Academic Knowledge Graphs

Walter J. Troiani Iván Martinez
Facultat d'Informàtica de Barcelona (FIB), UPC, 08034 Barcelona, Spain

I. Introduction

This report presents a comprehensive approach to constructing and analyzing academic knowledge graphs, along with generating effective embedding representations to support a variety of downstream classical machine learning tasks such as classification, clustering, and link prediction. The system models complex scholarly ecosystems by representing entities such as publications, authors, peer reviews, and citation networks, capturing their intricate interrelations. To build this knowledge graph, a robust data pipeline is designed encompassing data extraction, transformation, and loading (ETL) processes that integrate heterogeneous data sources including both real-world bibliographic datasets and synthetically generated records to ensure coverage and robustness.

The ontology design includes a well-structured Terminological Box (TBOX), defining the schema and semantic constraints, alongside an Assertional Box (ABOX) that encodes individual entity instances and their relationships. This dual-layer modeling strategy enables expressive and semantically rich representations, facilitating advanced querying and reasoning capabilities within GraphDB [1]. Embeddings are learned over this structured data, leveraging state-of-the-art knowledge graph embedding techniques to translate symbolic graph information into dense vector spaces suitable for machine learning applications [2], [3].

This integrated framework not only supports scalable knowledge graph construction but also fosters improved interpretability and predictive power in academic analytics tasks such as paper and author recommendation as demonstrated in recent literature [4], [5]. Through our end to end process that crystalizes in a real-world application, we aim to demonstrate the expressive power of knowledge graph embeddings and the utility of the knowledge graphs data model.

II. Ontology

A. TBOX

The first step in the ontology creation process was to define the **TBOX**, which specifies the schema or conceptual vocabulary of the domain. This involves declaring the core classes, their hierarchical relationships, and the properties that link instances of these classes. Our TBOX was constructed using the RDF Schema (RDFS) standard, following the design conventions introduced in the course.

A graphical representation of our interpretation of the TBOX is shown in the next page.

1) Declared Classes

The ontology defines the following classes to capture the core domain concepts:

- **Author** – Represents individuals who write papers.
- **Reviewer** – Represents individuals who perform peer reviews. This is a subclass of `Author`.
- **Paper** – Represents scientific articles and publications.
- **Conference**, **Workshop** – Represent types of events where research is presented. Both are modeled as subclasses of `Event`.
- **Journal** – Represents academic journals that publish papers.
- **Edition** – Represents a specific occurrence of a conference or workshop.
- **Proceedings** – A collection of papers presented at an edition.
- **Volume** – A volume of a journal.
- **Review** – Written feedback or evaluation of a paper.
- **City** – A location where an edition takes place.
- **Topic** – Represents the subject area(s) covered by a paper.
- **Event** – Generalization for both conferences and workshops.

2) Declared Properties

We defined a set of properties with clearly specified domains and ranges. These are grouped by conceptual area for clarity:

a) Paper-related Properties

- `hasAuthor` (Paper $\rightarrow$ Author): Links a paper to its authors.
- `hasCorrespondingAuthor` (Paper $\rightarrow$ Author): A subproperty of `hasAuthor`, identifying the main contact author.
- `hasAbstract` (Paper $\rightarrow$ `xsd:string`): Stores the abstract of the paper.
- `aboutTopic` (Paper $\rightarrow$ Topic): Specifies what the paper is about.
- `hasReview` (Paper $\rightarrow$ Review): Links a paper to its associated reviews.
- `cites` (Paper $\rightarrow$ Paper): Captures citation links between papers.

b) Review-related Properties

- `hasContent` (Review $\rightarrow$ `xsd:string`): Stores the textual content of a review.
- `performedBy` (Review $\rightarrow$ Reviewer): Associates a review with the reviewer who wrote it.

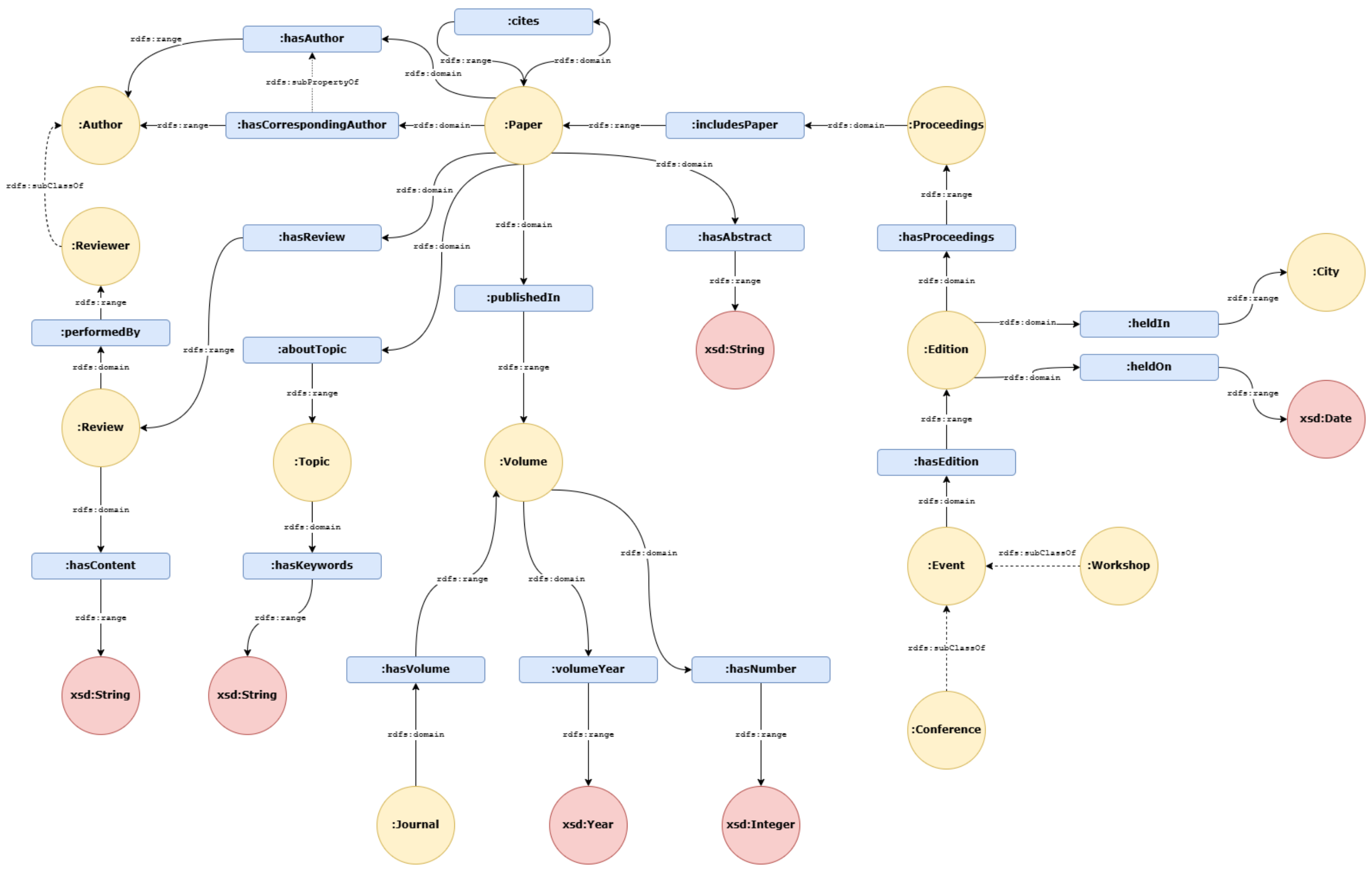


Fig. 1: TBOX schema for the research domain knowledge graph

c) Publication-related Properties

- `publishedIn` (Paper → Volume): Indicates what volume a paper is published in.
- `hasVolume` (Journal → Volume): Associates journals with their included volumes.
- `volumeYear` (Volume → `xsd:gYear`): The year of publication for the volume.
- `hasNumber` (Volume → `xsd:integer`): Volume number identifier.
- `includesPaper` (Proceedings → Paper): Papers contained in a proceedings.
- `hasProceedings` (Edition → Proceedings): A proceedings collection for a given edition.

d) Event-related Properties

- `hasEdition` (Event → Edition): Indicates that an event (conference or workshop) has one or more editions.
- `heldIn` (Edition → City): The city where the edition takes place (in other words, the venue).
- `heldOn` (Edition → `xsd:date`): The date when the edition was held.

e) Topic-related Properties

- `hasKeywords` (Topic → `xsd:string`): A literal string containing keywords that describe the topic.

3) Unmodeled Constraints and Limitations

Due to the expressiveness limitations of RDFS, some domain-specific constraints could not be enforced declaratively in the TBOX, though we acknowledged and documented them as follows:

- A paper should have **exactly one corresponding author**. Enforcing functional properties requires OWL.
- A paper is expected to have **exactly three reviewers** associated with it, which would require `owl:cardinality` constraints.
- An author should not be allowed to **review their own paper**. This requires SWRL rules or SHACL validation.
- While conferences and workshops are both treated as events, generalizing editions over both requires reasoning support such as `owl:UnionOf`.
- Keywords could have been modeled as individual resources (in other words, having class `Keyword` or similar, and having each distinct keyword be an instance of said class). Instead, they are modeled as literal strings for simplicity.

These constraints should be validated at the application level, or with ontology validation tools (e.g., SHACL or OWL reasoning), which go beyond the scope of RDFS.

B. ABOX

The construction of the ABOX required the transformation of non-semantic CSV data into semantically rich RDF triples, in accordance with the TBOX schema. The raw data originated from an earlier project and consisted of multiple CSV files that modeled a property graph using separate node and edge files. However, the original dataset lacked several entities and relationships required by the TBOX. To address this, we generated additional synthetic data through a Python script to ensure completeness and compliance with the ontology.

1) Synthetic Data Generation

Part of the ABOX construction effort was devoted to augmenting the initial dataset with synthetically generated nodes and edges. This was necessary to populate entities such as *Proceedings*, *Workshops*, *Reviews*, and their relationships, which were either absent or only partially present in the raw data. A Python script was developed to automate this process, utilizing existing data distributions to preserve plausibility and coherence.

The script first generates a one-to-one mapping between each existing *Edition* and a new *Proceedings* node. These are then linked accordingly using the relationships `edition_has_proceedings` and `proceedings_includes_paper`. Next, 30 new editions are synthetically created to support newly defined workshop events. These editions inherit plausible values (such as year and location) by sampling from the original *Edition* nodes.

Subsequently, 50 synthetic *Workshop* nodes are generated, each randomly assigned a theme sampled from existing *Conference* data. A matching relationship between workshops and the newly added editions is created to preserve logical consistency within the knowledge graph.

In order to introduce review data, each unique paper that has been reviewed is linked to a synthetic *Review* node using the `paper_has_review` relation. For each *Review*, a short, semi-random but plausible review text is generated from a predefined pool of sample sentences to mimic realistic reviewer feedback.

The script ensures that all generated entities are uniquely identified and that the relationships conform strictly to the structure and semantics defined by the TBOX. A summary of the synthetic CSVs created is as follows:

- `nodes_proceedings.csv`, `edges_edition_has_proceedings.csv`, `edges_proceedings_includes_paper.csv`
- `nodes_workshop.csv`, `edges_edition_held_for_workshop.csv`
- `nodes_review.csv`, `edges_paper_has_review.csv`
- Updated `nodes_edition.csv` with 30 additional entries (synthetic editions for newly created workshop instances)

This synthetic extension enabled full coverage of all required ABOX components, making it possible to instantiate the ontology completely using the available and augmented data.

2) ABOX Triple Generation Strategy

URI Generation and Disambiguation

URIs for key entities such as **Authors**, **Conferences**, **Journals**, and **Papers** are derived from sanitized versions of their names or titles to improve human readability and ensure semantic clarity. A utility function `clean_string_for_uri()` handles this transformation by removing punctuation and normalizing whitespace, converting spaces to underscores and stripping special characters. To ensure uniqueness, a deduplication mechanism is employed: entities with the same cleaned name are assigned numeric suffixes (e.g., `John_Smith`, `John_Smith_1`, etc.). Paper URIs are generated from sanitized titles, while topic URIs are constructed by joining a paper's fields of study with the "_and_" delimiter. For example, a topic URI might appear as: `http://SDM.org/research/Medicine_and_Physics`. If a paper lacks specific fields of study, a default URI labeled `NoSpecifiedTopic` is used. For city URIs, the system extracts the first part of the location string (before any commas, as this part corresponds to the city), and cities sharing the same name are assigned identical URIs.

Topic and Keywords Modeling

Topics are dynamically instantiated based on the `fields_of_study` metadata for each paper. Instead of splitting keywords into individual literals, the entire keyword string from the dataset is attached to the corresponding Topic using the `:hasKeywords` predicate. This approach reflects the idea that keywords collectively describe a topic, rather than each acting as standalone descriptors. Consequently, multiple papers can associate different keyword strings with the same Topic resource.

Reviewers Modeling

Reviewers are modeled as a subclass of `Author`, aligning with the class hierarchy defined in the TBOX schema. Their identifiers are sourced from `edges_author_reviews_paper.csv` and resolved to names using `nodes_author.csv`. The same URI generation rules applied to authors are used here for consistency. Although reviewers could be inferred from their relationships, they are explicitly typed as `res:Reviewer` to preserve classification without requiring reasoning. This enables statements like `<Review> res:performedBy <Reviewer>` to be asserted directly.

Edition Modeling

Editions of conferences and workshops include a `heldOn` date. Since the source data only provides the year, dates are standardized to the format `YYYY-01-01` to satisfy the ontology's requirement for a valid `xsd:date` value.

Volume Modeling

When volume entries lack a numerical value for the `number` field, a default of `0` is assigned. This ensures that the `hasNumber` property is always present and remains type-safe, conforming to the `xsd:integer` datatype.

Edge Modeling

Relationship triples (edges) are created by parsing various CSV files and mapping subject and object identifiers to URIs based on entity-specific resolution strategies. Predicate directions follow the definitions specified in the TBOX. For example, paper URIs are resolved via a `paper_title_to_uri` mapping, while conference and journal identifiers from the data are converted to name-based URIs. This modeling framework supports a wide range of relationships: **authorship** via `hasAuthor` and `hasCorrespondingAuthor`; **reviews** via `hasReview` and `performedBy`; **publication** through `publishedIn` and `hasVolume`; **event organization** with `hasEdition`, `heldIn`, and `heldOn`; **proceedings management** via `hasProceedings` and `includesPaper`; and **citations** via `cites`.

3) Inference Regime and Explicit `rdf:type` Declarations

This work adopts the **RDFS entailment regime** to support schema-level reasoning, including subclass and subproperty inference, as well as automatic typing based on declared `rdfs:domain` and `rdfs:range`.

Accordingly, many `rdf:type` triples are omitted from the ABOX when they can be inferred. For instance, given a triple like `<paper123> :hasAuthor <author456>` and a TBOX definition of `:hasAuthor` with domain `:Paper` and range `:Author`, an RDFS reasoner can deduce the types of both entities without explicit assertions.

Explicit `rdf:type` declarations are only added when inference is not feasible. This includes entities such as `:Review`, `:Proceedings`, `:Volume`, `:Edition`, `:City`, and `:Topic`, which are either generated synthetically or derived from standalone CSV records lacking inferable properties. Reviewers are also explicitly typed as `res:Reviewer` to ensure classification, even though they inherit from `:Author`.

4) Knowledge Graph Summary and Statistics

To provide an overview of the constructed ABOX, we present a series of descriptive plots that summarize the instances and structure of the knowledge graph. These visualizations highlight the number of classes and properties, instance counts for different entity types, and usage frequency of main predicates, amongst others. Together, they offer insight into the scale and composition of the generated graph.

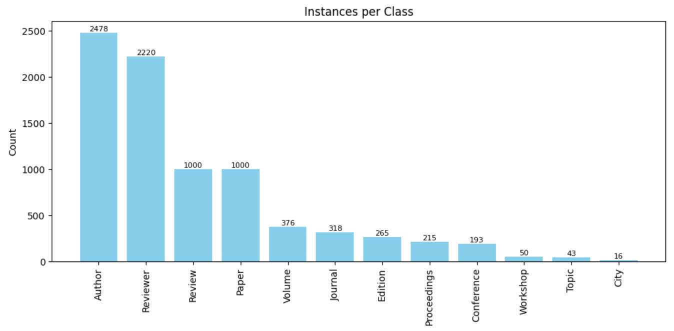


Fig. 2: Distribution of Instances Across Classes

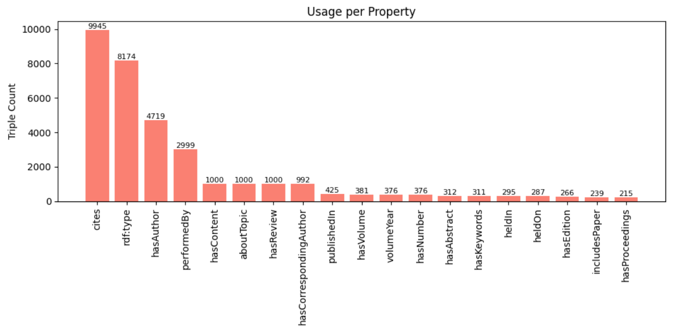


Fig. 3: Frequency of Property Usage in the ABOX

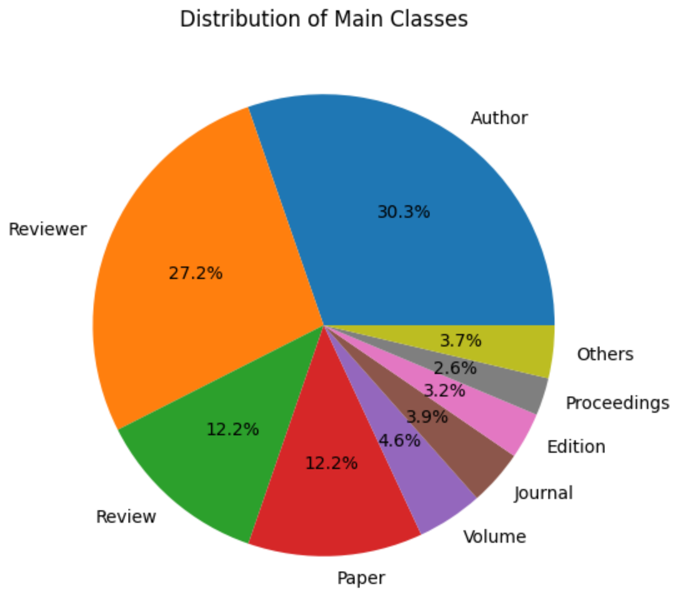


Fig. 4: Top 8 most prevalent classes

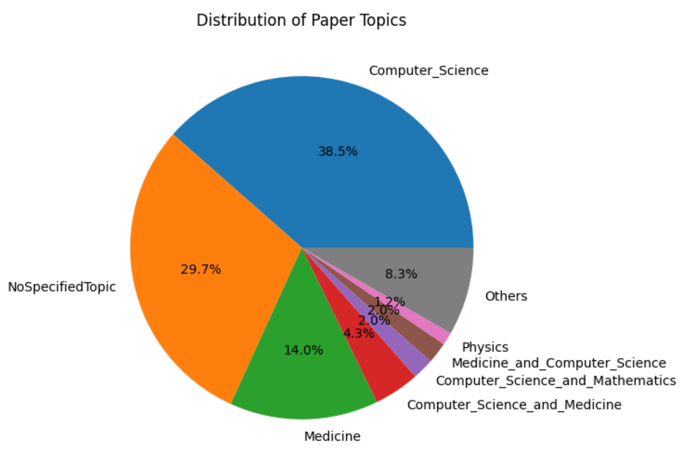


Fig. 5: Distribution of Topics Across All Papers

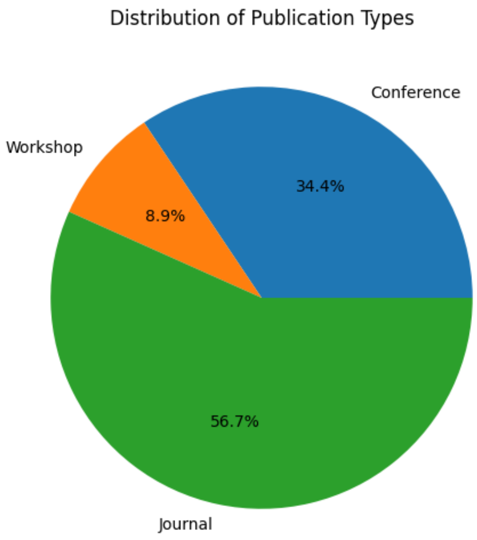


Fig. 6: Distribution of Publication Types

C. *Querying Our Ontology*

This section presents two SPARQL queries that showcase how explicit TBOX definitions and reasoning enhance data retrieval from our knowledge graph. Each query is accompanied by a brief explanation, the query code, and a visualization of part of the results.

Query 1: Retrieve Reviews by a Specific Reviewer

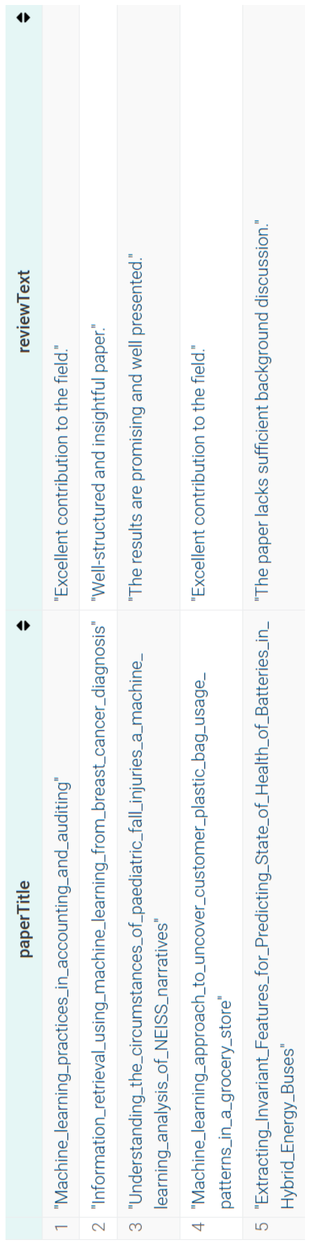
This query extracts all reviews performed by the reviewer `O_Stenzel` (used as an example), listing the titles of the reviewed papers along with the review content. It leverages the role-based modeling where `Reviewer` is a subclass of `Author` and shows how linked entities across multiple relations can be queried. The query depends on TBOX definitions for `res:performedBy` and `res:hasReview`, illustrating the benefit of semantic typing and inference.

```
PREFIX res: <http://SDM.org/research/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
    -ns#>

SELECT ?paperTitle ?reviewText
WHERE {
  ?review rdf:type res:Review ;
          res:performedBy res:O_Stenzel ;
          res:hasContent ?reviewText .

  ?paper res:hasReview ?review .

  BIND(REPLACE(STR(?paper), "http://SDM.org/research
    /", "") AS ?paperTitle)
}
LIMIT 100
```



	paperTitle	reviewText
1	"Machine_learning_practices_in_accounting_and_auditing"	"Excellent contribution to the field."
2	"Information_retrieval_using_machine_learning_from_breast_cancer_diagnosis"	"Well-structured and insightful paper."
3	"Understanding_the_circumstances_of_paediatric_fall_injuries_a_machine_learning_analysis_of_NEISS_narratives"	"The results are promising and well presented."
4	"Machine_learning_approach_to_uncover_customer_plastic_bag_usage_patterns_in_a_grocery_store"	"Excellent contribution to the field."
5	"Extracting_Invariant_Features_for_Predicting_State_of_Health_of_Batteries_in_Hybrid_Energy_Buses"	"The paper lacks sufficient background discussion."

Fig. 7: Results of Query 1

Query 2: Identify Top Topics and Aggregate Their Keywords

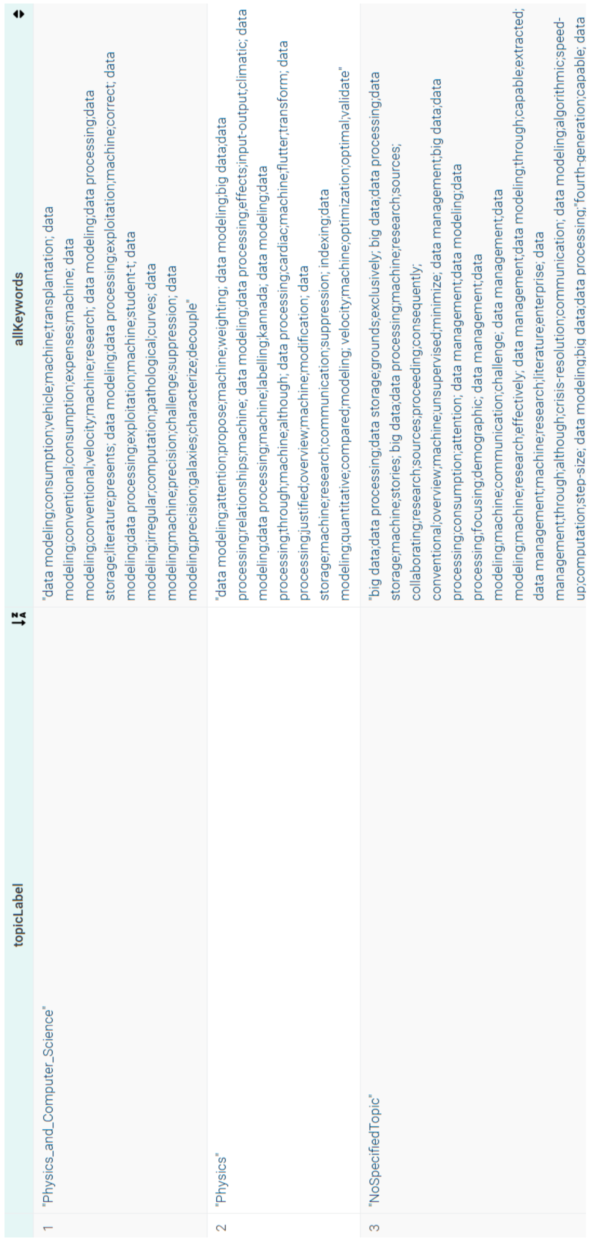
This query identifies the top research topics covered by papers in the knowledge graph, aggregating all keywords associated with each topic. It uses the `res:aboutTopic` and `res:hasKeywords` predicates, benefiting from the ontology's conceptual modeling of topics and their descriptive terms. Aggregation with `GROUP_CONCAT` illustrates how semantic data can be summarized and analyzed at a conceptual level rather than just raw literals.

```
PREFIX res: <http://SDM.org/research/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
    -ns#>

SELECT ?topicLabel (GROUP_CONCAT(DISTINCT ?keywords;
     separator="; ") AS ?allKeywords)
WHERE {
  ?paper rdf:type res:Paper ;
         res:aboutTopic ?topic .

  ?topic rdf:type res:Topic ;
         res:hasKeywords ?keywords .

  BIND(REPLACE(STR(?topic), "http://SDM.org/research
    /", "") AS ?topicLabel)
}
GROUP BY ?topicLabel
ORDER BY DESC(COUNT(?paper))
LIMIT 10
```



	topicLabel	allKeywords
1	"Physics_and_Computer_Science"	"data modeling;consumption;vehicle;machine;transplantation; data modeling;conventional;consumption;expenses;machine; data modeling;conventional;velocity;machine;research; data modeling;data processing;data storage;literature;presents; data modeling;data processing;exploitation;machine;correct; data modeling;data processing;exploitation;machine;student-t; data modeling;irregular;computation;pathological;curves; data modeling;machine;precision;challenge;suppression; data modeling;precision;galaxies;characterize;decouple"
2	"Physics"	"data modeling;attention;propose;machine;weighting; data modeling;big data;data processing;relationships;machine; data modeling;data processing;effects;input-output;climatic; data modeling;data processing;machine;labelling;kannada; data modeling;data processing;through;machine;although; data processing;cardiac;machine;flutter;transform; data processing;justified;overview;machine;modification; data storage;machine;research;communication;suppression; indexing;data modeling;quantitative;compared;modeling; velocity;machine;optimization;optimal;validate"
3	"NoSpecifiedTopic"	"big data;data processing;data storage;grounds;exclusively; big data;data processing;data storage;machine;stories; big data;data processing;machine;research;sources; collaborating;research;sources;proceeding;consequently; conventional;overview;machine;unsupervised;minimize; data management;big data;data processing;consumption;attention; data management;data modeling;data processing;focusing;demographic; data management;data modeling;machine;communication;challenge; data management;data modeling;machine;research;effectively; data management;data modeling;through;capable;extracted; data management;machine;research;literature;enterprise; data management;through;although;crisis-resolution;communication; data modeling;algorithmic;speed-up;computation;step-size; data modeling;big data;data processing;"fourth-generation;capable; data

Fig. 8: Results of Query 2

III. Embeddings

A. Data Extraction

The first step in order to train any embedding model is to carefully decide which information from our established knoweldge graph will be useful for this purpose. This data will be filtered using Python's RDFLib and the initial ABOX triples persisted in GraphDB.

1) **Publications:** The following list of relationships are picked in order to capture relationships with authors, topics, proceedings and volumes.
   - `:hasAuthor`
   - `:hasCorrespondingAuthor`
   - `:aboutTopic`

- `:IncludedInVolume` (Volumes)
- `:IncludesPaper` (Proceedings)

2) **Citations:** The `:cites` relationship becomes crucial for relating papers in the citation domain, besides topics, authors...
3) **Journals:** To capture journals in the triples we will also need to extract the `:hasVolume` relationship between journals and volumes.
4) **Reviews:** The 3 following relationships establish who reviews a paper and similarly, which papers are reviewed by a certain author.
   - `:hasReviewer`
   - `:performedBy`
   - `:assignedTo`
5) **Events:** Finally for sake of completion and capturing proceedings related information hierarchically (Proceedings, Edition, Workshop, Conference...)
   - `:hasProceedings`
   - `:hasEdition`

We present in file *abox_export.tsv* the most relevant semantic information that will be used to train the knowledge graph embeddings and in the downstream machine learning tasks. This subset contains the hierarchical structure of publications and the topologies of this network.

More information can be found in the script *src/embeddings/extract.py* inside our public open-source repository [6].

B. Familiarity

Before delving into modelling, a wise step to take is to review the current literature and most commonly used graph embedding models to gain familiarity on the properties of the embeddings they produce (Symmetry, Transitivity...).

Note that given the flexible nature of these numerical representations can lead to potential errors when performing similarity or evaluation tasks, so a filtering by types (rdf:type) has become a standard in our data processing pipelines.

More information can be found in the script *src/embeddings/-familiarity.ipynb* inside our public open-source repository [6].

1) The most basic model: TransE

Firstly, we start with the TransE model, one of the most basic and used embedding models currently despite its simplicity. We used the following hyperparameters

The pseudo-code procedure for finding the most likely citation paper and corresponding author are the following (In a general setting, we can use relations :cites and :hasCorrespondingAuthor to achieve our purposes specifically):

TABLE I: TransE Hyperparameters, seed 2025

Hyperparameter	**Value**
Iterations	25
Learning Rate α	0.01
Embedding Dimensionality (d)	256
Negatives samples per positive (np)	3
Loss Function $\mathcal{L}$	Margin Ranking
Margin γ	1.0

Given TransE embeddings:

$$\mathbf{h}, \mathbf{r}, \mathbf{t} \in R^d$$

Step 1: Compute the target translation vector:

$$\mathbf{v}_{\text{target}} = \mathbf{h} + \mathbf{r}$$

Step 2: For each candidate tail entity t_i, compute the L2 distance:

$$\text{score}(t_i) = \|\mathbf{v}_{\text{target}} - \mathbf{t}_i\|_2$$

Step 3: Choose the entity with minimum distance (And filter by types!):

$$t^* = \arg\min_{t_i \in \mathcal{T}} \text{score}(t_i)$$

The results in practice were the following:

Test Paper	`<MAGIC_Microlensing_Analysis_Guided_by_Intelligent_Computation>`
Predicted Cited Paper	`<Identifying_multi-target_drugs_for_prostate_cancer_using_machine_learning-ass`
Predicted Author	`<Hugo_Richard>`

TABLE II: Example predictions from the knowledge graph embedding model. The model predicts the cited paper and author for the given test paper based on learned entity and relation embeddings.

2) Beyond TransE

In this following section we want to showcase the weakpoints of TransE and present a discussion related to the properties different models have in order to deal with complex relationships. Answers are provided in the same order as the statement:

1) Consider the following triples: (Author1, writes, Paper1), (Author2, writes, Paper1), (Author1, writes, Paper2). Then TransE optimal solution to minimze the $\mathcal{L}$ would be, bearing in mind the translational relationship established by TransE ($t \approx h + r$)

$$\begin{aligned} &\mathbf{Author}_1 + \mathbf{writes} \approx \mathbf{Paper}_1 \wedge \\ &\mathbf{Author}_2 + \mathbf{writes} \approx \mathbf{Paper}_1 \wedge \\ &\mathbf{Author}_1 + \mathbf{writes} \approx \mathbf{Paper}_2 \\ &\qquad \Rightarrow \quad Author_1 \approx Author_2 \wedge Paper_1 \approx Paper_2 \end{aligned}$$

2) This N-to-N issue can be solved by means of other models like TransH, which allow each relationship r to have its own projection hyperplane w_r characterized by its normal vector, allowing the same entity to have

different projections for different relationships r_k. This avoids oversimplified solutions as $Author_1 \approx Author_2$.

Then, the TransH scoring function is:

$$f_r(h,t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|^2$$

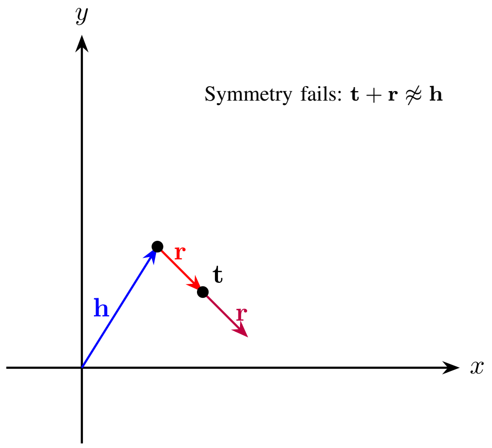
where:

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \cdot \mathbf{w}_r, \quad \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \cdot \mathbf{w}_r$$

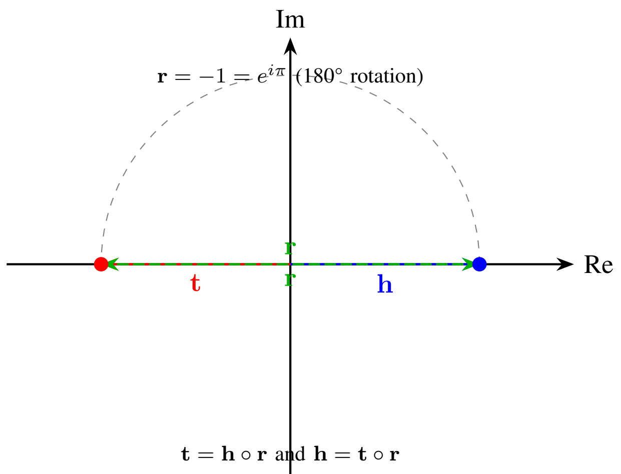
3) Consider the following triples: (Author1, collaboratesWith, Author2), (Author1, collaboratesWith, Author2). In the case of **Symmetric** relationships like collaboratesWith (which should undoubtedly give the pretty much the same scores), TransE will definitely fail, given its simple translational nature. Relationship r is fixed, so without a change in its direction its not possible to perform symmetry. So by means of optimizing $\mathcal{L}$, we would reach a degenerate solution as the following:

$$\begin{aligned} &\mathbf{Author}_1 + \mathbf{collaboratesWith} \approx \mathbf{Author}_2 \wedge \\ &\mathbf{Author}_2 + \mathbf{collaboratesWith} \approx \mathbf{Author}_1 \wedge \\ &\Rightarrow \quad \mathbf{collaboratesWith} \approx \mathbf{Author}_2 - \mathbf{Author}_1 \wedge \\ &\quad\quad \mathbf{collaboratesWith} \approx \mathbf{Author}_1 - \mathbf{Author}_2 \wedge \\ &\quad\quad\quad \Rightarrow \quad \mathbf{Author}_2 \approx \mathbf{Author}_1 \approx 0 \end{aligned}$$

4) The simple diagram below depicts the scenario where an already trained TransE model would be uncapable of capturing symmetricity. In case of training the model with symmetric triples, a degenerate solution would be found.


y
Symmetry fails: $\mathbf{t} + \mathbf{r} \not\approx \mathbf{h}$
$\mathbf{r}$
$\mathbf{t}$
$\mathbf{h}$
$\mathbf{r}$
x

5) RotatE solves the symmetry problem by using complex numbers and rotation in a complex plane. Instead of adding vectors like TransE, RotatE multiplies complex numbers, where every single entity and relation is represented as a complex number. For a symmetric relationship, RotatE can achieve symmetry by using a rotation of π radians using the Hadamard product $f(h,r,t) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$.


Im
$\mathbf{r} = -1 = e^{i\pi}$ (180° rotation)
Re
$\mathbf{r}$
$\mathbf{r}$
$\mathbf{t}$
$\mathbf{h}$
$\mathbf{t} = \mathbf{h} \circ \mathbf{r}$ and $\mathbf{h} = \mathbf{t} \circ \mathbf{r}$

Another example extracted from our collegues of stanford [7] (We also miss the OG spiderman...):

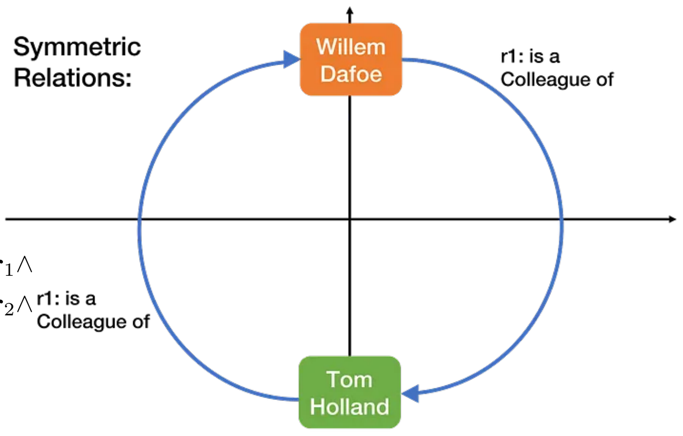


Fig. 9: Example of capturing a symmetrical relationship using RotatE

6) Finally, two other influential models that address the issue of symmetry in knowledge graph embeddings are the famous **DistMult** and **ComplEx** models:
DistMult is a bilinear model that represents relations as diagonal matrices, simplifying the scoring function to an element-wise product between embeddings of the head, relation, and tail entities. Its scoring function is given by:

$$f(h,r,t) = \sum_i h_i \, r_i \, t_i$$

Because the multiplication involved is commutative (i.e., $h_i r_i t_i = t_i r_i h_i$), DistMult is inherently suited for modeling **symmetric relations** such as ”sibling of” or ”related to”. However, this commutativity also limits its expressiveness in capturing **asymmetric relations** like ”parent of” or ”located in”, since it cannot differentiate directionality in relationships. Despite this limitation, DistMult remains popular due to its simplicity, efficiency, and relatively low computational cost.

ComplEx builds upon the DistMult framework by embedding entities and relations into a complex vector space. The key innovation in ComplEx is the use of complex conjugation on the tail entity embedding in the scoring function:

$$f(h,r,t) = \Re\left(\sum_i h_i\, r_i\, \overline{t_i}\right)$$

where $\overline{t_i}$ denotes the complex conjugate of the tail embedding component and $\Re(\cdot)$ extracts the real part of the complex number. This conjugation breaks the symmetry property present in DistMult, enabling ComplEx to effectively model both **symmetric and asymmetric relations**. This increased expressiveness comes at a moderate computational cost increase compared to DistMult, but it remains efficient enough for large-scale knowledge graphs.

In summary, the choice between DistMult and ComplEx represents a trade-off between simplicity and expressiveness. **DistMult** offers a simpler and faster model that works well for symmetric relations but fails to capture relational asymmetry. In contrast, **ComplEx** introduces complex-valued embeddings to overcome this limitation, providing a more powerful and flexible model capable of accurately representing a wider variety of real-world relationships at a reasonable computational cost.

C. Training KGE's

1) Methodology

After extracting and transforming the aforementioned data, getting familiarity and mathematical notions of the models, empirical exploration of multiple model graph embedding algorithms is presented below.

Recalling the discussion between DistMult, TransE, TransH and ComplEx, we have the prior intuition of that the most performant embedding model after a proper training, will be ComplEx.

Nevertheless, we have tried the 5 most promising embedding models (TransE, TransH, RotatE, DistMult and ComplEx) with the following simple methodology for each model:

1) Processing: Firstly process the data into head-relation-tail format and perform splitting and resampling, using 3 fold cross-validation in our case.
2) Hyperparameter Optimization: Perform grid-search cross validation over the following hyperparameters of the embedding models (Fixing epochs though), however note that this technique brutally shrinks the feasible search space:
   - Embedding dimension (d): [64, 128, 256]
   - Learning rate (α): [0.01, 0.001, 0.0001]
   - Number of negative samples per positive triple (np): [3, 5]
3) Model Fitting: via backpropagation coupled with ADAM optimization algorithm, using regularization techniques such as early stopping and fixing multiple hyperparameters such as batch size, patience...
4) Evaluation: All evaluations on test tests were guided by the following metrics (Note that d is not really and evaluation metric but rather a measure of model complexity):
   - Mean Reciprocal Rank.
   - Hits@k measure, used for $k \in [1, 5, 10]$.
   - Variance: This helps to address high variance (overfitting) models
   - Embedding Size d: In order to assess model complexity, the embedding size is a factor that should be considered, the simpler the model, the better.

2) Results and Discussion

The results were the following using the best hyperparameter combination for every single model:

TABLE III: Performance metrics of various knowledge graph embedding models with embedding size and variance.

Model	MRR	Variance	Hits@1	Hits@5	Hits@10	Size (d)
TransH	0.1961	1093800.25	0.1836	0.2042	0.2106	64
RotatE	0.1471	3251453.25	0.1280	0.1628	0.1765	256
TransE	0.0592	2440683.00	0.0277	0.0895	0.1067	64
DistMult	0.0511	7627781.50	0.0453	0.0564	0.0594	256
ComplEx	0.0017	6707285.00	0.0004	0.0016	0.0026	256

Surprisingly, TransH has been the most performant model by far followed by RotatE, and then the rest which do not even come close. The superior performance of TransH and RotatE on our publications KG suggests the presence of context-dependent and semantically diverse relations. These models' ability to capture asymmetry, relation-specific role dynamics, and complex multi-relational patterns aligns with the nature of scholarly networks where entities such as authors, papers, and institutions interact through richly typed and direction-sensitive relations. We suspect however that we could accidentally used the whole abox exportation data rather than the filtered version, which could be causing DistMult and ComplEx to have such abnormally low scores due to too much noise. This remains a topic to discover.

One important thing to note is that evaluation metrics absolute performance scores across all models were fairly low. This is most likely due to the limited size and sparsity of the knowledge graph, which restricts the models' ability to generalize. Moreover, its important to realize that due to our input data, the relationship distribution is not uniform, there are some relationships that monopolize most of the triples (:hasAuthor), making the learning unbalanced. Also this processed is affected by the feasibility of triple types, which they aren't considered at all during training.

The hyperparameters of this most performant embedding model is:

TABLE IV: Best Hyperparameters for TransH (seed 2025)

Hyperparameter	Value
Embedding Dimensionality (d)	64
Number of Epochs	10
Batch Size	64
Negatives per Positive (np)	3
Learning Rate α	0.001
Early Stopping	True
Patience	5
Relative Delta	0.002
Scoring Function Norm	2
Loss Function $\mathcal{L}$	Margin Ranking
Margin γ	1.0

More information can be found in the script *src/embeddings/-train.py* inside our public open-source repository [6].

D. Applications

Finally, after training our final embedding model, the focus of this work lies on the applications that this numerical representation offers from our knowledge graph. The applications are vast, but we will mainly focus on 3 quick-wins that can be extremely helpful for academic online systems, such as author recommendation, paper citation recommendation and clustering of the 2 aforementioned. Leveraging embeddings, the semantic structure of the knowledge graph can be ingested by common machine learning algorithms. We have opted for building an interactive Web UI for showcasing the results of our work, via Streamlit.

Firstly, one of the simplest yet most effective tasks that scientific users could make use of, would be a paper recommendation system in order to identify the K most similar papers to our publication. This can be extremely useful for detecting duplicates or quasi-duplicates (Which is a common thing nowadays in the machine learning research field, where 2 research groups are completely unaware that they are researching similar or even the same thing), citation recommendations and would simplify the creation of surveys. Thanks to embeddings numerical representations could be implemented via computing the similarity matrix between all papers and obtaining the K most relevant ones. The most important decisions for this model to be performant are both the embedding model and the distance metric, which in this case makes sense to use cosine similarity. The literature points out that joint cosine similarity (Hadamard product) and regular pairwise cosine similarity are the empirically most performant and interpretable metrics for training and evaluation tasks respectively [8].

Secondly, just like the previous task, the same recommendation logic can be applied to authors—helping find colleagues or topically related researchers. This is especially useful in niche domains (such as Deep Learning Compilers and ML-guided optimization, which the authors are familiar with), where exploring the literature can be challenging. Interestingly, this problem is likely easier than paper recommendation, thanks to the rich semantic structure around authors in the knowledge graph (e.g., affiliations, co-authorship, topics). This covers multiple use cases such as finding potential collaborators for research, hiring talent for research labs and tracking academic lineage.

Lastly, since we are working with similarity measures and recommendation systems, a valuable analytical feature to offer users would be the clustering of authors and papers separately. However, the high dimensionality of the embedding space ($R^d, \ d \gg 3$) presents a challenge for visualization—unfortunately, our current biological visualization hardware (also known as eyeballs) cannot directly perceive beyond three dimensions, at least until evolution grants us an upgrade. To overcome this, dimensionality reduction algorithms can be applied within the dashboard, allowing users to choose between t-SNE, PCA, and UMAP. We recommend using UMAP for this task due to its superior ability to capture non-linear relationships in the data, despite the additional computational cost involved. This covers also multiple potential use cases such as reviewers sugesstion, trend detection, and research groups identification.

Furthermore, the quality of training was severly impacted by 2 factors that are of great interest: Our dataset is skewed, some relationships appear way more often than others. Also types aren't enforced during optimization, leading to potential mismatches, so data processing still has room for improvement, after having finished this first iteration of modelling.

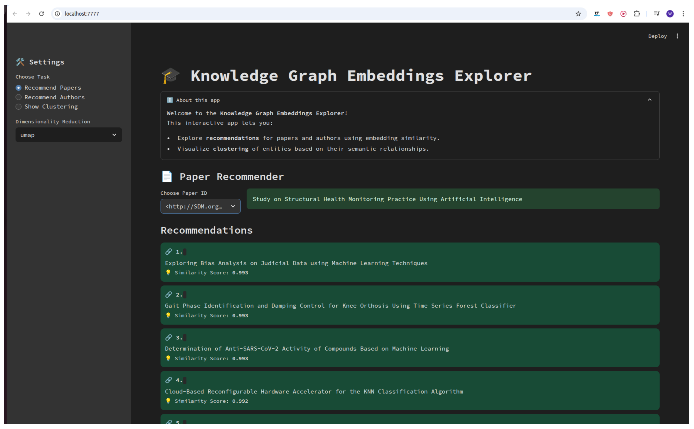
All in all, we seek to provide an interesting recommender system coupled with analytics for enhancing data-driven decisions in research groups and academic final users who may want to benefit from the benefits of this well-curated data and knowledge graph. More information can be found in the script *src/embeddings/application.py* inside our public open-source repository [6].

IV. Conclusions

For further investigation, we encourage the reader to explore the current state-of-the-art in knowledge graph embedding models, which have seen significant advances in recent years [9], [10], [11]. Another important avenue for future work lies in enhancing our model training pipeline. As noted, our hyperparameter optimization approach is relatively basic, relying primarily on resampling techniques such as K-fold cross-validation and grid search. This could be substantially improved by adopting more sophisticated black-box optimization methods, including Bayesian optimization frameworks like Tree-Parzen Estimators (e.g., Optuna) or Gaussian process-based methods, as well as Genetic Algorithms, Reinforcement Learning, or even the simplest random search strategies. While we deprioritized this due to the constrained timeline of the project, making use of these advanced techniques has the potential to yield significant improvements in embedding quality [12], [13].

In our application we have detected that the recommendation system find consistently papers and authors that are 90%+ similar in cosine similarity score, this could well be due to the low dimensionality of the embedding (64) or that TransH has maybe fallen into a degenerate solution that would be considered ”cheating”. Nevertheless, the recommendations are sensible and make a lot of sense, so we encourage the user to empirically check this fact.

All in all, this project has been an eye-opening experience to the knowledge graphs niche and has shown us that coupled with modern techniques such as embeddings, this semantic data model can be of great interest for business intelligence tasks by means of machine learning. After an extensive exploration and gaining intuition of the difference between embedding models (TransE, TransR, TransH, ComplEx, RotatE, DistMult...), the authors have implemented a fully fledged recommendation system for end users to utilize. We hope our work is of great interest for the research community.



Settings

Choose Task

- [✓ Recommend Papers]
- Recommend Authors
- Show Clustering

Dimensionality Reduction: umap

Knowledge Graph Embeddings Explorer

About this app

Welcome to the **Knowledge Graph Embeddings Explorer**!
This interactive app lets you:

- Explore **recommendations** for papers and authors using embedding similarity.
- Visualize **clustering** of entities based on their semantic relationships.

Paper Recommender

Choose Paper ID: <http://SDM.org…

Study on Structural Health Monitoring Practice Using Artificial Intelligence

Recommendations

1.
Exploring Bias Analysis on Judicial Data using Machine Learning Techniques
Similarity Score: 0.993

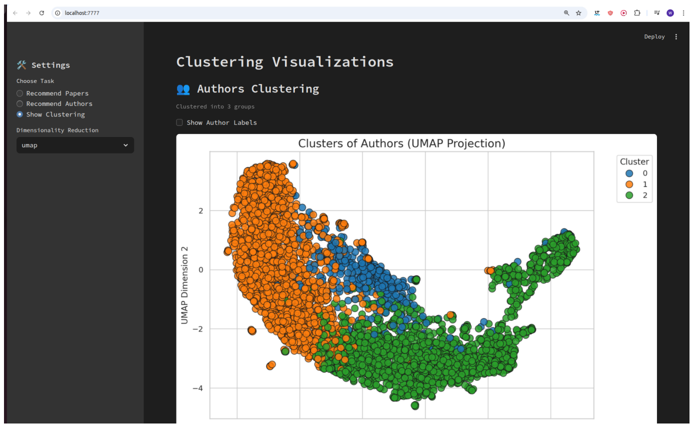
2.
Gait Phase Identification and Damping Control for Knee Orthosis Using Time Series Forest Classifier
Similarity Score: 0.993

3.
Determination of Anti-SARS-CoV-2 Activity of Compounds Based on Machine Learning
Similarity Score: 0.993

4.
Cloud-Based Reconfigurable Hardware Accelerator for the KNN Classification Algorithm
Similarity Score: 0.992

5.

Fig. 10: Publications recommendation system UI



Settings

Choose Task

- Recommend Papers
- Recommend Authors
- [✓ Show Clustering]

Dimensionality Reduction: umap

Clustering Visualizations

Authors Clustering

Clustered into 3 groups

Show Author Labels

Fig. 11: Clustering analytical dashboard UI

References

[1] “Graphdb knowledge graph database,” https://graphdb.ontotext.com/, accessed: 2025-06-13.

[2] Q. Wang *et al.*, “Knowledge graph embedding: A survey of approaches and applications,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[3] T. Trouillon *et al.*, “Complex embeddings for simple link prediction,” in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 2071–2080.

[4] Z. Huang *et al.*, “Knowledge graph embedding based recommendation: A survey,” *Frontiers of Computer Science*, vol. 13, pp. 1083–1102, 2019.

[5] X. Wang *et al.*, “Knowledge graph embedding: A review of approaches and applications,” in *IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2050–2056.

[6] I. M. Walter Troiani, “Mds-sdm-academicgraphs: Create, visualize, modify and analyze the graphs created from academic papers (kge),” https://github.com/eZWALT/SDM-Semantic-Data-Management, 2025, accessed: 13-06-2025.

[7] L. Tao, S. W. Tanuwidjaja, and P. Carlson, “Simple schemes for knowledge graph embedding,” 2022, accessed: 2025-06-13. [Online]. Available: https://medium.com/stanford-cs224w/simple-schemes-for-knowledge-graph-embedding-dd07c61f3267

[8] L. Wang, S. Deng, and X. Gao, “RoCS: Knowledge graph embedding based on joint cosine similarity,” *Electronics*, vol. 13, no. 1, p. 147, 2023, demonstrates that joint cosine similarity significantly improves Hits@1 on WN18RR and FB15K-237 compared to ComplEx and RotatE.

[9] Y. Cao *et al.*, “Rotate3d: Knowledge graph embedding with rotation in three-dimensional space,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[10] M. Abboud, H. Chekol, and et al., “Boxe: Embedding entities and relations as boxes,” in *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[11] S. Zhang, M. Yao, and et al., “Quate: Learning knowledge graph embeddings in quaternion space,” in *Advances in Neural Information Processing Systems*, 2022.

[12] S. Ott *et al.*, “Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs,” in *International Conference on Learning Representations*, 2023.

[13] F. Wei *et al.*, “Kepler: A unified model for knowledge embedding and pre-trained language representation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.