Knowledge Graphs Project: Research Publication Domain

Jonàs Salat, Albert Vidal

June 10, 2025

1 Exploring DBpedia

[This section is for exploration purposes only, no deliverables required]

2 Ontology Creation

2.1 TBOX Definition

The TBOX (Terminological Box) defines the schema, classes, and properties of our knowledge graph. It serves as the structural foundation for our ontology.

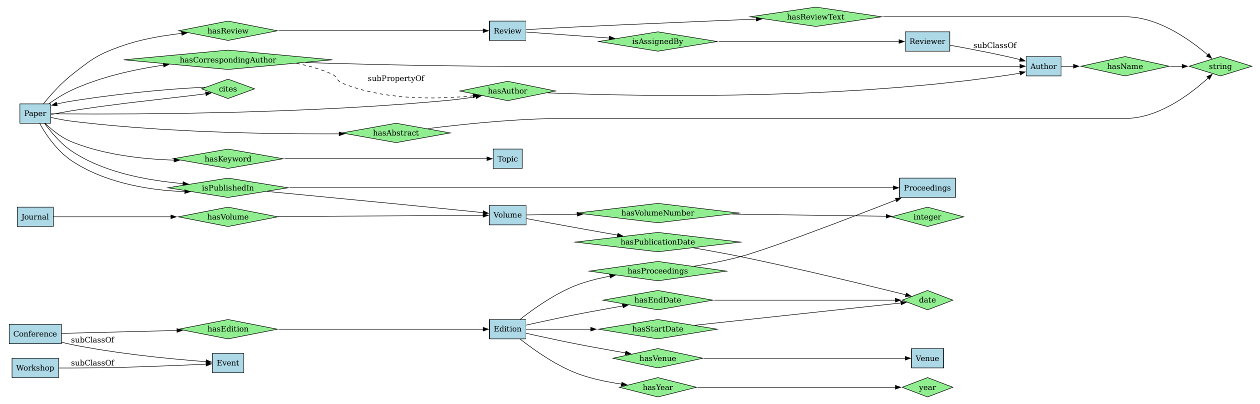

Figure 1: Visualization of the Publication Domain TBOX showing classes, properties, and their relationships. Classes are shown in light blue boxes, properties in light green diamonds, and subclass/subproperty relationships with dashed lines.

2.1.1 Classes

The TBOX defines the following main classes:

- **Paper**: A research paper
- **Author**: A person who writes papers
- **Event**: An academic event where research is presented
- **Conference**: A well-established research forum (subclass of Event)
- **Workshop**: A forum for exploring new trends (subclass of Event)
- **Edition**: A specific instance of a conference or workshop
- **Journal**: A periodical publication
- **Volume**: A collection of papers in a journal
- **Proceedings**: A collection of papers from a conference/workshop edition
- **Review**: An evaluation of a paper

- **Reviewer**: A scientist who reviews papers (subclass of Author)
- **Topic**: A subject area of a paper
- **Venue**: A location where an edition takes place

2.1.2 Properties

The TBOX defines various properties to establish relationships between classes:

Paper Properties

- **hasAbstract**: Links a paper to its abstract (domain: Paper, range: string)
- **hasKeyword**: Links a paper to its topics (domain: Paper, range: Topic)
- **cites**: Links a paper to papers it cites (domain: Paper, range: Paper)
- **isPublishedIn**: Links a paper to its publication venue (domain: Paper, range: Proceedings or Volume)
- **hasAuthor**: Links a paper to its authors (domain: Paper, range: Author)
- **hasCorrespondingAuthor**: Links a paper to its corresponding author (domain: Paper, range: Author, subproperty of hasAuthor)
- **hasReview**: Links a paper to its reviews (domain: Paper, range: Review)

Author Properties

- **hasName**: Links an author to their name (domain: Author, range: string)

Event Properties

- **hasEdition**: Links a conference/workshop to its editions (domain: Conference/Workshop, range: Edition)

Edition Properties

- **hasVenue**: Links an edition to its venue (domain: Edition, range: Venue)
- **hasStartDate**: Links an edition to its start date (domain: Edition, range: date)
- **hasEndDate**: Links an edition to its end date (domain: Edition, range: date)
- **hasYear**: Links an edition to its year (domain: Edition, range: year)
- **hasProceedings**: Links an edition to its proceedings (domain: Edition, range: Proceedings)

Journal Properties

- **hasVolume**: Links a journal to its volumes (domain: Journal, range: Volume)

Volume Properties

- **hasPublicationDate**: Links a volume to its publication date (domain: Volume, range: date)
- **hasVolumeNumber**: Links a volume to its number (domain: Volume, range: integer)

Review Properties

- **isAssignedBy**: Links a review to its assigner (domain: Review, range: Reviewer)
- **hasReviewText**: Links a review to its text content (domain: Review, range: string)

2.1.3 Inverse Properties

The following inverse relationships are defined:

- **isPublishedIn** and **containsPaper**
- **hasEdition** and **isEditionOf**
- **hasVolume** and **isVolumeOf**

2.2 ABOX Creation

The ABOX (Assertional Box) of our knowledge graph was created by processing and transforming the provided CSV datasets into RDF triples. The implementation handles the complex relationships between different entities while ensuring data integrity and consistency.

2.3 Implementation Details

The ABOX creation process begins by loading all CSV files into pandas DataFrames for efficient data manipulation. Each entity type is processed sequentially, with careful attention to maintaining referential integrity between related entities. The implementation uses a custom URI generation function that creates deterministic, unique identifiers for each entity by hashing their primary keys.

For topics, we extract unique keywords from the papers dataset and create corresponding Topic instances. This approach ensures that each keyword appears only once in the knowledge graph, regardless of how many papers use it. Venues are similarly deduplicated, with each unique venue name creating a single Venue instance.

The author creation process handles both required and optional attributes. While author ID and name are mandatory, email addresses are added only when available. This pattern of handling optional attributes is consistently applied throughout the implementation, ensuring that the knowledge graph only contains valid, non-null data.

Conference and Workshop instances are created as separate entities, each with their own unique identifiers. The implementation maintains a clear distinction between these event types while ensuring they can be properly linked to their respective editions and papers.

Edition instances are created with careful validation of their relationships to venues and events. The implementation ensures that each edition is properly associated with a valid venue and includes temporal information (year) when available. This temporal aspect is particularly important for maintaining the chronological integrity of the knowledge graph.

Journal and Volume instances are created with a focus on maintaining the hierarchical relationship between them. Each volume is properly linked to its parent journal, and the implementation ensures that this relationship is bidirectional, with both the volume referencing its journal and the journal referencing its volumes.

Paper instances form the core of the knowledge graph, with the most complex set of relationships. Each paper is linked to its authors, topics, and publication venue (either an edition or a volume). The implementation carefully handles the different types of publication venues (conference, workshop, or journal) and creates the appropriate relationships based on the venue type.

2.4 Data Validation and Error Handling

The implementation includes robust data validation to ensure the integrity of the knowledge graph. NaN values are handled explicitly, with appropriate error messages when required fields are missing. The code validates foreign key relationships before creating them, ensuring that references between entities are always valid.

For relationships between papers and their publication venues, the implementation performs additional validation to ensure that the referenced editions or volumes exist and are properly linked. This is particularly important for maintaining the semantic consistency of the knowledge graph.

The citation relationships are validated to ensure that both the citing and cited papers exist in the knowledge graph. This prevents the creation of dangling references and maintains the integrity of the citation network.

Review relationships are created with careful validation of both the paper and reviewer existence. The implementation also handles optional review text, adding it to the knowledge graph only when available.

2.5 Knowledge Graph Statistics

To better understand the structure and content of our knowledge graph, we analyzed various aspects of the data. Figure **??** shows the distribution of different entity types in the knowledge graph, revealing the relative abundance of each type of entity.

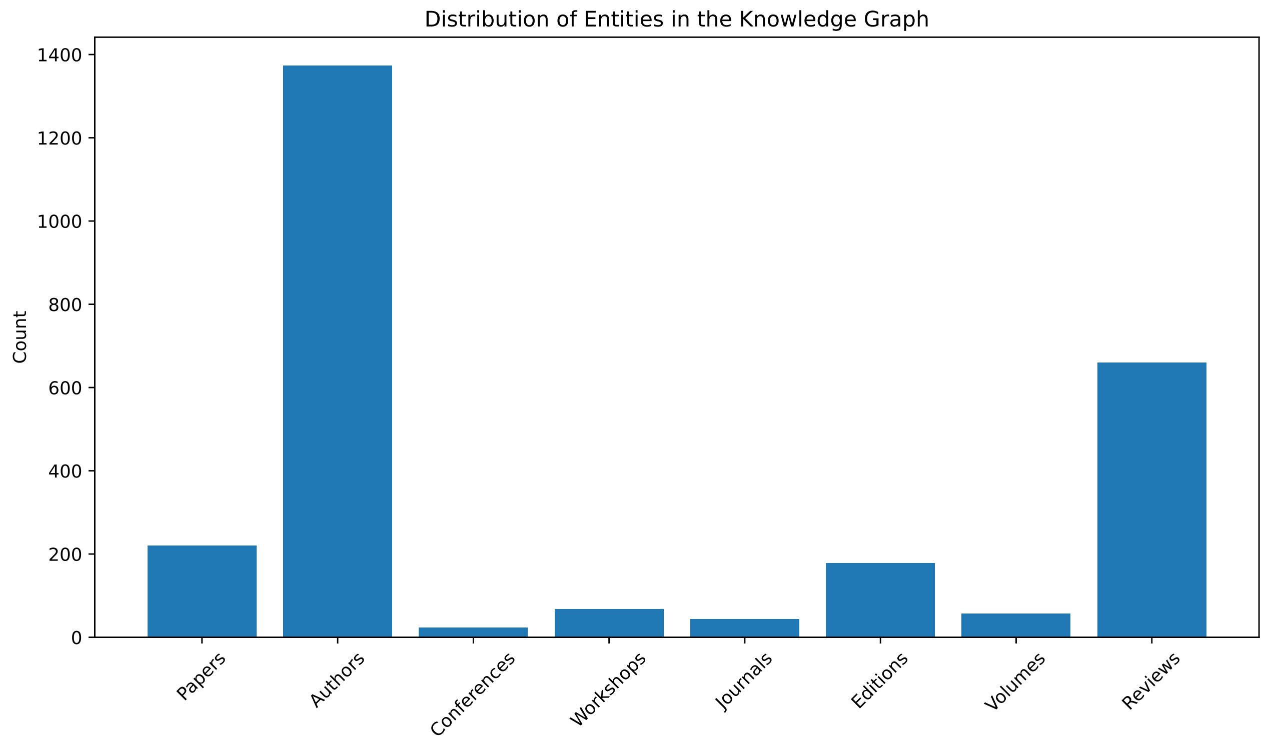

Figure 2: Distribution of entities in the knowledge graph. The plot shows the count of each entity type, providing insight into the scale and composition of the knowledge graph.

The relationships between entities are shown in Figure **??**, which illustrates the frequency of different types of connections in the knowledge graph. This helps us understand the density and nature of the relationships in our graph.

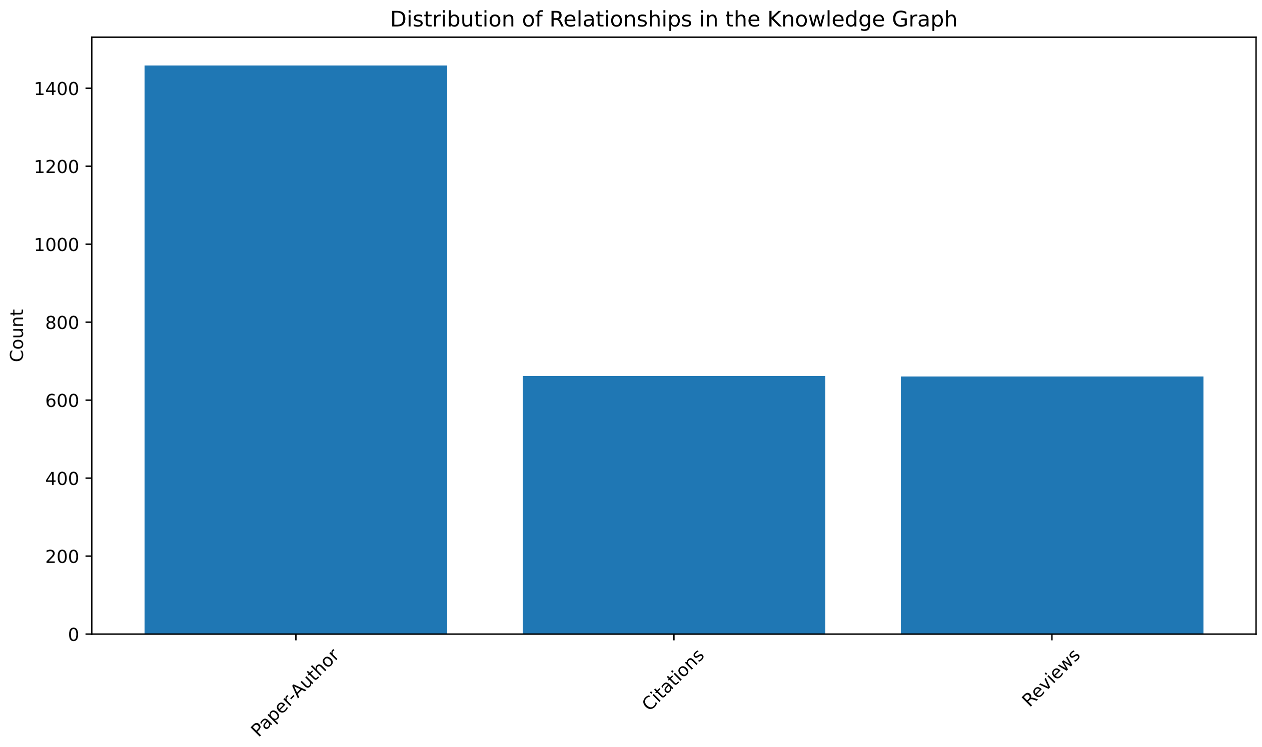

Figure 3: Distribution of relationships in the knowledge graph. The plot shows the count of different types of relationships, highlighting the most common connections between entities.

The distribution of publication venues is shown in Figure ??, which provides insight into the balance between different types of publication outlets in our dataset.

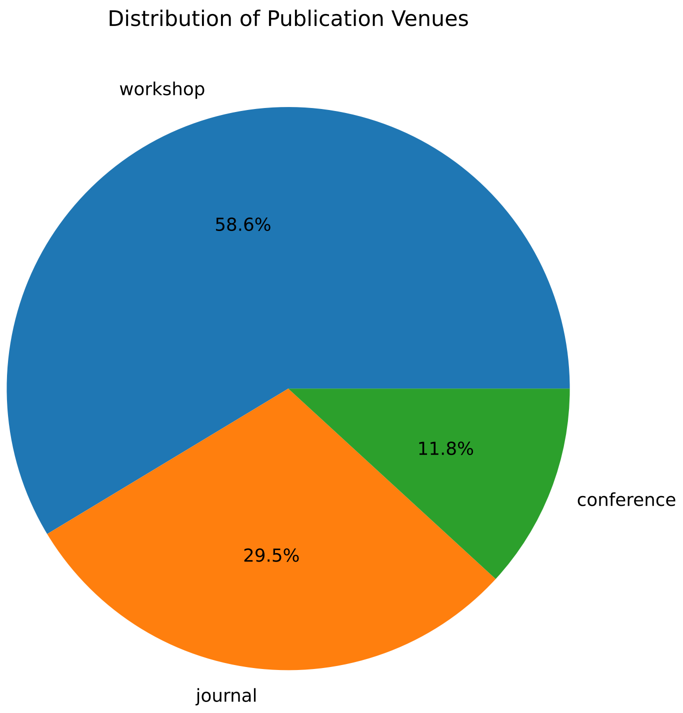

Figure 4: Distribution of publication venues. The pie chart shows the proportion of papers published in different types of venues (conferences, workshops, and journals).

The temporal distribution of papers is shown in Figure **??**, which helps us understand the time span covered by our knowledge graph and any trends in publication frequency.

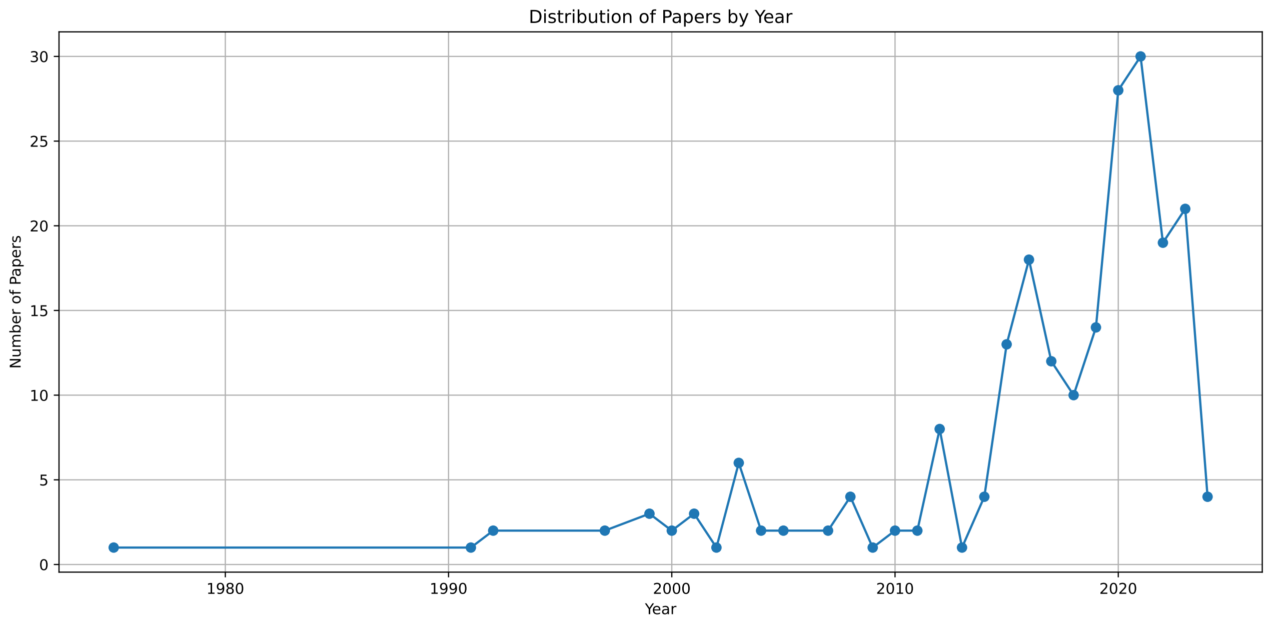

Figure 5: Distribution of papers by year. The line plot shows the number of papers published each year, revealing trends in publication frequency over time.

Finally, Figure **??** shows the most common keywords in the papers, providing insight into the main research topics covered in our knowledge graph.

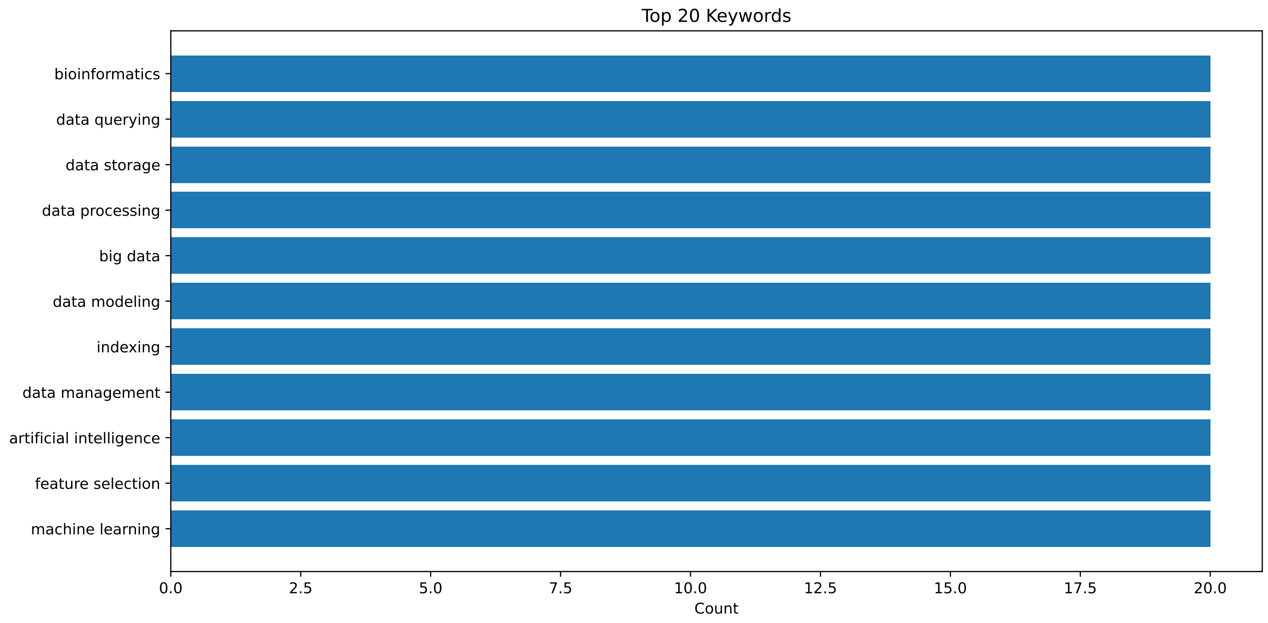

Figure 6: Top 20 keywords in the papers. The horizontal bar chart shows the most frequently occurring keywords, indicating the main research topics in the knowledge graph.

These statistics provide valuable insights into the structure and content of our knowledge graph, helping us understand its scale, composition, and the relationships between different entities.

2.6 Assumptions and Design Decisions

The implementation makes several key assumptions and design decisions to handle the complexity of the academic publication domain. First, we assume that each paper can be published in exactly one venue (either a conference/workshop edition or a journal volume). This assumption simplifies the relationship structure while maintaining the semantic accuracy of the knowledge graph.

For topics, we assume that keywords are case-sensitive and that whitespace should be preserved. This decision allows for more precise topic matching and querying. The implementation also assumes that keywords are comma-separated in the input data, with each keyword potentially containing internal spaces.

The implementation assumes that author IDs are unique across the entire dataset, allowing for consistent author identification even when the same author appears in multiple papers. This assumption is crucial for maintaining the integrity of author-related relationships.

For temporal data, we assume that years are represented as integers and are valid calendar years. This assumption allows for proper typing of temporal data in the knowledge graph and enables temporal reasoning.

The implementation assumes that each review is uniquely identified by the combination of paper ID and reviewer ID. This assumption allows for proper handling of multiple reviews for the same paper by different reviewers.

Finally, the implementation assumes that the CSV files are properly formatted and that the data types of each column are consistent with their expected usage in the knowledge graph. This includes proper handling of numeric IDs, string values for names and titles, and boolean values for flags like corresponding author status.

These assumptions and design decisions, combined with the robust data validation, ensure that the resulting knowledge graph is both semantically accurate and practically useful for querying and analysis.

2.7 Querying the Ontology

[To be filled with SPARQL queries and their results]

3 Knowledge Graph Embeddings

3.1 Importing the Data

[To be filled with data selection and import details]

3.2 Getting Familiar with KGEs

[To be filled with KGE model exploration and analysis]

3.3 Training KGEs

[To be filled with KGE training experiments and results]

3.4 Exploiting KGEs

[To be filled with KGE application details and results]