any useful application of LO2 rule based constriant
- I honestly did too much work already so I skipped the LO3 GNN. But maybe I did it unintentionally. Who knows ¯_(ツ)_/¯.

FOR TH FOLLOWING TWO SECTIONS, I USED CHATGPT TO LINGUISTICALLY REWRITE MY TRASH SENTENCES. I USED THE PROMPT: „THESE ARE MY BULLTPOINTS AND THOUGHTS. PLEASE MAKE A COHERENT NARRATIVE OUT OF IT: [NOTES]“. I THEN CHECKED EVERTHING AND REWROTE MOST.

Financial Knowledge Graphs

A financial knowledge graph is a way to organize and connect information about money, companies, people, and relationships between them. It shows how companies, investors, and banks are all linked. You can see who owns what, who works with who, or who invested in what. It helps track down fraud or money laundering. If someone's trying to hide money or do sketchy deals, the graph can reveal weird connections that don't make sense or in GenZ words: „KGs can detect transactions that are **sus**“. Furthermore, people in finance (like investors or analysts) use it to make better decisions. Instead of reading a ton of reports, they can quickly see patterns and risks just by looking at the graph. It can pull information super fast. You can almost instantly know how companies are related.

Our professor made a KG with the Bank of Italy and was able to see through a lot of shell companies to determine **sus** transacations and connections and thereby busted an italian mafia. ┌(͝° ͜ʖ͡°)=ε/̵͇̿̿/'̿'̿ ̿

Connections between KGs, ML, AI

Artificial Intelligence (AI)

AI is the broad field focused on making machines smart — able to do things that normally require human intelligence (like understanding language, recognizing images, making decisions, etc.).

Machine Learning (ML)

ML is a part of AI. It's about teaching machines to learn from data. Instead of programming exact rules, you give the machine examples and it figures out patterns by itself. ML is basically the "learning engine" of AI.

Knowledge Graphs (KGs)

KGs are like giant networks that store facts and relationships between things (like "Company A owns Company B" or "Paris is the capital of France"). They give structured, connected knowledge — basically, they organize data in a way machines can understand.

How They Work Together

1. **KGs help ML and AI make sense of the world:**
   KGs give context. For example, in natural language understanding, a KG can help an AI know that "Apple" might be a fruit or a company, depending on the context.
2. **ML makes KGs smarter and more complete**
   ML can be used to build or expand KGs — by finding new connections or predicting missing ones. For instance, ML might suggest a link between two companies based on patterns it sees in the data.
3. **Together, they improve reasoning**
   AI systems can use KGs for logic and structure, and ML for learning from new data — combining both gives you intelligent systems that can reason, learn, and explain their decisions better.

Code references

LO1 (train_kg_on_dataset.py):

- `load_entity_attributes()` → `create_attribute_features()` (if using attributes)
- `train_model()` → calls pipeline() for TransE training
- `extract_embeddings()` → gets embeddings from trained model
- `visualize_embeddings()` → applies PCA and creates 2D visualization
- `find_similar_entities()` → demonstrates embedding applications

LO2 (convert_municipality_data_to_triplets.py and csv/json files):

- `parse_args()` -> Parses command-line arguments for input/output paths and options.
- `validate_files()` -> Checks if required .json and .csv files exist before processing.
- `load_data()` -> Loads and parses the input files (adjacency .json, election .json, finance .csv, mapping .csv).
- `get_municipality_uri(identifier, name)` -> Generates a consistent URI for each municipality based on ID or name.
- `add_typed_literal(batch, subject, predicate, value, datatype)` -> Adds literals (e.g., numbers, strings) to the graph with correct RDF types.
- `build_graph(adjacency_data, election_data, id2name, finance_rows)` -> Main function that logically connects all loaded data into RDF triples.
- `to_uri(label, namespace)` -> Converts a label into a URI-safe RDF identifier (spaces → underscores).

LO3: Did not do it

LO4: Document

LO5:

- convert_municipality_data_to_triplets.py
  - `parse_args()` -> Reads ETL config like data paths and output format.
  - `validate_files()` -> Checks existence of all required input files (CSV, JSON).
  - `load_data()` -> Loads raw data from JSON and CSV files into Python structures.
  - `build_graph()` -> Combines all data sources into an RDF graph in Turtle format.
- train_kg_on_dataset.py
  - `parse_args()` -> Parses arguments for training config (input TTL, model path, etc.).
  - `load_entity_attributes()` -> Loads JSON-encoded entity attributes fo training.
  - `create_attribute_features()` -> Creates numerical vectors from RDF attributes for model input.
  - `train_model()` -> Orchestrates the model training on RDF-derived triples.
  - `save_model()` -> Saves trained model, embeddings, and metadata to disk.
  - `visualize_embeddings()` -> Creates PCA plot of entity embeddings for insight.
- kg_explorer_app.py
  - `load_rdf_data()` -> Parses RDF triples into visualizable format.
  - `load_model()` -> Loads trained embeddings and metadata for use in app.
  - `extract_embeddings()` -> Extracts entity and relation embeddings from a PyKEEN model.

    - create_feature_vectors() ->Builds numerical features from attributes for UI
    - calculate_similarity_matrix() -> Computes cosine similarity between entities.
    - create_basic_geographic_map() -> Generates interactive map visualizing similarities and topology.
    - create_network_graph() -> Builds NetworkX graph from triples for visualization.
    - visualize_network_plotly() -> Displays the graph in Plotly inside Streamlit.

LO6 (predict_kg.py):

- load_model() → load a trained TransE model and entity/relation mappings from disk
- extract_embeddings() → extract entity and relation vectors for inference
- predict_links() → perform link prediction using vector-based scoring (supports batching and GPU)
- find_similar_entities() → compute embedding-based entity similarity via vector distance
- interactive_prediction() → run link reasoning interactively via user input (loops through prediction/similarity)
- handle_link_prediction() → prompt and handle user input for link prediction
- handle_similarity_search() → prompt and handle user input for similarity search

LO7:

- convert_municipality_data_to_triplets.py:
    - parse_args() → parse command-line inputs like file paths and output format
    - validate_files() → check if required CSV/JSON files are present
    - load_data() → load election, finance, mapping, and adjacency data
    - get_municipality_uri() → generate consistent RDF URIs for municipalities
    - add_typed_literal() → add datatype-aware RDF triples to the batch
    - build_graph() → convert all loaded data into a typed RDF knowledge graph
- train_kg_on_dataset.py:
    - parse_args() → parse training parameters (model dir, epochs, etc.)
    - load_entity_attributes() → load JSON attributes for each entity
    - create_attribute_features() → convert attributes into numeric vectors
    - train_model() → train a TransE model on RDF triples using PyKEEN
    - save_model() → persist model weights and metadata to disk
    - visualize_embeddings() → create PCA plot of entity embeddings
- kg_explorer_app.py
    - load_rdf_data() → parse RDF triples for use in the frontend
    - load_model() → load trained PyKEEN TransE model from disk
    - extract_embeddings() → extract entity and relation embeddings
    - create_feature_vectors() → build feature vectors from attribute data
    - calculate_similarity_matrix() → compute pairwise entity similarity
    - create_basic_geographic_map() → visualize entities and similarity on a map
    - create_network_graph() → build graph structure from RDF triples
    - visualize_network_plotly() → render the graph as an interactive Plotly chart

LO8 (predict_kg.py):

- `load_model()` → load a trained TransE model and attribute data for inference
- `extract_embeddings()` → extract entity and relation embeddings for reasoning tasks
- `predict_links()` → suggest new links using embedding scores and attribute similarity
- `find_similar_entities()` → find similar entities using embedding and attribute distances
- `show_entity_attributes()` → display raw attribute values for a given entity
- `interactive_prediction()` → interactive CLI to test link prediction, similarity, and inspect entities
- `handle_link_prediction()` → process user input for predicting likely triple completions
- `handle_similarity_search()` → process user input to find similar entities based on vectors
- `handle_attribute_display()` → show attribute values in the interactive shell

LO9 - 12: Described in the Document