SDM Project 2

Giovanni Spisso
Hang Yao

June 14, 2025

B.1 TBOX Definition

Figure 1 shows the graphical representation of the TBOX used in our model. For clarity, only the schema-level structure is reported, while metamodeling elements have been omitted. Classes are depicted in blue, properties in green, and `XSD:string` in yellow.

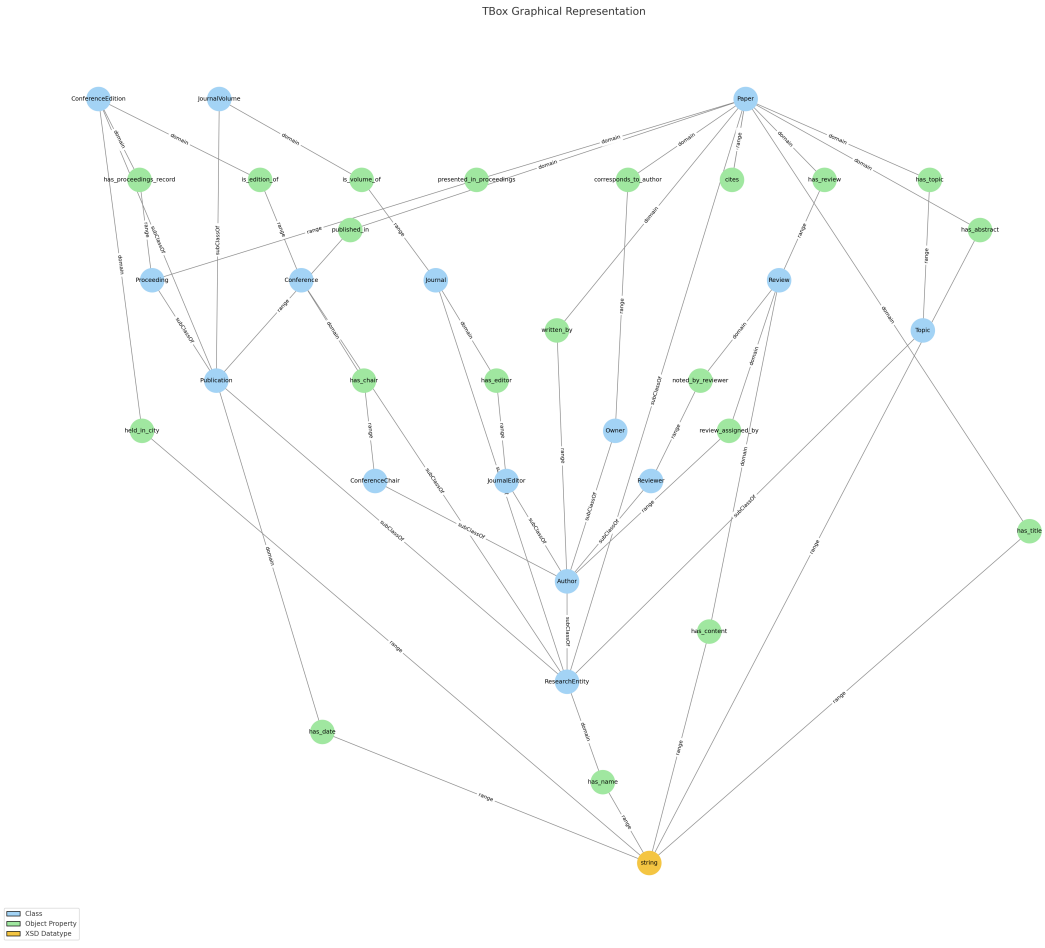

Figure 1: Graphical representation of the TBOX. Classes are shown in blue, properties in green.

B.2 ABOX Definition

Note that some logical constraints stated in the lab instructions, such as "a paper has exactly one corresponding author" or "each paper is reviewed by three reviewers", cannot be formally enforced using **RDFS** alone. These would require **OWL** axioms (e.g., cardinality constraints), which are beyond the scope of this lab. We assume our input data respects these constraints. Furthermore, we consider that the conference chairs and journal editors are authors from other conferences or journals—specifically, authors who did not publish or present papers in the same conference or journal for which they served as chair or editor.

Methodology for ABOX Construction To construct the ABOX, we reused the dataset from Lab 1, which was stored as a CSV file (`research_papers.csv`) containing metadata about scientific papers. These data were loaded with `pandas` and then transformed into RDF triples using the `rdflib` library. Each row represents a research paper and includes all the information related to that paper. Each instance is assigned a URI based on the namespace `http://example.org/-research_paper/`, using a URI-safe encoding of either an identifier or descriptive content. For entities such as papers, the URI is directly derived from their unique IDs provided in the dataset. In contrast, for entities like authors, journals, conferences, volumes, editions, etc., which do not have predefined IDs, we generate consistent and unique URIs by combining descriptive elements (e.g., names, years, or other contextual data) to ensure their distinctiveness.

We adopt a single modeling approach: relationships between instances (such as `written_by`, `cites`, etc.) are created using the `create_property_instance` function. This function asserts RDF triples linking a subject and an object, with the object being either another instance or a literal value. Since all the properties used are defined in the TBOX with appropriate domain and range declarations, GraphDB automatically infers the types of connected resources. This strategy allows us to avoid explicitly declaring `rdf:type` triples under the RDFS inference regime.

The final RDF graph was serialized into Turtle format as `12_B.2_SpissoYao_abox.ttl`, and uploaded into GraphDB alongside the corresponding TBOX.

Inference Regime We used the **RDFS (Optimized)** inference regime in GraphDB. This reasoning regime supports inference based on core RDF Schema constructs, including `rdfs:subClassOf` (e.g., inferring that every instance of `Reviewer` is also an `Author`), as well as `rdfs:domain` and `rdfs:range`, which allow the class membership of subjects and objects to be inferred from their use in property assertions.

Thanks to this inference regime, we avoided manually asserting `rdf:type` triples for all instances, since every instance is part of at least one property or is a subclass of a superclass.

Summary Statistics The table below presents summary statistics for the resulting ABOX. These values were computed after RDF triple generation and inference using GraphDB. To maintain focus on the most relevant information, we report only a subset of triples, specifically those considered most interesting. Some triples, such as those involving the property `has_abstract`, were omitted due to their triviality (e.g., `has_abstract` appears once for each paper, thus equaling the total number of papers).

Importantly, the reported counts for classes and properties are restricted to those defined within the custom namespace `ex:`, and therefore do not reflect the full set of entities present in the ontology.

Element	Count
Number of Classes	15
Number of Properties	20
Paper Instances	246
Author Instances	700
Publication Instances	127
Topic Instances	26
Conference Instances	5
Journal Instances	5
Triples for `written_by`	1928
Triples for `cites`	11189
Triples for `has_topic`	1133
Triples for `published_in`	212
Triples for `presented_in_proceedings`	153
Total Triples (ABOX)	24408

Table 1: Summary statistics of the ABOX

B.3

Query One: Find reviewers and their reviewed topics who were assigned by a conference chair to review a paper on topics they have previously published on. The reviewed paper must have been presented at a conference, and the related publication must have occurred in the same year or earlier than the corresponding conference edition.

This query aims to verify the alignment between reviewers' expertise and the topics of the papers they are assigned to review. By selecting reviewers who have previously published on the same topics, it provides insight into the quality of the reviewer assignment process. The temporal constraint ensures that the reviewer's expertise precedes or coincides with the corresponding conference edition, avoiding the inclusion of publications that occurred after the review assignment.

```
PREFIX : <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/research_paper/>

SELECT DISTINCT ?reviewer_name (GROUP_CONCAT(DISTINCT ?topic_name; separator=",␣") as ?
    topics)
WHERE {
    ?paper1 ex:has_review ?review .
    ?review ex:noted_by_reviewer ?reviewer;
              ex:review_assigned_by ?chair .
    # Since we only return reviewers from conferences, all the reviewed papers are assumed
        to have been presented in the conference proceedings.
    ?chair rdf:type ex:ConferenceChair .

    ?paper1 ex:has_topic ?topic;
              ex:presented_in_proceedings ?proceeding .

    # Get the topic and reviewer name for better query visualization
    ?topic ex:has_name ?topic_name .
    ?reviewer ex:has_name ?reviewer_name .

    ?conference_edition ex:has_proceedings_record ?proceeding;
                                  ex:has_date ?year1 .

    ?paper2 ex:written_by ?reviewer;
              ex:has_topic ?topic;
              ex:published_in ?publication2 .
```

```
    ?publication2 ex:has_date ?year2 .

        BIND(xsd:integer(xsd:decimal(?year1)) AS ?year1_int)
        BIND(xsd:integer(xsd:decimal(?year2)) AS ?year2_int)

    FILTER (?paper1 != ?paper2 && ?year1_int >= ?year2_int)
}
GROUP BY ?reviewer_name
```

Listing 1: Query 1

Listing 1 reports the SPARQL code for the query. Table 2 presents a sample of the retrieved results. For instance, `Flavius Frasincar` has reviewed papers on topics such as `language modeling`, `data querying`, and `deep learning`, which are also topics he has written about in his own publications.

reviewer_name	**topics**
Flavius Frasincar	language modeling, data querying, deep learning
Mitsuru Ikeda	sentiment analysis, classification, IoT in automation
Gianluigi Liva	convolutional networks
Yixue Li	visual tracking
Chih-Ming Chen	information extraction, data modeling

Table 2: Some results of Query 1

Query 2: Self-citations across different topics. This query retrieves cases of self-citation—returning attributes such as author, conference, citing paper, citing paper topics, cited paper, and cited paper topics—in which an author cites one of their own previously written papers. The cited paper must have been presented in an earlier edition of the same conference series, and the two papers must address different research topics. This helps to detect topic shifts in an author's research trajectory or potential overuse of citations across unrelated work.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/research_paper/>
SELECT ?author_name ?conference_name
          ?citing_paper_title (GROUP_CONCAT(DISTINCT ?citing_topic_name; separator=",␣")
              as ?citing_topics)
      ?cited_paper_title (GROUP_CONCAT(DISTINCT ?cited_topic_name; separator=",␣") as ?
          cited_topics)
WHERE {
    # Citing paper (newer)
    ?citing_paper ex:written_by ?author ;
                  ex:cites ?cited_paper ;
                  ex:has_topic ?citing_topic ;
                  ex:presented_in_proceedings ?proceeding1 .

    ?conference_edition1 ex:has_proceedings_record ?proceeding1 ;
                         ex:is_edition_of ?conference ;
                         ex:has_name ?edition_name1 .

    BIND(xsd:integer(STRAFTER(?edition_name1, "Edition␣")) AS ?edition_number1)

    # Cited paper (older, same author, same conference series)
    ?cited_paper ex:written_by ?author ;
                 ex:has_topic ?cited_topic ;
                 ex:presented_in_proceedings ?proceeding2 .

    ?conference_edition2 ex:has_proceedings_record ?proceeding2 ;
                         ex:is_edition_of ?conference ;
                         ex:has_name ?edition_name2 .
```

```
    BIND(xsd:integer(STRAFTER(?edition_name2, "Edition␣")) AS ?edition_number2)

    # Get names for display
    ?author ex:has_name ?author_name .
    ?conference ex:has_name ?conference_name .
    ?citing_topic ex:has_name ?citing_topic_name .
    ?cited_topic ex:has_name ?cited_topic_name .
    ?citing_paper ex:has_title ?citing_paper_title .
    ?cited_paper ex:has_title ?cited_paper_title .

    # Ensure the citing paper is present in a newer edition of the conference
    FILTER (?edition_number1 > ?edition_number2)
    # Ensure NO topics overlap between the two papers
    FILTER NOT EXISTS {
        ?citing_paper ex:has_topic ?shared_topic .
        ?cited_paper ex:has_topic ?shared_topic .
    }
}
GROUP BY ?author_name ?conference_name ?citing_paper_title ?cited_paper_title
```

Listing 2: Query 2

Listing 2 shows the code for this second query. To identify papers that address completely different topics, we use the `FILTER NOT EXISTS` clause, which ensures that there are no overlapping topics between the citing and cited papers. For example, Table 3 shows that the author `K. W. Chew` presented a paper citing a previous one presented in the same conference, the `International Conference on Management of Data`, where the topics of both papers are entirely disjoint.

author_name	conference_name	citing_paper_title	citing_paper_topics	cited_paper_title	cited_paper_topics
Joel W. Burdick	international conference on management of data	Towards runtime testing in automotive embedded systems	visual tracking, regression, intelligent control systems, convolutional networks, image segmentation	DQAINF: an algorithm for automatic integration of infinite oscillating tails	information extraction, data management, data modeling, data processing
K. W. Chew	international conference on management of data	Circuit Delay Models and Their Exact Computation Using Timed Boolean Functions	big data, language modeling, convolutional networks, sentiment analysis, data querying	DQAINF: an algorithm for automatic integration of infinite oscillating tails	information extraction, data management, data modeling, data processing

Table 3: Some results of Query 2

C.1

To construct the input for the **KGE** training, we selected only the semantically meaningful object properties from the knowledge graph. The aim is to include triples that represent the relational structure between entities, as these are the most informative for learning embeddings with models.

For the exploitation task (Section C.4), we developed a classifier—specifically a **Multi-Layer Perceptron** (**MLP**)—to predict paper publication type. The target labels were the publication type categories (several journals/conferences), which were not included as explicitly relationships in the KGE training.

We retained the predicates `ex:has_topic`, `ex:published_in`, and `ex:presented_in_proceedings` as they express core structural relationships in the exploitation domain. These relationships enable the learning of meaningful patterns: papers with similar topics are likely to be published in similar conference/journal, and papers published in the same journal volumes or conference editions share inherent similarities in their publication context. The generated embeddings capture these latent patterns, allowing papers to cluster based on their topical and venue characteristics.

In contrast, we excluded datatype properties such as `ex:has_title`, `has_name`, and `has_date` from the KGE training. These properties provide literal values that do not contribute to the graph's structural semantics or relational patterns. We also excluded other relational properties that do not provide useful patterns for predicting publication type.

The final RDF triples were exported from GraphDB using a SPARQL query that filters for the selected predicates, and saved in `TSV` format for compatibility with PyKEEN. We then imported the data into Python using `pandas`, and generated stratified train/test splits using `PyKEEN`'s `TriplesFactory.split()` method. Stratification ensures that the distribution of relation types is preserved across all splits, which is essential for robust KGE training and evaluation.

C.2

The most basic model We trained a TransE model on our KG using PyKEEN. Given a paper p and the relation `ex:cites`, the most likely embedding vector of a cited paper is approximated as:

$$\vec{p} + \vec{r}_{\text{cites}} \approx \vec{cited\_paper}$$

To find the cited author, we apply the same logic:

$$\vec{cited\_paper} + \vec{r}_{\text{written\_by}} \approx \vec{author}$$

Using the resulting author vector, we compute the Euclidean distance between it and all entity embeddings in the KG to identify the closest matching author.

We note that the exported query from the previous section did not include the properties `ex:cites` and `ex:written_by`. To address this, we generated an updated query, based on the previous one, with these two properties included. The resulting query output was saved as `C.2_-query.tsv` and used for the experiments in this subsection.

We selected the embedding of the paper `ex:4ab60d22-67d1-4683-8648-f1f2601d2ce0` as the citing paper. Using the first approximation formula, we computed an estimated embedding vector representing a potential cited paper. We then applied the Euclidean distance to identify the closest embedding to this approximation, excluding only the citing paper itself from the list of candidate entities. The resulting entity was indeed a paper, `ex:4ab4c17b-a9ac-4f37-8e71-5977d5db5a07`, which corresponds to an actual cited paper, as shown in Figure 2.

Figure 2: Cited Paper Triple

Next, we applied the second approximation formula to infer the author of the cited paper, excluding only the citing paper from the list of candidate, as it served as the initial embedding. The predicted author was `ex:AUTHOR_331`, who is indeed one of the actual authors, as shown in Figure 3.

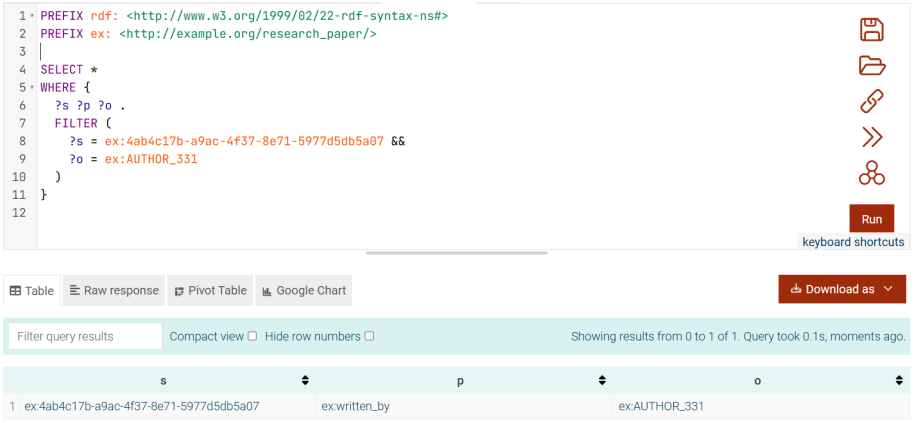

Figure 3: Author Triple

Improving TransE **Task 1.** TransE models relations as translations in vector space:

$$\vec{h} + \vec{r} \approx \vec{t}$$

Given the triples:

- (Author1, writes, Paper1)
- (Author2, writes, Paper1)
- (Author1, writes, Paper2)

TransE will try to satisfy:

$$\vec{Author1} + \vec{writes} \approx \vec{Paper1} \quad \text{and} \quad \vec{Author2} + \vec{writes} \approx \vec{Paper1}$$

$$\vec{Author1} + \vec{writes} \approx \vec{Paper2}$$

However, this creates a contradiction: the same vector $\vec{Author1} + \vec{writes}$ must be close to both $\vec{Paper1}$ and $\vec{Paper2}$, and simultaneously, $\vec{Author2} + \vec{writes}$ must also be close to $\vec{Paper1}$. The result is that the embeddings for Paper1 and Paper2 are forced close together, and the same holds for Author1 and Author2.

This illustrates a core limitation of TransE: it struggles with modeling **One-to-Many**, **Many-to-One**, and **Many-to-Many** relationships because it uses a single translation vector for each relation.

Task 2. We decide to use model TransH, which projects entity embeddings onto a relation-specific hyperplane before applying the translation.

Formally, for a relation r, there is a hyperplane defined by a normal vector $\vec{w}_r$ and a translation vector $\vec{d}_r$. The entity embeddings are projected onto this hyperplane before the translation is applied.

This mechanism allows TransH to assign different semantics to the same relation depending on the entities involved, improving the model's ability to deal with One-to-Many and Many-to-One relationships more flexibly.

Task 3. In TransE, the score of a triple (h, r, t) is based on:

$$\text{score} = -\|\vec{h} + \vec{r} - \vec{t}\|$$

For a symmetric relation like `collaboratesWith`, we would expect:

$$\text{score}(Author1, collaboratesWith, Author2) = \text{score}(Author2, collaboratesWith, Author1)$$

However, unless the embeddings and the relation vector satisfy specific constraints (e.g., $\vec{r} = \vec{0}$ and $\vec{Author1} = \vec{Author2}$), the scores will generally be different. Therefore, TransE does not naturally support symmetry in relations.

Task 4. In TransE, the translation $\vec{h} + \vec{r} \approx \vec{t}$ breaks symmetry because reversing the head and tail leads to a different vector operation: $\vec{t} + \vec{r} \approx \vec{h}$ would only hold if $\vec{h} = \vec{t}$ or $\vec{r} = \vec{0}$, which is rarely the case.

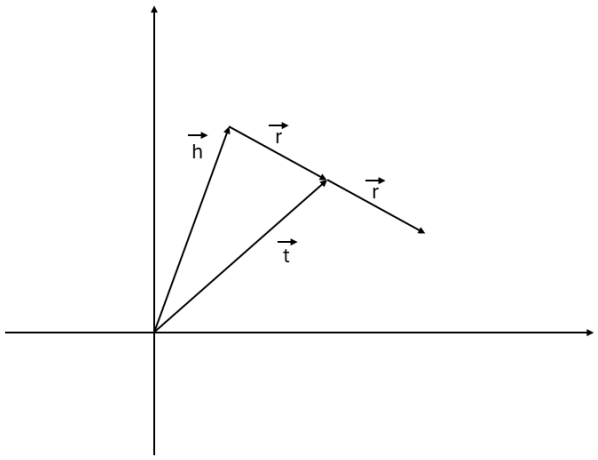

Figure 4: TransE Symmetry (not respected)

As shown in the diagram, the path from $\vec{h} \to \vec{t}$ via $\vec{r}$ is not equal in reverse unless the embedding space collapses to a degenerate configuration.

Task 5. RotatE models relations as rotations in complex space: $\vec{t} = \vec{h} \circ \vec{r}$, where $\circ$ is element-wise complex multiplication. For symmetry to hold:

$$\vec{h} \circ \vec{r} = \vec{t} \quad \text{and} \quad \vec{t} \circ \vec{r} = \vec{h} \Rightarrow \vec{r} \circ \vec{r} = 1$$

This is satisfied when $\vec{r}$ is a 0° or a 180° rotation (i.e., multiplication by -1 in complex space). Hence, symmetry is achieved when the relation vector is an involutive rotation.

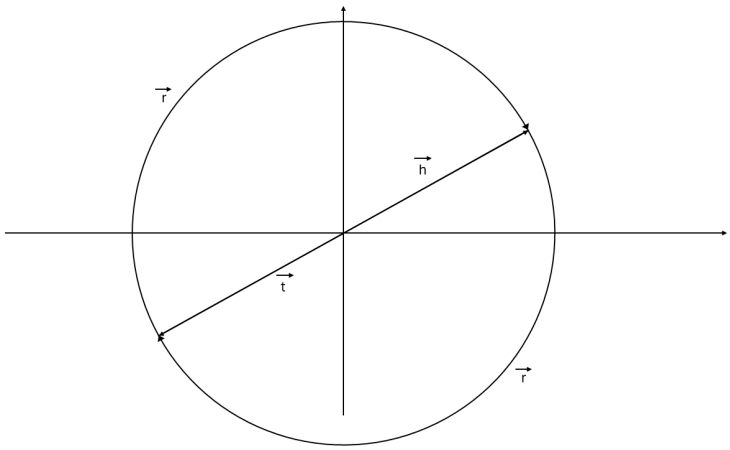

Figure 5: RotatE Symmetry (respected)

Task 6. ComplEx extends DistMult into the complex space. Its scoring function:

$$\mathrm{Re}(\langle \vec{h}, \vec{r}, \bar{\vec{t}} \rangle)$$

is able to model symmetric, anti-symmetric, and asymmetric relations, depending on the values of $\vec{r}$ (and its imaginary part).

Unlike TransE, it does not rely on directional vector translation, and thus can inherently capture symmetry without requiring any special tuning or parameter constraints.

C.3

In this section, we evaluate the following **KGE** models: `TransH`, `RotatE`, `ComplEx`, and `ConvB`. These models were selected for their capacity to address specific limitations of the simpler `TransE` model. For each model, we performed dedicated hyperparameter tuning using the *hpo_pipeline()* function [1], aiming to identify an optimal configuration. We then compared the evaluation results to determine the best-performing model.

In addition to model-specific settings, we adopted a shared hyperparameter tuning framework across all models:

- `sampler`: Set to `tpe`, a more efficient alternative to random search, and the default in `PyKEEN`.
- `n_trials`: Fixed at 30, indicating that 30 distinct hyperparameter configurations were evaluated per model.
- `num_epochs`: Tuned to assess the impact of training duration on performance.
- `optimizer`: Used `Adam`, with hyperparameters such as `lr` (learning rate) and `weight_decay` tuned. An appropriate learning rate accelerates convergence, while weight decay helps prevent overfitting.
- `negative_sampler`: Configured as `basic`, with various values of `num_negs_per_pos` explored. The number of negative samples directly influences both model quality and training time: higher values provide stronger learning signals but increase computational cost.
- `metric`: Set the default evaluation (objective) metric to `both.realistic.inverse_harmonic_mean_rank`, which corresponds to the inverse harmonic mean rank—equivalent to the mean reciprocal rank (MRR). This metric reflects the overall ranking quality by aggregating the reciprocal of the predicted ranks for both head and tail entities across all test triples and tries to maximize during optimization.

We focused only on trials with state COMPLETE, as those marked PRUNED were terminated early because the optimization algorithm determined that continuing training with those specific hyperparameters was unlikely to improve the overall **HPO** (Hyper-Parameter Optimization) objective.

Moreover, we used the dataset introduced in Section C.1 (`C.1_query.tsv`) to select the best-performing model for the exploitation task. The dataset was divided into training, validation and test sets. The training set was used to fit the model during hyperparameter tuning, the validation set served to select the optimal configuration, and the test set was reserved for the final evaluation. This approach helps to ensure that the selected model generalizes well to unseen data.

TransH Model As discussed earlier, `TransH` extends `TransE` by allowing entities to have different vector representations depending on the relation [2]. This enables the model to better handle **One-to-Many** and **Many-to-One** relationships, which are problematic for simpler translation-based models.

For `TransH` model, we experimented with different embedding sizes (128, 192, and 256). Additionally, the `hpo_pipeline()` automatically tuned several hyperparameters, including `loss.margin`, `model.scoring_fct_norm`, `regularizer.weight`, and `training.batch_size`. As illustrated in Figure 6, we only considered the trials with the state COMPLETE. Among them, **trial 1** achieved the highest objective score(0.074748), with its optimal parameter configuration detailed in Table 4.

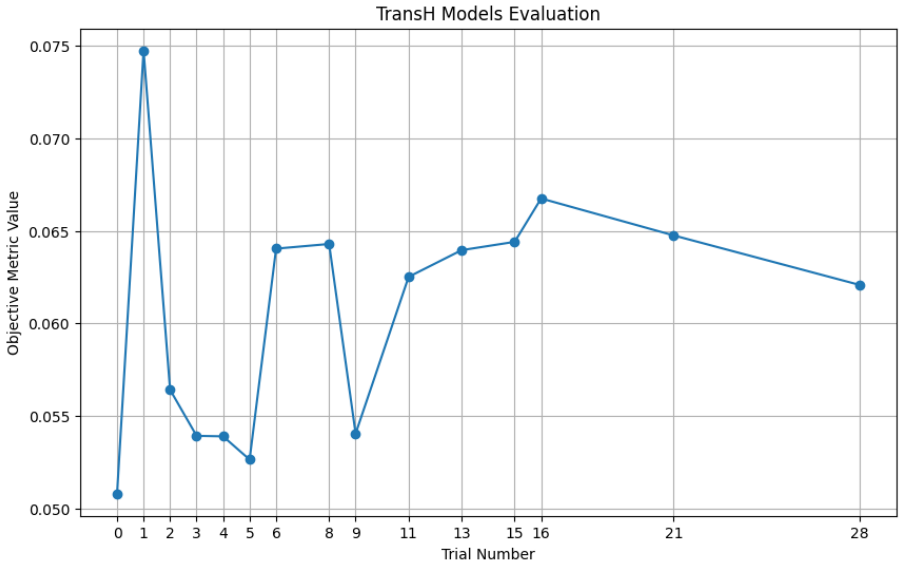

Figure 6: TransH Models Evaluation

Parameter	Optimal Value
loss.margin	0.211326
model.embedding_dim	192
model.scoring_fct_norm	2
negative_sampler.num_negs_per_pos	4
optimizer.lr	0.000655
optimizer.weight_decay	0.000231
regularizer.weight	0.646674
training.batch_size	16
training.num_epochs	100

Table 4: TransH Optimal Parameters

RotatE Model We selected `RotatE` for its ability to model both symmetric and antisymmetric relationships. It represents relations as rotations in the complex vector space, allowing it to capture a wider range of relational patterns, including composition and inversion [3].

For the `RotatE` model, we similarly explored only variations in the embedding size. Figure 7 shows that **trial 22** achieved the highest score of 0.100515. The corresponding optimal configuration is shown in Table 5.

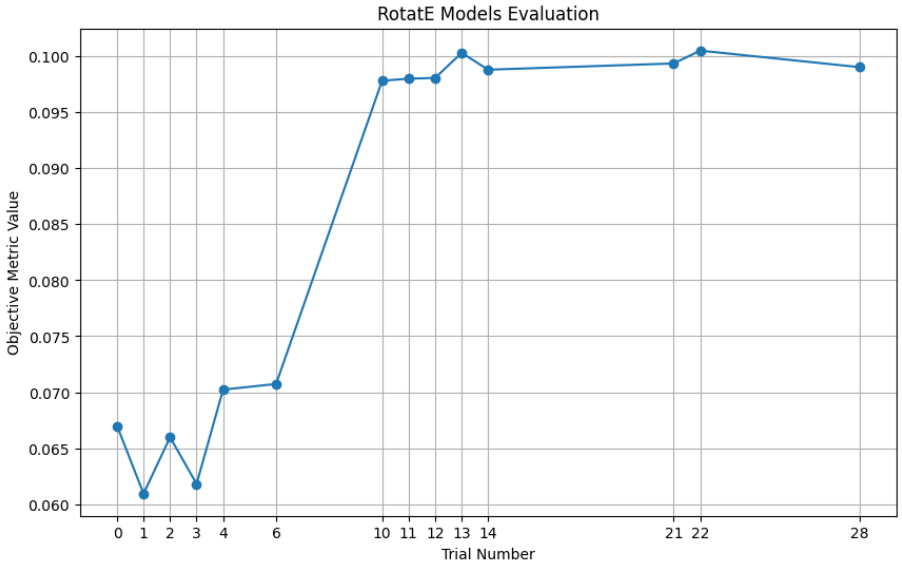

Figure 7: RotatE Models Evaluation

Parameter	Optimal Value
loss.margin	2.625342
model.embedding_dim	256
negative_sampler.num_negs_per_pos	7
optimizer.lr	0.000991
optimizer.weight_decay	0.000634
training.batch_size	16
training.num_epochs	150

Table 5: RotatE Optimal Parameters

ComplEx Model `ComplEx` extends `DistMult` into the complex space, which enables it to model both symmetric and antisymmetric relations [4]. Its scoring function accounts for the directionality of relationships, which makes it suitable for more realistic and varied knowledge graphs.

For the `ComplEx` model, we similarly explored different embedding sizes. Figure 8 shows that **trial 29** achieved the highest score of 0.023987. The corresponding optimal configuration is shown in Table 6.

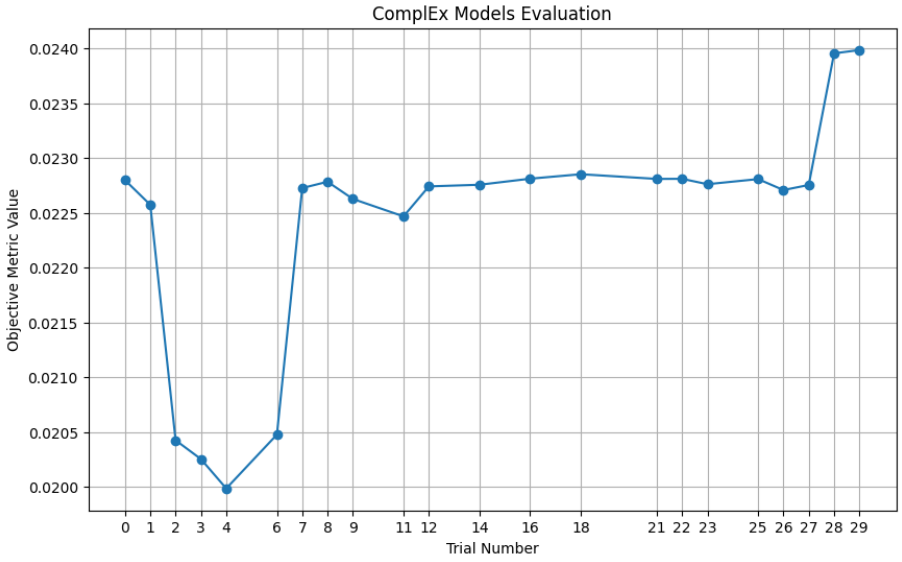

Figure 8: ComplEx Models Evaluation

Parameter	Optimal Value
model.embedding_dim	192
negative_sampler.num_negs_per_pos	1
optimizer.lr	0.000725
optimizer.weight_decay	0.000585
training.batch_size	16
training.num_epochs	50

Table 6: ComplEx Optimal Parameters

ConvKB Model We included `ConvKB` to explore a neural-based approach. Its convolutional architecture enables it to learn non-linear and more abstract interactions between head, relation, and tail entities, thus potentially capturing hidden semantic patterns that are not easily modeled by linear transformations [5].

For the `ConvKB` model, we varied the embedding size, hidden dropout rate, and number of filters, given that this model employs as convolutional neural network architecture for predictions and therefore requires more careful hyperparameter tuning. As illustrated in Figure 9, **trial 15** achieved the best performance with a score of 0.090764. The details of its configuration are provided in Table 7.

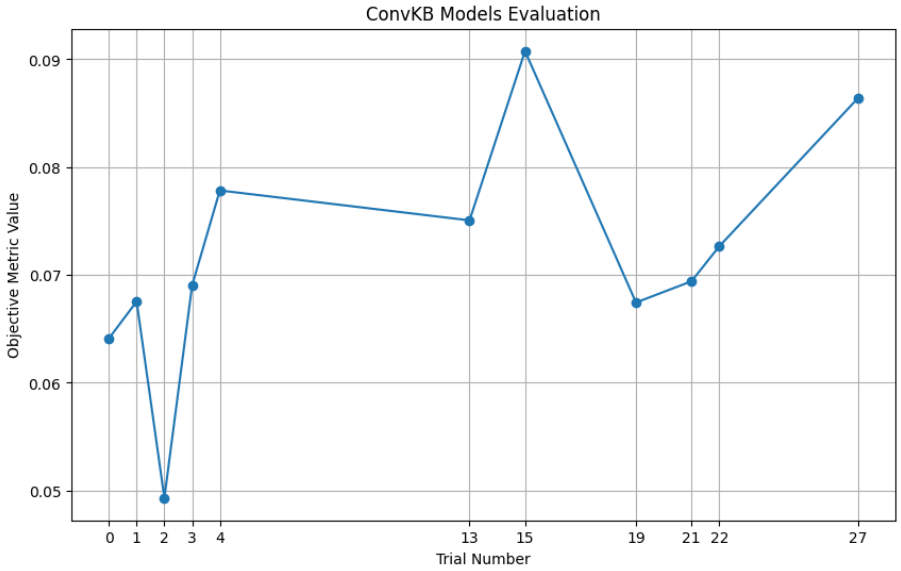

Figure 9: ConvKB Models Evaluation

Parameter	Optimal Value
loss.margin	0.537169
model.embedding_dim	192
model.hidden_dropout_rate	0.4
model.num_filters	96
negative_sampler.num_negs_per_pos	7
optimizer.lr	0.000305
optimizer.weight_decay	0.000113
regularizer.weight	0.677800
training.batch_size	128
training.num_epochs	50

Table 7: ConvKB Optimal Parameters

Model Selection We compared the optimal configurations of the different models, as summarized in Table 8. Among them, the `RotatE` model achieved the highest score according to the `both.realistic.inverse_harmonic_mean_rank` metric, followed by `ConvKB`. Although `RotatE` outperformed the others, the overall metric values remained relatively low across all models. This may be attributed to the inherent limitations of the dataset—which contains noisy and sparsity—as it was primarily generated using **synthetic data**.

Despite the low overall scores (noting that `both.realistic.inverse_harmonic_mean_rank` ranges from 0 to 1), we selected `RotatE` as the best-performing model for use in the next section.

It is important to note that the other models may require more extensive hyperparameter tuning to reach their full potential, particularly due to their increased architectural complexity, specifically `ComplEx` and `ConvKB`. For example, `ConvKB` likely demands more careful tuning of parameters, such as the dropout rate and the number of convolutional filters. According to the original paper, it also benefits from initializing entity and relation embeddings with pretrained embeddings from `TransE` to enhance robustness and performance. Similar considerations may apply to the other models, potentially explaining their relatively lower performance under the current tuning scope, in addition to the limitations provided by the dataset itself.

Model	Objective Metric Value
TransH	0.074748
RotatE	0.100515
ComplEx	0.023987
ConvKB	0.090764

Table 8: Comparison of Models

Final Evaluation We evaluated the performance of the best-performing model—`RotatE`—using the test set. As shown in Table 9, we focused exclusively on the `both realistic` evaluation metrics. This setting was chosen to assess the model's generalization capability across both head and tail prediction tasks, using the `realistic` rank strategy. This rank strategy represents the average of the optimistic and pessimistic ranks and corresponds to the expected value over all permutations that preserve the sort order [6].

The results in the table indicate that the `Inverse Harmonic Mean Rank` on the test set is close to its value on the validation set, suggesting that the model does not suffer from overfitting. However, the other metrics show less favorable results. For instance, the `Hits@5` metric, which reflects the proportion of triples where the true missing entity appears in the top 5 predicted candidates, is only 14.95%. Additionally, the `Arithmetic Mean Rank` is 63.50, meaning that on average, the correct entity is ranked 64th.

These relatively low performance scores may be attributed to the limited quality and completeness of the dataset, as previously discussed. The synthetic nature of the original data likely introduced noise and sparsity, which impeded the model's ability to learn more accurate representations.

Metric	Score
variance	4808.25
arithmetic mean rank	63.502579
inverse harmonic mean rank	0.098996
hits@5	0.149485

Table 9: Both Realistic Metrics of RotatE

C.4

As we explained in Section C.1, we built a publication-type classifier. We used the KGEs generated by the RotatE model to perform a multiclass classification of research papers based on their publication venues. The objective was to predict the specific journal or conference where each paper was published, even though this information was not explicitly represented in the knowledge graph used to train the KGE model. This highlights the purpose of the classifier: to learn meaningful patterns from the embeddings that enable accurate predictions of publication venues.

The embeddings were first extracted from the trained RotatE model for each paper entity. We used RotatE since it has proven to be the best model in Section C.3. As RotatE embeddings are complex-valued, we transformed each embedding into a real-valued vector by concatenating its real and imaginary parts, thus making them suitable for input into machine learning models.

We merged these embeddings with the original paper metadata, which includes publication information. Each paper was assigned a label corresponding to its journal or conference name, depending on where it was published. The resulting dataset consisted of embedding vectors as features and venue names as target labels.

We trained a Multi-Layer Perceptron (MLP) classifier—configured with parameters such as `hidden_layer_sizes`, `random_seed`, and `learning_rate`[7]—on these data to learn the mapping from embedding representations to publication venues. The model was evaluated on a held-out test set and achieved a classification accuracy of 0.44 (as shown in Table 10). Although this value may seem modest, it is important to note that the task involved 10 classes and only 246 paper samples. We believe that with a larger dataset, the model performance could significantly improve.

This experiment demonstrates that the embeddings learned by the KGE model successfully capture meaningful semantic information related to publication venues, even though such information was not directly encoded in the graph structure during embedding training.

Class	Precision	Recall	F1-score	Support
International Journal of Computer Aided Engineering and Technology	0.00	0.00	0.00	1
International Journal of Intelligent Systems Technologies and Applications	1.00	0.20	0.33	5
International Journal of Learning Technology	0.20	0.50	0.29	2
International Journal of Satellite Communications and Networking	0.00	0.00	0.00	4
Journal of the Brazilian Computer Society	0.20	0.20	0.20	5
euromicro conference on real-time systems	0.25	0.25	0.25	8
international conference on computer graphics and interactive techniques	0.86	0.55	0.67	11
international conference on image processing	0.50	0.80	0.62	5
international conference on management of data	0.60	0.60	0.60	5
international conference on robotics and automation	0.67	1.00	0.80	4
Accuracy	0.44			

Table 10: Classification report for the publication venue prediction task. The reported metrics include precision, recall, and F1-score for each publication venue. The *support* column indicates the number of instances belonging to each class in the test set.

References

[1] PyKEEN Developers. Hyper-parameter optimization — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/reference/hpo.html`, n.d. Accessed: 2025-05-26.

[2] PyKEEN. Transh — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/api/pykeen.models.TransH.html`, n.d. Accessed: 2025-05-27.

[3] PyKEEN. Rotate — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/api/pykeen.models.RotatE.html`, n.d. Accessed: 2025-05-27.

[4] PyKEEN. Complex — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/api/pykeen.models.ComplEx.html`, n.d. Accessed: 2025-05-27.

[5] PyKEEN. Convkb — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/api/pykeen.models.ConvKB.html`, n.d. Accessed: 2025-05-27.

[6] PyKEEN Developers. Understanding the evaluation — pykeen 1.11.1 documentation. `https://pykeen.readthedocs.io/en/stable/tutorial/understanding_evaluation.html`, 2023. Accessed: 2025-06-01.

[7] Scikit learn Developers. sklearn.neural_network.mlpclassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`, 2023. Accessed: 2025-06-12.