UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

FIB

SDM Lab2: Knowledge Graphs

Master in Data Science

Semantic Data Management
Facultat d'Informàtica de Barcelona, UPC

Authors: RONG XING, SILVIA FERRER VAZQUEZ
Professor: OSCAR ROMERO

June 15, 2025

Contents

1 Part B - Ontology Implementation

We implemented the ontology for the scientific publication domain using RDF Schema (RDFS) via the `rdflib` Python library. The TBox (terminological box) defines the conceptual structure of the domain, including classes, subclass hierarchies, and object properties representing relationships between entities.

Two separate namespaces were used:

- `http://research.publications.com/ontology#` for schema elements (classes and properties)
- `http://research.publications.com/instance#` for individuals (ABox)

Figure 1: Diagram of the TBox model for the research publication domain

We defined core classes such as `Paper`, `Author`, `Reviewer`, `Review`, `Conference`, `Workshop`, `Journal`, `Edition`, `Volume`, `Proceedings`, `Keyword`, and `City`. To better reflect the hierarchy and allow reuse, we introduced superclass `Event` for

Conference and Workshop, and superclass Organization for University and Company.

We defined several object properties to capture the relationships between entities. Each Paper is connected to one or more Author instances using the property research:writtenBy. A designated research:correspondingAuthor is specified as a subproperty of writtenBy, capturing the role of the primary contact author. Papers are linked to publication venues depending on type: they are connected to a Volume using research:publishedInJournal if published in a journal, or to Proceedings via research:publishedInProceedings if accepted to a conference or workshop.

Topical content is modeled with the research:hasKeyword property, which attaches one or more Keyword instances to a paper. Citation relationships are expressed through research:cites, allowing a paper to reference others in the corpus.

Peer review processes are captured explicitly using the Review class. A review instance connects a Reviewer to a target paper using research:reviewOf, and the actual authorship of the review is represented by research:writtenByReviewer. Assignment of reviewers to papers is handled by research:assignedTo, and although this relation is not automatically enforced, our data processing ensures that assigned reviewers do not overlap with the original authors, adhering to ethical review constraints.

The main object properties implemented are summarized as:

- writtenBy, with correspondingAuthor as a subproperty
- publishedInJournal, publishedInProceedings
- hasKeyword, cites
- assignedTo, reviewOf, writtenByReviewer

Figure 2: Class hierarchy in GraphDB

1.1 B.2 - ABox Population

The ABox was populated using a script that reads multiple preprocessed CSV files from Lab 1, including papers, authors, reviews, keywords, venues, and affiliations. These datasets were transformed into RDF triples using the `rdflib` library, conforming to the TBox schema described in the previous section. Instance URIs were constructed by combining class names and identifiers under the namespace `http://research.publications.com/instance#`.

We incorporated RDFS-based inference by asserting subclass relations for entities such as `Reviewer` (as a subclass of `Author`), and `University`/`Company` (as subclasses of `Organization`). This allowed us to derive implicit types at query time, e.g., recognizing all reviewers as authors.

A total of **8,806** RDF triples were generated in our knowledge base. The following table summarizes instance counts by class (`*Note: The reported statistics only cover selected core classes and properties. The total triple count (8806) includes additional metadata, datatype properties (e.g., names,`

titles, years), and the TBox schema itself):

- **Papers:** 100
- **Authors:** 100
- **Reviewers:** 0
- **Conferences:** 51
- **Workshops:** 0
- **Journals:** 49
- **Editions:** 273
- **Proceedings:** 273
- **Volumes:** 288
- **Reviews:** 208
- **Cities:** 9
- **Keywords:** 14
- **Universities:** 76
- **Companies:** 71

Property usage:

- `writtenBy`: 312 triples
- `hasKeyword`: 311 triples
- `cites`: 1,285 triples
- `publishedInProceedings`: 0 triples
- `publishedInJournal`: 476 triples
- `reviewOf`: 208 triples
- `affiliatedTo`: 147 triples

1.2 B.3 - SPARQL Querying and Reasoning

We designed two SPARQL queries to explore the reasoning potential of our ontology and derive insights.

Query 1: Identify Influential Authors in the Network

This query ranks authors by an influence score that combines: number of papers written, co-authorship connections, received citations, reviews performed (inferred via `Reviewer` as a subclass of `Author`), and topical diversity measured via keyword associations.

The use of `rdfs:subClassOf` inference allows the system to treat `Reviewer` individuals as `Author` instances, enriching the author profile automatically during query execution.

Query 1:

```
PREFIX research: <http://research.publications.com/ontology#>
PREFIX instance: <http://research.publications.com/instance#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?author ?authorName
       (COUNT(DISTINCT ?paper) as ?paperCount)
       (COUNT(DISTINCT ?coAuthor) as ?collaboratorCount)
       (COUNT(DISTINCT ?citingPaper) as ?citationCount)
       (COUNT(DISTINCT ?reviewedPaper) as ?reviewCount)
       (COUNT(DISTINCT ?keyword) as ?topicDiversity)
       (?paperCount + ?collaboratorCount * 2 + ?citationCount * 3
   + ?reviewCount * 2 + ?topicDiversity as ?influenceScore)
WHERE {
    ?author a research:Author ;
            research:hasName ?authorName .
    ?paper research:writtenBy ?author .
    ?paper research:writtenBy ?coAuthor .
    FILTER(?coAuthor != ?author)
    OPTIONAL { ?citingPaper research:cites ?paper . }
    OPTIONAL { ?paper research:hasKeyword ?keyword . }
    OPTIONAL {
        ?reviewer rdfs:subClassOf research:Reviewer .
        ?reviewer research:hasName ?authorName .
        ?review research:writtenByReviewer ?reviewer ;
                research:reviewOf ?reviewedPaper .
    }
}
GROUP BY ?author ?authorName
HAVING (?paperCount > 2)
ORDER BY DESC(?influenceScore)
LIMIT 20
```

Listing 1: Query 1 – Influential Author Ranking

Top 5 influential authors:

Name	Papers	CoAuthors	Citations	Reviews	Topics	Score
Палишева Наталья Витальевна	8	25	75	0	11	294
Sofija Pavković-Lučić	6	19	69	0	11	262
A. Buzzati-Traverso	6	15	67	0	11	248
M.N.M. Naleef	6	19	61	0	14	241
M. Winter	8	24	54	0	12	230

Query 2: Discover Cross-Domain Venue Quality and Trends

This query evaluates venues (journals, conferences, and workshops) using citation-based and topical metrics. It combines:

- Property paths (to retrieve editions, volumes, and papers)
- Aggregations (for average citation and topic diversity)
- RDFS-based type inference
- Nested subqueries to extract top keywords

Query 2:

```
PREFIX research: <http://research.publications.com/ontology#>
PREFIX instance: <http://research.publications.com/instance#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?venue ?venueName ?venueType
       (COUNT(DISTINCT ?paper) as ?totalPapers)
       (COUNT(DISTINCT ?citedPaper) as ?citedPapers)
       (AVG(?citationPerPaper) as ?avgCitations)
       (COUNT(DISTINCT ?keyword) as ?topicCoverage)
       (GROUP_CONCAT(DISTINCT ?topKeyword; SEPARATOR=", ") as ?
   topKeywords)
       (?citedPapers / ?totalPapers * 100 as ?citationRate)
WHERE {
    {
        ?venue a research:Conference ;
               research:hasName ?venueName .
        BIND("Conference" as ?venueType)
        ?venue research:hasEdition ?edition .
        ?proceedings research:includesPaper ?paper .
```

```
    } UNION {
        ?venue a research:Workshop ;
               research:hasName ?venueName .
        BIND("Workshop" as ?venueType)
        ?venue research:hasEdition ?edition .
        ?proceedings research:includesPaper ?paper .
    } UNION {
        ?venue a research:Journal ;
               research:hasName ?venueName .
        BIND("Journal" as ?venueType)
        ?venue research:hasVolume ?volume .
        ?paper research:publishedInJournal ?volume .
    }
    OPTIONAL {
        SELECT ?paper (COUNT(?citingPaper) as ?citationPerPaper)
        WHERE { ?citingPaper research:cites ?paper . }
        GROUP BY ?paper
    }
    OPTIONAL {
        ?anyCitingPaper research:cites ?paper .
        BIND(?paper as ?citedPaper)
    }
    OPTIONAL {
        ?paper research:hasKeyword ?keyword .
        ?keyword research:hasName ?keywordName .
        {
            SELECT ?keyword (COUNT(?p) as ?keywordFreq)
            WHERE { ?p research:hasKeyword ?keyword . }
            GROUP BY ?keyword
            HAVING (?keywordFreq > 3)
        }
        BIND(?keywordName as ?topKeyword)
    }
}
GROUP BY ?venue ?venueName ?venueType
HAVING (?totalPapers > 5)
ORDER BY DESC(?citationRate) DESC(?avgCitations)
```

Listing 2: Query 2 – Venue Citation and Topic Metrics

Most cited venues:

Venue	Type	Papers	Cited	Avg Cit.	Topics	Top Keywords
The Korea Journal of Buddhist Professors	Journal	6	6	20.63	12	AI, NLP, ML, CV, RL, IoT
TERAJU	Journal	11	11	19.42	13	robotics, ML, DL, CV, RL, IoT, cloud, big data, NN, pattern recognition, AI

2 Part C - Knowledge Graph Embeddings

2.1 C.1 - Importing the data

Data Selection for KGE Training

We constructed our knowledge graph embeddings dataset from the ABox of the domain ontology file `research_ontology.ttl`. The following preprocessing steps were applied:

1. **Triple Extraction.** We parsed the TTL file with RDFlib and extracted all RDF triples (s, p, o). Any triple whose object was a literal longer than 200 characters was discarded to avoid high-dimensional lexical noise:

$$\text{keep if } |o| \leq 200 \quad (\text{MAX\_LIT\_LEN} = 200).$$

2. **Relation Filtering.** We counted occurrences of each predicate and marked those with fewer than 3 instances as "rare" (`MIN_COUNT`=3). This ensures sufficient samples per relation for stratification.

3. **Stratified Splitting.** Taking into account that we have a large dataset, we decided to, using scikit-learn's `train_test_split` with `random_state=42`, perform a very common two-stage split:

   - 80% of triples for training, stratified by predicate (rare relations grouped under "rare").
   - Remaining 20% split evenly into validation and test (10% each), again stratified.

The result was approximately:

$$|\mathcal{T}_{\text{train}}| \approx 80\,000, \quad |\mathcal{T}_{\text{valid}}| \approx 10\,000, \quad |\mathcal{T}_{\text{test}}| \approx 10\,000.$$

These splits were used for all KGE model training (Section C.3) and downstream tasks (Section C.4).

2.2 C.2 – Getting Familiar with KGEs

2.2.1 KGE Models Background

In this section, we introduce and compare six popular knowledge graph embedding (KGE) models using our project triples dataset $\mathcal{T}$, partitioned as in Section C.1. Each triple $(h, r, t) \in \mathcal{T}$ consists of a head entity h, a relation r, and a tail entity t. For training, we use $(h, r, t) \in \mathcal{T}_{\text{train}}$. Below, we first present TransE with project-specific instantiations and then illustrate how more advanced models address its limitations.

1. Translating Embeddings (TransE)

TransE represents entities $h, t \in R^k$ and relations $r \in R^k$ in the same k-dimensional space. Given a training triple $(h, r, t) \in \mathcal{T}_{\text{train}}$, the model enforces:

$$\mathbf{h} + \mathbf{r} \approx \mathbf{t},$$

and scores each triple by:

$$f_{\text{TransE}}(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p$$

Project Point of View. In our domain, important relations include `cites` (paper-to-paper) and `writtenBy` (paper-to-author). We denote embeddings:

$$e_{\text{cited}} = e_{\text{paper}} + w_{\text{cites}} \tag{1}$$

$$e_{\text{author}} = e_{\text{cited}} - w_{\text{writtenBy}} \tag{2}$$

Here, e_{paper} is the embedding of a cited paper, w_{cites} the relation vector, and so on.

Drawbacks. TransE struggles with modeling *one-to-many*, *many-to-one*, and *many-to-many* relations, as it enforces the same translation vector $\mathbf{r}$ for all triples sharing the same relation. For example, consider the `writes` relation with triples:

$$\{(A_1, \text{writes}, P_1),\ (A_2, \text{writes}, P_1),\ (A_1, \text{writes}, P_2)\}.$$

The translation equations

$$e_{A_1} + w_{\text{writes}} \approx e_{P_1}, \quad e_{A_2} + w_{\text{writes}} \approx e_{P_1}, \quad e_{A_1} + w_{\text{writes}} \approx e_{P_2}$$

imply $e_{A_1} \approx e_{A_2}$ and $e_{P_1} \approx e_{P_2}$, leading to the degenerate solution:

$$e_{A_1} = e_{A_2} = v,\ w_{\text{writes}} = 0,\ e_{P_1} = e_{P_2} = v$$

2. Hyperplane Projection (TransH) TransH projects entity embeddings onto a relation-specific hyperplane. For (h, r, t):

$$\mathbf{e}_\perp = \mathbf{e} - (\mathbf{n}_r^\top \mathbf{e})\, \mathbf{n}_r,$$

$$f_{\text{TransH}}(h, r, t) = -\|\mathbf{e}_{h,\perp} + \mathbf{w}_r - \mathbf{e}_{t,\perp}\|_2^2$$

In our writes-example, A_1 and A_2 project differently onto the hyperplane for `writes`, preventing collapse.

Drawbacks. While better at handling complex relations, TransH introduces extra parameters per relation and may still conflate entities that behave differently under multiple relations.

3. Relation-specific Space (TransR) TransR uses per-relation matrices $\mathbf{M}_r$ to project entities into a relation space:

$$\mathbf{h}_r = \mathbf{M}_r \mathbf{h}, \quad f_{\text{TransR}}(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2$$

This decoupling better models relations like `cites` vs. `writtenBy` with different subspaces.

Drawbacks. The per-relation projection matrices dramatically increase the number of parameters, leading to higher computational cost and risk of overfitting on smaller datasets.

4. Rotational Embeddings (RotatE) RotatE embeds entities as complex vectors on the unit circle and relations as rotations:

$$f_{\text{RotatE}}(h, r, t) = -\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|_2,$$

modeling symmetry iff the rotation angle $\theta_r \in \{0, \pi\}$, so symmetric relations like `coauthorOf` can be captured.

Drawbacks. RotatE elegantly models symmetry, antisymmetry, inversion, and composition, but training can be sensitive to initialization and negative sampling strategy.

5. Bilinear Model (DistMult) DistMult scores triples via a diagonal bilinear form:

$$f_{\text{DistMult}}(h, r, t) = \sum_i h_i \, r_i \, t_i,$$

nativelysymmetric, ideal for `coauthorOf` but cannot handle asymmetry in `cites`.

Drawbacks. Because the score is symmetric in h and t, DistMult cannot distinguish between (h, r, t) and (t, r, h), limiting its ability to model asymmetric relations.

6. Complex Bilinear Model (ComplEx) ComplEx uses complex embeddings:

$$f_{\text{ComplEx}}(h, r, t) = \Re\big(\sum_i h_i \, r_i \, \overline{t_i}\big),$$

enabling asymmetry and composition with doubled embedding dimensions.

Drawbacks. ComplEx handles asymmetric relations well but doubles memory usage (storing real and imaginary parts) and may require careful regularization to prevent overfitting.

Summary of Model Trade-offs

- **TransE:** simple translation, fails on multi-valued relations (e.g. `writes`).
- **TransH/TransR:** add projection, alleviating collapse by relation-specific spaces.
- **RotatE:** rotation handles multi-pattern symmetry/antisymmetry.
- **DistMult/ComplEx:** bilinear forms; DistMult for symmetry, ComplEx for asymmetry.

This enriched discussion grounds each model in our author–paper–citation scenario and sets up the comparative experiments in Section C.3.

2.2.2 TransE Implementation

We implement TransE in PyTorch to answer two concrete tasks: (1) predict the embedding of a paper cited by a given paper, and (2) predict the embedding of that cited paper's author. Here we use the specific URIs:

- `PAPER_URI` = `"http://research.publications.com/instance#Paper_9169568"`
- `CITES_URI` = `"http://research.publications.com/ontology#cites"`
- `WRITTENBY_URI` = `"http://research.publications.com/ontology#writtenBy"`

1. Cited-paper embedding
After training, we have learned entity embeddings $E \in R^{|\mathcal{E}|\times d}$ and relation embeddings $W \in R^{|\mathcal{R}|\times d}$. Given the query paper URI `PAPER_URI`, its index is

$$i = \text{ent2id}[\texttt{PAPER\_URI}],$$

and the `cites` relation index is

$$j = \text{rel2id}[\texttt{CITES\_URI}].$$

The predicted embedding of the cited paper is

$$\hat{e}_{\text{cited}} = E[i] + W[j].$$

2. Author embedding
To invert the `writtenBy` relation, we subtract its learned vector. Let

$$k = \text{rel2id}[\texttt{WRITTENBY\_URI}].$$

Then

$$\hat{e}_{\text{author}} = \hat{e}_{\text{cited}} - W[k].$$

To map $\hat{e}_{\text{author}}$ back to a real author, we:

1. Collect all indices $A = \{\, i : \text{entities}[i] \text{ contains "Author"}\}$.
2. Compute distances $d_i = \|E[i] - \hat{e}_{\text{author}}\|_2$ for $i \in A$.
3. Select $i^* = \arg\min_{i\in A} d_i$ and report entities$[i^*]$ as the closest author.

Results Running this procedure produced:

- Predicted embedding for the cited paper (dimension 50):

```
[-0.45376277  0.93226314  0.69966507  0.58718026 -0.19521074  0.3763272
  0.8304146  -0.95101005  0.7886403  -0.79172385 -0.1198615   0.4994396
 -0.22046277  0.32836032 -0.4052437   0.35319167  0.16296911 -0.81582433
 -0.60866517  0.4206226  -0.61726785  0.25676614  0.7304939   0.21548039
 -0.4355697  -0.57070416  0.34556055  0.02106497  0.71703315 -0.3776857
  0.7721556  -0.05974499  0.554941    0.6176584   0.8568245  -0.40208638
  0.45329976  0.3212998   0.24485582  0.435603   -0.50821173 -0.746611
  0.32302928  0.9695684   0.5685413  -0.67313266  0.05766132 -0.5254693
 -0.41733003 -0.68235624]
```

  Closest paper in the KG:
  `http://research.publications.com/instance#Paper_120690647`
  Distance to the predicted vector: 1.5778

- Closest author match:
  `http://research.publications.com/instance#Author_2230924467`
  Distance to predicted author vector: 5.5228

2.2.3 Improving TransE

As observed, TransE's simplicity brings interpretability but struggles with One-to-Many, Many-to-One, Many-to-Many and symmetric relations. We now discuss these issues and outline models designed to alleviate them.

1. TransE on One-to-Many / Many-to-One
Given the small KG:

$(\text{Author}_1, \text{writes}, \text{Paper}_1), \quad (\text{Author}_2, \text{writes}, \text{Paper}_1), \quad (\text{Author}_1, \text{writes}, \text{Paper}_2),$

TransE enforces

$$\mathbf{e}_{\text{Author}_i} + \mathbf{w}_{\text{writes}} \approx \mathbf{e}_{\text{Paper}_j}.$$

An *optimal* (zero-loss) solution is

$$\mathbf{e}_{\text{Paper}_1} = \mathbf{e}_{\text{Author}_1} + \mathbf{w}, \quad \mathbf{e}_{\text{Paper}_1} = \mathbf{e}_{\text{Author}_2} + \mathbf{w},$$

so necessarily

$$\mathbf{e}_{\text{Author}_1} = \mathbf{e}_{\text{Author}_2}.$$

Thus all citing authors collapse to the same point, destroying distinction. Similarly, the *writes* tail Paper_2 forces Paper_2 to share same author offset.

2. A model to mitigate One-to-Many: TransH
TransH (Wang et al. 2014) projects entities onto a relation-specific hyperplane before translation:

$$\mathbf{e}_\parallel = \mathbf{e} - \mathbf{w}_\perp^\top \mathbf{e}\, \mathbf{w}_\perp, \quad \mathbf{e}_\parallel + \mathbf{d}_r \approx \mathbf{t}_\parallel$$

By allowing each relation r its own normal $\mathbf{w}_\perp$ and translation $\mathbf{d}_r$, TransH can assign different "writes"-projected embeddings to Author_1 and Author_2, avoiding the collapse.

3. Symmetry in TransE
For a symmetric relation s (e.g. collaboratesWith), one would expect

$$\text{score}(\text{Author}_1, s, \text{Author}_2) \;=\; \|\mathbf{e}_1 + \mathbf{w}_s - \mathbf{e}_2\| \;\stackrel{?}{=}\; \|\mathbf{e}_2 + \mathbf{w}_s - \mathbf{e}_1\| \;=\; \text{score}(\text{Author}_2, s, \text{Author}_1)$$

In general these are *not* equal in TransE, since $\|\mathbf{a} + \mathbf{w} - \mathbf{b}\| \neq \|\mathbf{b} + \mathbf{w} - \mathbf{a}\|$ unless $\mathbf{w} = \mathbf{0}$ or $\mathbf{a} = \mathbf{b}$.

4. Why TransE fails symmetry (2D sketch)

TransE Translation in 2D

$\mathbf{e}_1$ $\qquad$ $\mathbf{e}_2$

$$\longrightarrow\ \mathbf{e}_1 + \mathbf{w} \;\approx\; \mathbf{e}_2$$

$$\longrightarrow\ \mathbf{e}_2 + \mathbf{w} \;\approx\; \mathbf{e}_1 \text{ (fails)}$$

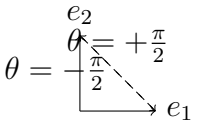
Unless $\mathbf{w} = \mathbf{0}$ (degenerate) or $\mathbf{e}_1 = \mathbf{e}_2$ (trivial), the two arrows cannot both land exactly on the other point.

5. RotatE and symmetry (2D sketch)
RotatE (Sun et al. 2019) represents entities as complex vectors and relations as element-wise rotations:

$$\mathbf{e}_1 \mathbf{r} \;\approx\; \mathbf{e}_2, \quad \left|\mathbf{r}_k\right| = 1 \;\forall k.$$

In 2D this is a rotation by angle θ. Symmetry holds *iff* $\theta = 0$ or π:

$$\mathbf{r} = e^{\mathrm{i}\theta}, \quad e^{\mathrm{i}\theta} \cdot e_1 = e_2 \iff e^{\mathrm{i}\theta} \cdot e_2 = e_1$$



6. A model with built-in symmetry: SimplE
SimplE (Kazemi & Poole, 2018) assigns each entity two vectors (as head and tail) and scores

$$\text{score}(h, r, t) = \tfrac{1}{2}\Big(\langle \mathbf{h}_{\text{head}}, \mathbf{r}, \mathbf{t}_{\text{tail}} \rangle + \langle \mathbf{t}_{\text{head}}, \mathbf{r}^{-1}, \mathbf{h}_{\text{tail}} \rangle\Big)$$

Because it considers both directions with r and its inverse r^{-1}, any symmetric r naturally yields equal scores for (h, r, t) and (t, r, h) without further constraints.

2.3 C.3 - Training KGEs

Models & Hyperparameters

- Models tested: TransE, DistMult, ComplEx
- Embedding dimensions: 50 and 100
- Negatives per positive: 5 and 20

Comparison Conclusions

Table 1: Validation results for different KGE models and hyperparameters

Model	Dim	#Neg	MRR	Hits@10
TransE	50	5	0.0557	0.1552
TransE	50	20	0.0472	0.1267
TransE	100	5	0.0400	0.1240
TransE	100	20	0.0467	0.1233
DistMult	50	5	0.0979	0.2533
DistMult	50	20	0.0939	0.2540
DistMult	100	5	0.1041	0.2725
DistMult	100	20	0.0946	0.2653
ComplEx	50	5	0.1006	0.2692
ComplEx	50	20	0.0933	0.2573
ComplEx	100	5	**0.1074**	**0.2858**
ComplEx	100	20	0.1001	0.2719

Based on the validation experiments summarized in Table 1, we observe:

- **Model performance ranking:** ComplEx outperforms DistMult, which in turn outperforms TransE across almost all settings.

- **Embedding dimension:** Increasing from 50 to 100 dimensions yields clear gains in MRR and Hits@10 for both DistMult and ComplEx, while TransE shows no consistent benefit.
- **Negative sampling:** Using 5 negatives per positive example gives slightly better or equal performance compared to 20 negatives for both DistMult and ComplEx, suggesting limited returns from larger negative budgets.
- **Best configuration:** ComplEx with embedding dimension 100 and 5 negative samples achieves the highest validation scores (MRR = 0.1074, Hits@10 = 0.2858).

We selected **ComplEx** (dim=100, #Neg=5) for the exploitation phase. To further validate its generalization, we evaluated it on the test set and obtained:

$$\text{MRR}_{\text{test}} = 0.1130, \quad \text{Hits@10}_{\text{test}} = 0.3042.$$

This improvement over validation confirms the robustness of our choice.

2.4 C.4 - Exploiting KGEs

In this exercise we demonstrate a supervised application of the learned ComplEx embeddings (dim=100, #Neg=5) by training a binary classifier to distinguish valid triples from corrupted (negative) ones. Our goal is to show that the embedding scores produced by the KGE model can serve as effective features for downstream decision tasks.

Data Preparation

We use the held-out test set of true triples X_{test} as positive examples. For each positive triple $(h, r, t) \in X_{\text{test}}$, we generate one negative example by corrupting the tail entity:

$$(h, r, t') \quad \text{with} \quad t' \sim \text{Uniform}(\mathcal{E}),\ t' \neq t.$$

This yields a balanced dataset of $2|X_{\text{test}}|$ examples. We repeat the same procedure on the validation split to train the classifier.

Feature Extraction

For each triple (h, r, t), we compute its ComplEx plausibility score:

$$s(h, r, t) = \text{model.predict}(\,[(h, r, t)]\,) \in R,$$

higher values indicating more plausible triples. We assemble a one-dimensional feature vector $[\,s(h, r, t)\,]$ for each example.

Classification Methodology

1. **Validation classifier training:** We fit a logistic regression model on the validation examples' scores and labels.
2. **Test evaluation:** We apply the trained classifier to the test examples, computing predicted probabilities and labels.
3. **Metrics:** We report area under the ROC curve (AUC) and classification accuracy.

Results

- **AUC** = 0.8863
- **Accuracy** = 0.8397

These results demonstrate that a simple linear model on top of raw ComplEx scores can successfully separate valid from invalid triples with nearly 90% discrimination power.

Discussion

The high AUC confirms that the ComplEx embeddings capture the necessary signal for triple plausibility. This supervised evaluation exemplifies how KGE scores can be harnessed for downstream classification tasks. Further improvements could include:

- Incorporating separate head- and tail-side scores as two features.
- Using more challenging "local" negative sampling (corrupt entities within the same relation).
- Employing non-linear classifiers (e.g. small MLP) on top of multiple scoring features.