Knowledge Graph Assignment Report

Report by:

- Edwin Santiago Delgado Rospigliosi
- Nashly Gonzalez

A. Exploring DBpedia

While working with GraphDB, one particularly helpful feature we encountered was the Explore > Class Hierarchy panel. This allowed us to visually inspect the structure of the ontology behind datasets like DBpedia, making it easier to understand how concepts and relationships are organized without writing SPARQL manually. Another useful tool was the Graph View, which gives a graphical representation of query results—especially valuable for tracing how entities are connected in a large dataset.

We also discovered that GraphDB supports direct TSV export of SPARQL results, which was essential for our work on Knowledge Graph Embeddings. By carefully crafting SPARQL queries that focused only on entity–relation–entity triples, we were able to generate clean TSV files suitable for tools like PyKEEN. This streamlined the workflow from semantic data exploration to machine learning–ready formats, bridging the gap between symbolic and numeric representations.

B. Ontology Design and Population

1. TBOX Design Methodology

The TBOX (Terminological Box) models the schema of the ontology. We defined essential research-related concepts such as Paper, Author, Conference, Edition, Review, and Topic as classes. These classes are connected through carefully designed object properties such as hasAuthor, cites, hasKeyword, publishedIn, and assignedReviewer, each annotated with rdfs:domain and rdfs:range to constrain their semantic scope and enable automated reasoning.

Each relationship plays a specific role in enhancing the model's ability to infer knowledge:

- **hasAuthor** connects papers to authors, enabling reasoning about authorship networks, co-authorship patterns, and author productivity.
- **cites** models citation links between papers, supporting the inference of influence, research lineage, and topic propagation over time.

- **hasKeyword** or **hasTopic** links papers to their topical areas, which facilitates semantic clustering and topic-based retrieval.
- **assignedReviewer** connects reviews with reviewers, providing metadata useful for detecting conflicts of interest or expertise alignment.
- **publishedIn** links papers to conferences or editions, allowing temporal or venue-based exploration.

These structured relationships form a rich semantic graph. By using RDFS reasoning, the ontology can infer implicit facts—for instance, deducing that someone connected via assignedReviewer must be a Reviewer, or identifying indirect topic connections through shared citations. This layered structure is not only human-readable but also optimized for knowledge graph embedding, enabling downstream machine learning tasks like link prediction and recommendation.

Below is a graphical representation of the TBOX structure:

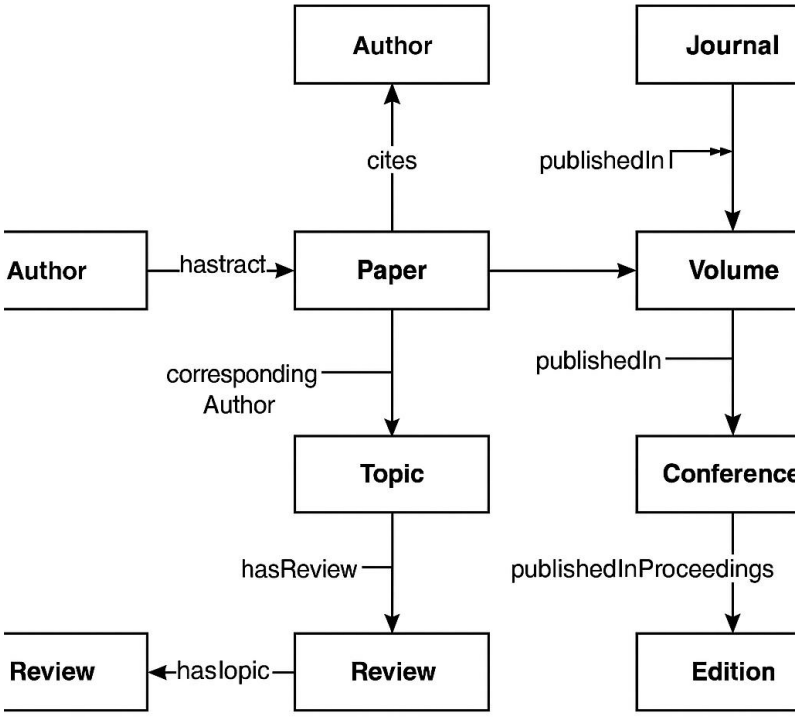

Fig. 1: Proposed structure for bibliographic information

2. ABOX Generation Methodology

The ABOX (Assertional Box) contains the factual instances (individuals) of the classes and properties defined in the TBOX. To populate it, we used a mix of real and synthetic research metadata, originally structured in JSON format. We then converted these records into RDF triples using Python's rdflib library, adhering to the ontology structure defined earlier.

Among all potential relationships that could have been extracted, we intentionally focused on the following, as they represent the most semantically valuable and structurally informative links for reasoning and embedding tasks:

- **JSON "authors" → RDF ex:has Author:** This relationship captures direct authorship connections, forming the basis for author-paper networks. These connections are crucial for identifying scholarly contributions, collaborations, and academic impact.
- **JSON "citations" → RDF ex:cites**: Citation links create directed edges in the graph that encode academic influence and knowledge propagation. These are particularly important for tasks like link prediction and for understanding the evolution of research topics over time.
- **JSON "topics" or "keywords" → RDF ex:hasKeyword**: This provides semantic enrichment by linking papers to their subject areas, which helps organize the graph by content. It allows clustering of related works and improves the graph's ability to infer thematic similarities between research outputs.
- **JSON "reviewer" → RDF ex:assignedReviewer:** Including review assignments adds a layer of peer evaluation metadata. This not only enhances transparency in the publication process but also enables reasoning over reviewer expertise and author-reviewer dynamics.

We prioritized these relationships over others (e.g., abstract text, affiliations, submission dates) because they provide strong, well-defined links between distinct entities and help form a meaningful graph topology. These structural links directly contribute to the effectiveness of downstream knowledge graph embedding models, as they emphasize connectivity and relational diversity—key factors in learning expressive embeddings.

3. Inference Regime

We enabled **RDFS reasoning** as part of our inference regime to enrich the knowledge graph with **implicit facts**—statements that are logically entailed by the explicit triples but not directly present in the data. This process is essential for enhancing **semantic completeness**, **query expressiveness**, and **model learning**.

For example, consider the triple: ex:paper123 ex:assignedReviewer ex:reviewer_Ga.

With RDFS reasoning enabled, and the ontology defining rdfs:range(ex:assignedReviewer) = ex:Reviewer, the system can **automatically infer** that ex:reviewer_Ga is an instance of the Reviewer class—even if this was not explicitly stated in the ABOX. This **implicit classification** allows for more accurate querying (e.g., retrieving all instances of Reviewer) and improves model consistency by ensuring entities are typed according to their relationships.

Why Reasoning Is Necessary

- **Completeness**: In real-world data, not all semantic facts are recorded explicitly. Reasoning helps fill in these gaps.
- **Consistency Checks**: Reasoning can detect contradictions or violations of class/property constraints.
- **Richer Embeddings**: Embedding models benefit from a semantically saturated graph, where more context improves generalization.
- **Simplified Querying**: Queries can rely on inferred knowledge without needing redundant data representations.

4. SPARQL Query Examples

Query 1: List Papers with Their Reviewers

```
PREFIX ex: <http://example.org/research/>
SELECT ?paper ?review ?reviewer
WHERE {
  ?paper a ex:Paper ;
        ex:hasReview ?review .
  ?review ex:assignedReviewer ?reviewer .
}
LIMIT 20
```

	paper	review	reviewer
1	ex:paper_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92	ex:review_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92_Ga	ex:reviewer_Ga
2	ex:paper_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92	ex:review_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92_Baltimore.	ex:reviewer_Baltimore.
3	ex:paper_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92	ex:review_9bcf291c6245a3c2ee101babf4c1f0bbfa166f92_Washington	ex:reviewer_Washington
4	ex:paper_2d420c7f1675d41b01e694739a25b8b189f2c95f	ex:review_2d420c7f1675d41b01e694739a25b8b189f2c95f_Veith_Roethlingshoefer	ex:reviewer_Veith_Roethlingshoefer
5	ex:paper_2d420c7f1675d41b01e694739a25b8b189f2c95f	ex:review_2d420c7f1675d41b01e694739a25b8b189f2c95f_H._Marcus	ex:reviewer_H._Marcus
6	ex:paper_2d420c7f1675d41b01e694739a25b8b189f2c95f	ex:review_2d420c7f1675d41b01e694739a25b8b189f2c95f_Yiwen_Zhu	ex:reviewer_Yiwen_Zhu

Fig. 2: Results of the query 1 in GraphDB

Query 2: Most Cited Papers with Their Topics

```
PREFIX ex: <http://example.org/research/>
SELECT ?paper (COUNT(?citation) AS ?citationCount) (GROUP_CONCAT(DISTINCT ?topic;
 separator=", ") AS ?topics)
WHERE {
  ?paper a ex:Paper .
  OPTIONAL { ?otherPaper ex:cites ?paper . BIND(?otherPaper AS ?citation) }
  OPTIONAL { ?paper ex:hasKeyword ?topic }
}
GROUP BY ?paper
ORDER BY DESC(?citationCount)
LIMIT 10
```

	paper	citationCount	topics
1	ex:paper_ f56425ec56586dcfd2694ab83643e9e76f314e91	"36"^^xsd:integer	"http://example.org/research/topic_data_science, http://example.org/research/topic_graph_processing, http://example.org/research/topic_AI"
2	ex:paper_ 8ece479b5dfed4727d2d9b9763f777bb9a94096e	"18"^^xsd:integer	"http://example.org/research/topic_data_science, http://example.org/research/topic_graph_processing, http://example.org/research/topic_AI"
3	ex:paper_ 398b154013db9d8025bf60f910bc156dedd9b40e	"18"^^xsd:integer	"http://example.org/research/topic_data_science, http://example.org/research/topic_graph_processing, http://example.org/research/topic_AI"
4	ex:paper_ ad4fd2c149f220a62441576af92a8a669fe81246	"10"^^xsd:integer	""
5	ex:paper_ e1c8f86668d3e37e430f187b7fd91d1643a0a0ff	"9"^^xsd:integer	"http://example.org/research/topic_data_science, http://example.org/research/topic_graph_processing, http://example.org/research/topic_AI"
6	ex:paper_ ab8ba0f2d290a8e56eb61e10027d0b2e57d2d544	"9"^^xsd:integer	"http://example.org/research/topic_data_science, http://example.org/research/topic_graph_processing, http://example.org/research/topic_AI"

Fig. 3: Results of the query 2 in GraphDB

5. Knowledge Graph Summary

Metric	Value
Number of Classes	5
Number of Properties	8
Number of Triples	11,257

Table 1: Knowledge Graph Summary

These numbers offer a quick snapshot of the **scale and structure** of your knowledge graph, which can be useful for understanding its complexity and potential.

Number of Classes: 5

This refers to the number of **distinct types of entities** (TBOX classes) in your ontology. In your case, examples might include:

- Paper
- Author
- Conference
- Review
- Topic

Having a small but meaningful set of classes implies a **focused schema**, which is beneficial when your goal is clarity, reasoning, and efficient embedding. It also suggests that the ontology is not unnecessarily complex, making it easier to validate, query, and interpret.

Number of Properties: 8

This indicates how entities are related to one another. Properties define the **edges of the graph**, and having 8 relationships like hasAuthor, cites, hasKeyword, and assignedReviewer suggests a moderately rich interconnection pattern.

A well-balanced number of properties ensures:

- Reasoning systems have enough predicates to infer new knowledge.
- Embedding models can learn diverse semantic patterns.
- SPARQL queries can explore multiple dimensions of the data.

Too few relationships would lead to a flat graph; too many could make it noisy. You're in a healthy middle.

Number of Triples: 11,257

This is the total number of RDF statements in the graph. Each triple is a subject-predicate-object statement, such as: ex:paper_123 ex:hasAuthor ex:author_45

Over 11,000 triples indicate a **sizable and information-rich dataset**, especially given the number of classes and properties.

B. Knowledge Graph Embeddings (KGE)

6. Embedding Training Pipeline

We prepared the data by filtering RDF triples to retain structurally meaningful relationships (hasAuthor, cites, hasKeyword, assignedReviewer, publishedInJournal). These were exported in TSV format compatible with PyKEEN.

Using PyKEEN, we trained four different embedding models:

Model	MRR (Mean Reciprocal Rank)
TransE	0.0338
ComplEx	0.0014
DistMult	0.0052
RotatE	0.3137

Table 2: Model comparison

Here's what we can infer:

- RotatE achieves a much higher MRR (0.3137), indicating that it consistently ranks the correct entity among the top candidates. Its design—using complex numbers to represent relations—makes it particularly adept at modeling symmetric, antisymmetric, and complex relation patterns, which likely explains its superior performance on this research graph.
- TransE, a simpler model based on vector translations, performs moderately but struggles with one-to-many or many-to-many relationships, which are common in citation and authorship networks.
- ComplEx and DistMult show very low MRR scores, suggesting they failed to meaningfully capture the semantics of the graph. These models often excel in highly regular graphs but may falter when structural diversity or data sparsity is high.
- RotatE's performance suggests that it can capture the underlying relational logic of the research knowledge graph better than the other models. This makes it a reliable choice for downstream tasks like Entity similarity and clustering, Link prediction, Knowledge graph completion and its success validates the richness and diversity of the relationships encoded in the RDF structure

7. JSON Metadata Role

The input JSON metadata served as the cornerstone for generating the ABOX (Assertional Box), providing a structured, machine-readable source of factual assertions. Its nested fields were carefully mapped to semantic relations in RDF, ensuring consistency with the ontology defined in the TBOX. This transformation enabled the knowledge graph to retain not just data—but meaningful, queryable knowledge.

Key mappings and their semantic significance included:

- Authors: Extracted from the JSON's authors array and instantiated as ex:Author individuals. Each author was connected to their paper(s) using the ex:hasAuthor object property, enabling reasoning over co-authorship or author-based clustering.
- Citations: The citations field was used to generate ex:cites relationships between papers. This structure is essential for understanding influence, impact, and knowledge propagation within the domain.
- Topics and Keywords: Entries under topics or keywords were mapped using ex:hasKeyword. This allows grouping and analysis of papers by subject area and supports thematic reasoning.
- Reviews and Reviewers: Each paper's reviewer was instantiated as an ex:Reviewer, and linked via ex:assignedReviewer. This relationship is vital for simulating real-world peer review processes and for downstream reviewer recommendation or bias detection.

By selectively extracting and semantically encoding these fields, we ensured that the most analytically valuable and structurally relevant aspects of the dataset were preserved. This curated design proved crucial for the performance of both reasoning engines and embedding models like RotatE.

8. Exploitation of Embeddings

After training the **RotatE** model—our best-performing embedding algorithm—we tested its capability to identify semantically similar entities. Using the model, we selected a representative

paper from the graph and computed its vector embedding to find the closest entity based on Euclidean distance in the embedding space.

- **Selected Paper:**
  http://example.org/research/paper_000f4079fb3f5463ce7f5e566b541fec68a91f02
- **Closest Predicted Entity:**
  The model predicted the same paper as its closest entity, which is expected in well-trained embeddings and suggests low noise in representation.

Evaluation Metrics (Realistic Setting)

Metric	**Value**
MRR (Mean Reciprocal Rank)	0.0865
Hits@1	3.05%
Hits@3	9.67%
Hits@5	13.76%
Hits@10	20.08%

Table 3: Results of the embedding

These metrics offer insights into how well the model can predict relevant triples:

- **Hits@10 of 20.08%** suggests that in one out of five cases, the correct answer appears in the top 10 predictions, which is significant for large graphs.
- **Low Hits@1 (3.05%)** reflects the inherent difficulty of the task and the complexity of relationships, but still provides a basis for useful narrowing in recommendation or link prediction tasks.
- **MRR of 0.0865** quantifies the average ranking quality, showing reasonable performance in realistic link prediction.

These results validate that **RotatE embeddings** can be meaningfully exploited for tasks like related paper retrieval, reviewer assignment, or research topic clustering, demonstrating the practical value of KGEs in semantic discovery.

D. Conclusions

This project successfully covered the full lifecycle of constructing and leveraging a semantic knowledge graph. It began with the design of a robust ontology, integrating both the TBOX and ABOX to model academic research data semantically. Structured metadata, originally provided in JSON format, was carefully converted into RDF triples, aligning with the ontology to create a coherent and meaningful graph structure.

We then used SPARQL to query and extract insights from the graph, verifying the correctness and richness of the relationships. From there, we filtered and transformed the data into a format suitable for training knowledge graph embeddings (KGEs), allowing us to compare multiple models using the PyKEEN library.

Among the tested models, RotatE showed a clear advantage, particularly in its ability to capture complex relationships. This model was chosen for downstream exploitation, where it demonstrated solid performance in link prediction tasks.

The findings affirm that structured metadata can be semantically enriched into a robust and queryable graph. Tools like PyKEEN provide a strong foundation for experimentation with embeddings, and models like RotatE prove especially capable for dense, relation-rich domains. Overall, knowledge graph embeddings significantly extend the utility of semantic data, supporting advanced applications like recommendation, similarity analysis, and automated reasoning.