



# AUTO RDF2GML: Facilitating RDF Integration in Graph Machine Learning

Michael Färber<sup>1</sup>(✉) , David Lamprecht<sup>2</sup> , and Yuni Susanti<sup>3</sup>

<sup>1</sup> ScaDS.AI & TU Dresden, Dresden, Germany

[michael.faerber@tu-dresden.de](mailto:michael.faerber@tu-dresden.de)

<sup>2</sup> metaphacts GmbH, Walldorf, Germany

[dl@metaphacts.com](mailto:dl@metaphacts.com)

<sup>3</sup> Fujitsu Ltd., Kanagawa, Japan

[yuni.susanti@fujitsu.com](mailto:yuni.susanti@fujitsu.com)

**Abstract.** In this paper, we introduce AUTO RDF2GML, a framework designed to convert RDF data into data representations tailored for graph machine learning tasks. AUTO RDF2GML enables, for the first time, the creation of both content-based features—i.e., features based on RDF datatype properties—and topology-based features—i.e., features based on RDF object properties. Characterized by automated feature extraction, AUTO RDF2GML makes it possible even for users less familiar with RDF and SPARQL to generate data representations ready for graph machine learning tasks, such as link prediction, node classification, and graph classification. Furthermore, we present four new benchmark datasets for graph machine learning, created from large RDF knowledge graphs using our framework. These datasets serve as valuable resources for evaluating graph machine learning approaches, such as graph neural networks. Overall, our framework effectively bridges the gap between the Graph Machine Learning and Semantic Web communities, paving the way for RDF-based machine learning applications.

- Code & Framework:** <https://github.com/davidlamprecht/AutoRDF2GML/MIT License>
- GML Dataset LPWC:** <https://doi.org/10.5281/zenodo.10299366CCBY-SA4.0 License>
- GML Dataset SOA-SW:** [https://doi.org/10.5281/zenodo.10299429CreativeCommonsZero \(CC0\)](https://doi.org/10.5281/zenodo.10299429CreativeCommonsZero (CC0))
- GML Dataset AIFB:** <https://doi.org/10.5281/zenodo.10989595CCBY4.0 License>
- GML Dataset LinkedMDB:** <https://doi.org/10.5281/zenodo.10989683CCBY4.0 License>

## 1 Introduction

Knowledge representation based on RDF is designed to be interpretable by both humans and machines. Integrating RDF with graph machine learning, such

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

G. Demartini et al. (Eds.): ISWC 2024, LNCS 15233, pp. 115–133, 2025.

[https://doi.org/10.1007/978-3-031-77847-6\\_7](https://doi.org/10.1007/978-3-031-77847-6_7)

as in Graph Neural Network (GNN) approaches, however, presents significant challenges, as RDF differs remarkably from the data representations used in machine learning. The primary challenge lies in modeling entity relationships and attributes as feature vectors, diverging from RDF with its explicit knowledge representation. Additionally, the inherent heterogeneity (variety of entity and relation types) and sparsity of RDF data (few relations per entity) potentially affect the consistency and robustness of the learning process [50, 51].

Existing *frameworks* for preparing RDF data for graph machine learning (GML) tasks typically lack the capability to transform RDF data into a propositionalized format, such as a feature matrix format. Instead, they convert RDF data into a standard feature matrix without considering the graph structure [3]. Thus, they currently ignore both the different entity types and the object properties of RDF instances, which are crucial parts of RDF data.

Furthermore, current *benchmarks* in graph machine learning, such as those provided by PyTorch Geometric, differ in the provisioning of node features, i.e., the modeling of nodes. Typically, we can categorize the available node features for datasets for graph machine learning into the following types: (1) content-based natural language descriptions (NLD), (2) other content-based literals (e.g., numeric, categorical, or boolean values), and (3) topology-based features that encapsulate the graph structure [16, 25, 36]. While existing benchmarks cover both homogeneous and heterogeneous graphs, they focus on different aspects. For homogeneous graphs, they typically prioritize the content-based features, i.e., the node features derived from natural language descriptions (NLD) of node attributes such as their labels and descriptions, while benchmarks for heterogeneous graphs typically prioritize the diversity in the graph structures or topology. As a consequence, there is a significant gap in these benchmarks regarding the consideration of different kinds of semantics and a systematic analysis of their impact on graph machine learning models. This issue becomes evident when evaluating GNN-based models, as they frequently compute topology-based features for benchmarks that do not provide node features for all node types on-the-fly. Thus, analyzing whether a superior performance of a GNN-based model stems from its advanced architecture, or merely from the topology-based node features (which is then feature engineering), presents a significant challenge [35].

In this paper, we present AUTORDF2GML, a framework to effortlessly transform any given RDF data into ready-to-use heterogeneous graph datasets for graph machine learning. The generated datasets contain numeric vector features represented in feature matrices as the node features, derived from content-based (i.e., RDF datatype properties) and topology-based (i.e., RDF object properties) information of the RDF data. A notable advantage of the framework is its ability to automatically select and transform content-based features from the RDF data. Our framework allows users who are less familiar in RDF and SPARQL, such as those in the GNN field, to easily leverage RDF data for their research and applications. AUTORDF2GML can be installed via `pip install autordf2gml` and is easily set-up with a single-file configuration design: users are only required to define the RDF classes and properties, eliminating the need

for specifications of complex SPARQL queries. Therefore, it effectively serves as a bridge between the Graph Machine Learning and Semantic Web communities, facilitating an access to a vast amount of Linked Open Data for GML purposes.

Overall, in this paper, we make the following contributions:

1. We introduce AUTORDF2GML, a framework to semi-automatically transform RDF data into ready-to-use heterogeneous graph datasets for graph machine learning.
2. With the proposed AUTORDF2GML framework, we transform four exemplary publicly available, large RDF knowledge graphs into heterogeneous graph benchmark datasets for GML (see links on page 1).

The paper is structured as follows: In Sect. 2, we review related work on RDF data propositionalization and heterogeneous graph benchmarks. Section 3 introduces our framework for transforming RDF data for graph machine learning. In Sect. 4, we propose our RDF-based benchmarks. We show the potential usage of our framework and benchmarks in Sect. 5 and conclude in Sect. 6.

## 2 Related Work

In this section, we first address the processing of RDF data for use in graph machine learning applications, such as graph neural networks, a process known as *propositionalization*. Subsequently, we outline heterogeneous graph benchmarks.

### 2.1 Propositionalization of RDF Data

Propositionalization of RDF data refers to the task of transforming raw RDF data into the format required by a given learning algorithm, such as a graph neural network [32]. Most data mining algorithms require a feature vector representation of the data as input, thus each instance is represented as a feature vector  $(f_1, f_2, \dots, f_n)$ , where the features can be binary, numerical, or nominal values [40, 42]. Several approaches to generate such features from RDF data have been proposed. A comparison of the prominent techniques for the propositionalization of RDF data is summarized in Table 1, and outlined below.

**Table 1.** Overview prepropositionalization of RDF data.

	Degree of automation	Feature generation		Output	
		Data type properties	Object properties	Feature vectors	Graph structure encoding
Cheng et al. [7]	non-automatic	✗	✓	✓	✗
LiDDM [30]	non-automatic	✓	✗	✓	✗
RapidMiner [31]	non-automatic	✓	✗	✓	✗
FeGeLOD [37]	automatic	✓	✗	✓	✗
Literal2Feature [3]	automatic	✓	✗	✓	✗
AUTORDF2GML	semi-automatic	✓	✓	✓	✓

Cheng et al. [7] present an approach for extracting features from RDF data based on user-specified feature types and SPARQL. The evaluation results suggest that utilizing semantic features (e.g., the taxonomy) improves the performance of the models in comparison to utilizing solely standard features, such as the attributes. However, unlike AUTORDF2GML, no content-based information (i.e., RDF datatype properties such as descriptions) is used for feature construction and the user is required to define the SPARQL queries manually.

LiDDM [30] is a framework for data mining on the Semantic Web, and the data is typically retrieved via SPARQL to extract the features. LiDDM supports the integration of data from various Linked Open Data resources alongside a range of pre-processing techniques, including data filtering and data segmentation. However, these operations must be performed manually by the user.

RapidMiner’s semweb plugin [31] follows a similar approach and transforms RDF data into feature vectors, enabling its use within RapidMiner. However, unlike AUTORDF2GML, the user still needs to define SPARQL queries to obtain the desired data.

FeGeLOD [37] and its successor, the RapidMiner Linked Open Data Extension [39], are techniques for automatically enriching data with features derived from multiple Linked Open Data sources without the need to specify SPARQL queries. However, in contrast to AUTORDF2GML, their main purpose is to *enrich* an existing dataset with relevant features instead of transforming RDF data into a graph dataset for graph machine learning.

Literal2Feature [3] is a framework to automatically transform RDF data into a standard feature matrix by traversing the RDF graph. It starts with a set of entities of interest and automatically retrieves literals to a pre-configured walk length to build the feature matrix. For generating the feature vectors, only the literals are used. Literal2Feature is mainly used to obtain Spark DataFrames as input for conventional machine learning models [12, 33]. In contrast to AUTORDF2GML, no graph structure is used for feature generation.

In summary, there is currently no approach available for transforming RDF data into a propositionalized format, i.e., feature matrix, that considers both RDF data type and object properties. Our proposed framework AUTORDF2GML allows the representation of nodes and edges with their corresponding features as vectors. In addition, as with the other automatic approaches, the feature selection and transformation is performed automatically without requiring the user to define any SPARQL queries. The user only needs to define the key aspect of the desired GML datasets in a configuration file (e.g., the node and edge types), and the rest of the processes are handled automatically.

*Knowledge Graph Embeddings.* In addition to classical propositionalization methods, knowledge graph embeddings offer an approach to convert entities into dense vector representations. For instance, RDF2Vec [41] transforms RDF graphs into graph random walks and Weisfeiler-Lehman graph kernels, and further applies CBOW and Skip-gram models to learn latent entity representations based on the knowledge graph topology. Other graph embedding techniques

includes *TransE* [6], a translation distance model, *DistMult* [53] and *ComplEx* [46], semantic matching models, and *RotateE* [44], a rotation model in the complex embedding space. However, these techniques neglect literals in their learning process, or only consider the literals without the topological information [3, 18].

To sum up, existing frameworks to convert RDF data for graph machine learning lack an efficient method to transform the RDF data into a propositional format considering both the RDF data types and object properties of instances. AUTORDF2GML addresses this by facilitating the integration and utilization of both content-based (RDF data types) and topology-based (RDF object properties) embeddings, described in detail in Sect. 3.

**Table 2.** Overview of heterogeneous graph benchmark datasets, sorted by the variety of available semantic node features.

Heterogeneous Graph Benchmark Datasets	Nodes			Node Feature Characteristics			Edges			Domain	Common Task
	#Nodes	#Nodes Types	#Node Types w. Features	NLD	Literals\ NLD	Topology	#Edges	#Edge Types	#Edge Types w. Features		
AMiner [11]	4,891,819	3	0	x	x	x	12,518,144	2	0	Academia	Node level
MovieLens [20]	10,352	2	0	x	x	x	100,836	1	1	Entertainment	Edge level
LastFM [17]	20,612	3	0	x	x	x	128,804	3	0	Entertainment	Edge level
HGB_Freebase [36]	180,098	8	0	x	x	x	1,057,688	36	0	Multi Domain	Node level
HGB_LastFM [36]	20,612	3	0	x	x	x	141,521	3	0	Entertainment	Edge level
Douban [56]	6,000	2	0	x	x	x	136,891	1	1	E-commerce	Edge level
Flixster [56]	6,000	2	0	x	x	x	26,173	1	1	E-commerce	Edge level
Yahoo-Music [56]	6,000	2	0	x	x	x	5,335	1	1	E-commerce	Edge level
AmazonBook [23]	144,242	2	0	x	x	x	2,984,108	1	0	E-commerce	Edge level
OGB_MAG [25]	1,939,743	4	1	✓	x	x	21,111,007	4	0	Academia	Node level
DBLP [17]	26,128	4	1	✓	x	x	296,563	3	0	Academia	Node level
IMDB [17]	11,616	3	1	✓	x	x	17,106	2	0	Entertainment	Node level
HGB_DBLP [36]	26,128	4	3	✓	x	x	239,566	6	0	Academia	Node level
HGB_ACM [36]	10,942	4	3	✓	x	x	547,872	8	0	Academia	Node level
HGB_PubMed [36]	63,109	4	4	✓	x	x	244,986	10	0	Academia	Edge level
HGB_Amazon [36]	10,099	1	1	✓	✓	x	148,659	2	0	E-commerce	Edge level
HGB_IMDB [36]	21,420	4	3	✓	✓	x	86,642	6	0	Entertainment	Node level
H&M [24]	1,477,522	2	2	✓	✓	x	31,788,324	1	1	E-commerce	Edge level
SOA-SW	170,966	6	6	✓	✓	✓	1,856,146	7	3	Academia	Edge level
LPWC	391,247	4	4	✓	✓	✓	990,303	6	0	Academia	Edge level
AIFB	2,260	3	1	✓	✓	x	4,154	2	0	Academia	Edge level
LinkedMDB	383,290	8	8	✓	✓	x	718,675	7	0	Entertainment	Edge level

## 2.2 Heterogeneous Graph Benchmarks

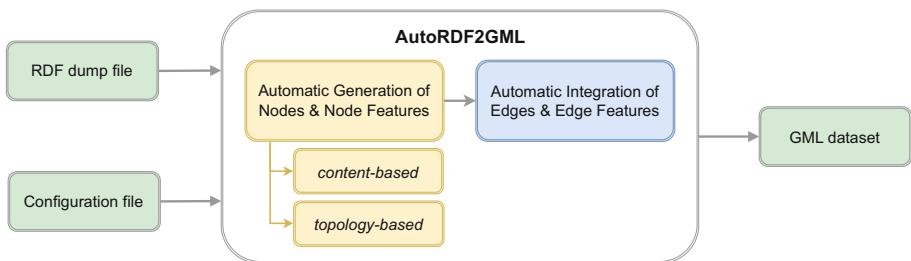
Several benchmarks for tasks on heterogeneous graphs (i.e., graphs with several node types) have been proposed (see Table 2). Our comparison includes all heterogeneous graph benchmark datasets provided by PyTorch Geometric, including heterogeneous graph benchmarks from Open Graph Benchmark (OGB) and Heterogeneous Graph Benchmark (HGB) [16, 25, 36]. For the purpose of our analysis, we categorize node features into three distinct groups: (1) The first

category encompasses *natural language description (NLD)* features. These are content-based features that are derived from textual descriptions given in natural language (e.g., label or description). (2) The second category is referred to as *content-based other Literals*, denoted as *Literals\NLD*. This group includes content attributes that are not natural language descriptions, such as numeric, categorical, or boolean values. Together, both *NLD* features and *Literals\NLD* constitute the broader set of *content-based features (literals)*. (3) The third category diverges from content-based attributes and is focused on the graph structure, i.e., *topology-based* features.

From Table 2, it becomes apparent that existing graph benchmarks offer either content-based or topology-based node features across all node types, but not both. This means that most benchmark datasets focus on the heterogeneity of graph structures instead of the diversity of the node features. The benchmarks also seldom provide node features for all node types, and when they do, the generation of these features largely depends on the inherent natural language description (NLD) of the elements.

Furthermore, existing benchmarks do not include separately evaluated topology-based features, which is a remarkable oversight. This is particularly relevant when new graph neural network architectures are developed, as they often calculate topology-based features on-the-fly during evaluations. Thus, it remains unclear whether the performance is a result of advancements in the graph neural network model itself, or due to the optimized feature engineering through the topology-based node features [35]. The issues with the current benchmarks thus motivated us to construct new benchmarks datasets with our proposed AUTORDF2GML framework. We created *SOA-SW*, *LPWC*, *AIFB*, and *LinkedMDB* based on publicly-available RDF knowledge graphs (see Sect. 4).

### 3 AutoRDF2GML



**Fig. 1.** Overview of AUTORDF2GML.

In this section, we present our new framework AUTORDF2GML that seamlessly transforms RDF data into data representations for graph machine learning tasks.

The generated data representations contain numeric vector features represented in feature matrices as node features. The features can be derived from content-based or topology-based information in the underlying RDF data. A notable strength of the framework is its ability to automatically select and transform the content-based features. This enable the users, even those rather unfamiliar with RDF and SPARQL, to utilize RDF data in a straightforward manner. The user experience is further enhanced by a user-centric setup: users are only required to define the RDF classes and properties for node and edge transformation, eliminating the need for complex specifications of SPARQL queries.

Figure 1 provides an overview of AUTORDF2GML. First, the user supplies an RDF dump file and configuration file, specifying the RDF classes and properties for feature construction. AUTORDF2GML uses the *rdflib* Python package, thus it supports all common RDF dump formats (e.g., Turtle, N-Triples, JSON-LD). Next, nodes are extracted from the RDF data, and their features are automatically generated based on either content-based or topology-based semantic information. Edges between the nodes are then automatically formed, completing the graph structure integration. The output of AUTORDF2GML is a ready-to-use heterogeneous graph machine learning dataset compatible with graph machine learning frameworks such as PyTorch Geometric [16] and DGL [48].

In the following, we outline the two main steps of AUTORDF2GML: (1) Automatic Generation of Nodes and Node Features in Sect. 3.1, and (2) Automatic Integration of Edges and Edge Features in Sect. 3.2.

### 3.1 Automatic Generation of Nodes and Node Features

In RDF data, entities belong to specific classes and are uniquely identified by URIs [37]. Given the relevant classes specified in the configuration file, all corresponding entities are extracted to represent the nodes in the resulting graph dataset. This step is necessary for isolating the relevant classes for a specific use-case (e.g., recommendation). Subsequently, AUTORDF2GML provides two approaches to compute the node features: (1) content-based node features, and (2) topology-based node features, outlined in the following.

#### 3.1.1 Content-Based Node Features.

After identifying the relevant entities and their corresponding URIs, AUTORDF2GML can generate features using RDF datatype properties. RDF datatype properties link entities to specific types of data, known as literals. These literals hold valuable information about the entity and can serve as important input features for machine learning models. Within its architecture, AUTORDF2GML includes an automatic module tailored to the construction of numeric node features based on available RDF datatype properties. The automatic transformation of RDF datatype properties and their associated literal values into usable vectorized features includes the automatic feature selection and transformation, as outlined in the following:

**a) Automatic Feature Selection:** Automating the feature selection is necessary because RDF data typically contains a huge number of datatype properties. A manual analysis and evaluation of all datatype properties is time consuming, and especially challenging for data scientists from other disciplines. In addition, feature selection based on RDF datatype properties is a complex task that requires addressing the following challenges:

1. *Property Sparsity*: The filling degree of some datatype properties can be extremely sparse.
2. *Identicality and Uniqueness of Values*: Datatype properties can include predominantly identical values or, conversely, be characterized by completely unique entries for each entity.
3. *Redundancy*: Different properties can sometimes reflect similar information patterns, resulting in high correlation between properties.

We do not consider the features with any of the above listed characteristics in feature selection because they distort the underlying data dynamics, lack the necessary variance, or pose a risk of overfitting due to redundant information [37]. Therefore, it is necessary to pre-process the available datatype properties for feature generation and only select datatype properties that do not break into any of the mentioned characteristics. The discarding of properties with unique values is only applied to nominal features that are not an NLD [37]. If the values of certain selected properties strongly correlate with each other (based on the Pearson correlation score), one of them is discarded [21].

**b) Automatic Feature Transformation:** After the relevant features, i.e., the relevant datatype properties, are selected, they need to be transformed into a numeric vector representation to build the node features. AUTORDF2GML distinguishes between 6 literal types and their associate transformation rules (see Table 3). Strings that are natural language descriptions (NLD) are encoded using a text encoder (e.g., a language model like BERT [10] or SciBERT [4]), following a common practice to generate node features [25, 26, 57]. Categorical values are either one-hot encoded or label encoded depending on the number of unique values. Numeric values and years are normalized. Boolean values are label encoded with 0 and 1. For dates, the unix timestamp is calculated and then normalized. In general, we adhere to the established machine learning feature transformation techniques as outlined in previous works [37, 42]. Finally, the features are normalized into a standard range, and *not-a-number* (NaN) values are filled with their mean, following [21].

### 3.1.2 Topology-Based Node Features.

Another approach for generating features for RDF graphs is to leverage the topological information of the underlying RDF data.<sup>1</sup> This is because the structure and relationships of the RDF object properties (e.g., relations to other entities)

---

<sup>1</sup> In case both topology-based and content-based features are needed, one can run both settings and combine the features.

**Table 3.** Transformation methods of AUTORDF2GML.

Literal Type	Encoding Method
String (NLD)	Text Encoder
String (Categorical)	One-Hot-Encoder Label Encoder & Normalization
Numerical value	Normalization
Boolean	Label Encoder (0/1)
Year	Normalization
Date	Unix Timestamp & Normalization

provide a rich semantic information about the data. Topology-based node features are particularly flexible because they retain the complete semantic information from the topological structure, even when using a subgraph with only a small subset of the RDF classes. One widely used approach to obtaining the topology-based representation are knowledge graph embedding techniques. Knowledge graph embedding models such as TransE [6], DistMult [53], ComplEx [46], and RotatE [44] have gained great popularity in recent years due to their effectiveness in representing RDF entities as encoded feature vectors [3, 14, 28, 43]. Following that, AUTORDF2GML automatically computes the topology-based node feature vectors using these widely recognized knowledge graph embedding techniques.

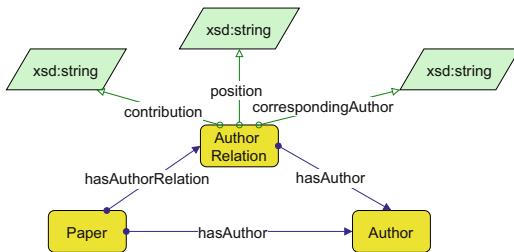
### 3.2 Automatic Integration of Edges and Edge Features

To represent a complete graph structure in our data representation for Graph Machine Learning, we need to construct the edges. The edges of the transformed graph are based on RDF object properties. RDF object properties are directed relations that link two entities [50]. Since RDF knowledge graphs may contain several object properties sharing the same range and domain and have similar semantic meanings (e.g. two properties linked with `owl:equivalentProperty` [34]), these properties might need to be mapped to the same edge type. Thus, AUTORDF2GML enables defining a list of RDF object properties that can be mapped to the same edge type. In the following, we describe four types of RDF object property patterns and how they are used within AUTORDF2GML.

**(a) Binary Relations:** In RDF, an object property is a binary relation that links two entities. AUTORDF2GML creates an edge list for each edge type based on the specified RDF object properties in the configuration file. Although the possible subjects (i.e., start nodes) and objects (i.e., end nodes) of a relation can be defined via `rdfs:range` and `rdfs:domain` in the ontology [51], our framework does not depend on this information, as we do not perform any OWL reasoning.

**(b) N-ary Relations:** N-ary relations [19, 49] are used when a simple binary relationship between two entities is insufficient, for instance when we need to model the certainty, strength, or relevance of the relationship, as well.

The ontological pattern for modeling additional attributes that describe a relationship involves introducing a new auxiliary class. This class is linked between the subject and the object of the relationship containing additional attributes with information about the relationship [19, 49]. Figure 2 shows an example of n-ary relation between *Paper* and *Author*, where the auxiliary class *AuthorRelation* contains additional information about the author's *contribution* (e.g., conceptualization, supervision, or software), the author's *position* (e.g., first author, middle author or last author), and if the author is the *corresponding author*.



**Fig. 2.** Example n-ary relation.

The datatype properties of the auxiliary classes that contain information about the relationships are used as edge features in the transformed heterogeneous graph dataset. If the datatype properties do not contain numerical values, they are either one-hot encoded or label encoded.

**(c) Multi-hop Relations:** In RDF, two entities can be interconnected through a chain of object properties (see, e.g., Fig. 3). To represent these chained connections directly, edges can be formed between entities that are connected across multiple object properties. This approach simplifies the representation and connects entities that might be several hops apart in the original RDF data [49]. In addition, this further enables new use cases, such as link prediction across multiple properties in the underlying RDF data. Figure 3 shows an example multi-hop relation from the real-world RDF knowledge graph *Linked Papers With Code* [13] where *Dataset* and *Method* are connected through the properties *hasPaper* and *hasMethod*. Given such property chain, a new edge directly connecting *Dataset* and *Method* is created, allowing the recommendation of methods for a specific dataset.



**Fig. 3.** Example multi-hop relation from Linked Papers With Code.

**(d) Custom Relations:** RDF data can also contain indirect relations between classes that cannot be extracted by linear graph traveling. To extract such complex and non-linear relations and map them as explicit edges in the transformed graph, SPARQL queries can be used that explicitly define the relation.

## 4 Semantic Graph Machine Learning Benchmarks

In the following, we present how to apply our framework to large RDF knowledge graphs, considering the semantic features of the knowledge graphs. We provide the resulting Graph Machine Learning datasets publicly available for the community as benchmarks (see links on page 1).

### 4.1 SemOpenAlex-SemanticWeb (SOA-SW)

SemOpenAlex [14] is a vast RDF dataset containing over 26 billion RDF triples that describe 249 million publications from various academic disciplines. It features a rich schema and is interconnected with other Linked Open Data sources.

**SOA-SW Knowledge Graph Curation.** In line with previous research on dataset curation for graph machine learning, we derive a subgraph of SemOpenAlex [14] based on specific filter rules to create a basis for a graph benchmark dataset for GNN-based recommendations [8,9].<sup>2</sup> To ensure its validity, SOA-SW contains only SemOpenAlex entities that meet the following conditions: (1) Every author included has at least one semantic web paper published and between 3 and 200 papers published in total. (2) From these authors, only papers with an abstract, publication year  $\geq 2005$  and citation count  $\geq 10$  are included. We exclude authors whose papers do not meet these requirements.

SOA-SW based on the SemOpenAlex version from 2023-04-24, consists of 21,978,026 RDF triples. Table 4 shows the number of included entities in SOA-SW. SOA-SW includes the comprehensive semantic information about these entities as defined in the rich SemOpenAlex ontology, covering 13 entity types, including the entity types `works`, `authors`, `institutions`, `sources`, `publishers` and `concepts`, as well as 87 semantic relation types [14].

**Benchmark Creation.** For creating the benchmark, we use the SOA-SW data dump as input for AUTORDF2GML. We also add a custom relation to model the *co-author* relations directly in the transformed graph (see GitHub). They are not directly included in the underlying RDF data, but can be retrieved using a SPARQL query. Modeling this relationship directly in the data allows to consider a new use case like collaboration recommendation.

We construct both content-based and topology-based node features for the benchmark. For the content-based node features, the relevant data type properties are automatically selected and subsequently transformed. From all available

---

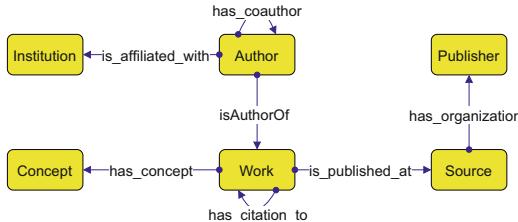
<sup>2</sup> See <https://github.com/davidlamprecht/semopenalex-semanticweb>.

data type properties, the following are selected: (a) **Work** (8 out of 18 data type properties), (b) **Author** (4 out of 10 data type properties), (c) **Concept** (4 out of 11 data type properties), (d) **Source** (10 out of 19 data type properties), (e) **Institution** (5 out of 14 data type properties), and **Publisher** (5 out of 12 data type properties). Furthermore, for *work-author* edges, the data type values of the property **soa:position** and for *work-concept* edges, the data type values of the property **soa:score** are used as edge features.

AutoRDF2GML detects NLD features for the *work* nodes, specifically the properties **dcterms:title** and **dcterms:abstract**. The work titles and abstracts are concatenated, and subsequently, a 128-dimensional embedding is generated for this combined data using SciBERT embeddings [4].

For the topology-based feature generation, we apply TransE [6], since it yields the best results in embedding evaluation (summarized in our repository). Using the entire SOA-SW RDF dump, AutoRDF2GML computes 128-dimensional embeddings for all nodes, utilizing all available data for the training process.

**Heterogeneous Graph Dataset SOA-SW.** Fig. 4 shows an overview of the schema of the created heterogeneous graph dataset based on SOA-SW KG, including in total 6 different node types and 7 different edge types. An overview of the number of nodes and the availability of different categories of node features for them is summarized in Table 4. Table 5 presents the number of edges by edge type and indicates whether they include features. It can be seen that the transformed heterogeneous graph dataset has rich node features capturing different kinds of semantics that is available in the raw RDF data. The semantic node features can then be used for GNN-based machine learning tasks, such as link prediction.



**Fig. 4.** Overview heterogeneous graph dataset SOA-SW.

## 4.2 Linked Papers With Code (LPWC)

*Linked Papers With Code (LPWC)* [13] is an RDF knowledge graph that provides extensive information on approximately 400,000 publications in the Machine Learning field. It includes details on the tasks addressed, datasets used, methods implemented, and evaluations conducted, along with their results. We use

**Table 4.** Node types, instance counts, and feature availability in *SOA-SW*.

Node Type	# Instances	Literals\ NLD	NLD	TransE
Work	95,575	✓	✓	✓
Author	19,970	✓	✗	✓
Concept	38,050	✓	✗	✓
Source	10,739	✓	✗	✓
Institution	5,846	✓	✗	✓
Publisher	786	✓	✗	✓

**Table 5.** Edge types, instance counts, and feature availability in *SOA-SW*.

Edge Type	# Instances	Features
work-concept	1,320,949	✓
work-source	247,667	✗
work-work	115,271	✗
author-work	112,565	✓
author-author	38,632	✓
author-institution	19,281	✗
source-publisher	1,781	✗

**Table 6.** Node types, instance counts, and feature availability in *LPWC*.

Node Type	# Instances	Literals\ NLD	NLD	TransE
Paper	376,557	✓	✓	✓
Dataset	8,322	✓	✓	✓
Task	4,267	✓	✓	✓
Method	2,101	✓	✓	✓

**Table 7.** Edge types and instance counts in *LPWC*.

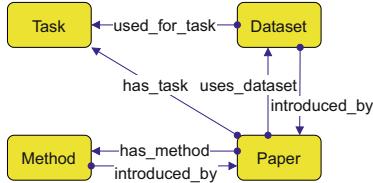
Edge Type	# Instances
paper-task	589,784
paper-method	362,918
paper-dataset	15,520
dataset-task	14,343
dataset-paper	5,838
method-paper	1,900

the AUTORDF2GML framework to transform *LPWC* into a Graph Machine Learning dataset. We construct both content-based and topology-based node features. For the content-based features the following data type properties are selected: (a) **Paper** (3 out of 7 data type properties selected), (b) **Method** (4 out of 7 data type properties selected), (c) **Task** (2 out of 2 data type properties selected), (d) **Dataset** (7 out of 10 data type properties selected).

AUTORDF2GML detects NLD features for all node types. The detected NLD features are concatenated, and a 128-dimensional embedding is calculated for the combined data with SciBERT [4]. For the topology-based features, again we use TransE [6], since it gives the best results in the embedding for LPWC [13]. Using the entire LPWC RDF dump, AUTORDF2GML computes 128-dimensional embeddings for all nodes, with all available training data.

**Heterogeneous Graph Dataset LPWC.** Fig. 5 shows an overview of the schema of the created heterogeneous graph dataset based on LPWC, including the nodes and the edges. In total it is composed of 4 different nodes types and 6 different edge types. Table 6 gives an overview of the number of nodes and the presence of different categories of node characteristics for them. Remarkably, all node types of LPWC have node features from all three categories (Literals\NLD, NLD and topology). This allows for detailed analyses of the impact of semantic node features on the performance of GNN-based machine learning tasks, such

as recommendation tasks. An overview of the number of edges of the different edge types is shown in Table 7.



**Fig. 5.** Overview heterogeneous graph dataset LPWC.

### 4.3 Further Benchmark Datasets

We applied AUTORDF2GML to two other RDF knowledge graphs to show its applicability across various settings and domains. The resulting benchmarks (see page 1) were created from the RDF knowledge graphs **AIFB** [5], a commonly used dataset for reasoning tasks, and **LinkedMDB** [22], the RDF version of IMDb, covering *movies* and related entities such as *actors* and *directors*.

## 5 Applications and Use Cases

In this section, we outline the impact and use case examples of our framework, demonstrating its utility in both academic research and industry applications.

**Enhanced Accessibility and User-Friendliness for Semantic Data.** AUTORDF2GML enables individuals, including both established researchers and newcomers unfamiliar with RDF(S) and SPARQL, to leverage semantic web data without the need for SPARQL queries. This includes a significant number of researchers in the core Machine Learning community, including those focused on areas such as Graph Neural Networks, as well as those involved in explainable AI (XAI) and human-computer interaction (HCI). In the industry, data scientists represent a major user group for our framework, reflecting the growing demand for AI expertise.

**Increasing Use of RDF Knowledge Graphs and Linked Open Data.** The availability of RDF knowledge graphs, particularly in the Linked Open Data cloud, is increasing across various sectors such as e-commerce, academia, and entertainment. AUTORDF2GML supports RDF knowledge graphs without being constrained by any schema restrictions from OWL files or RDFS. We have established benchmarks in these domains as well, allowing for a systematic evaluation of systems such as recommender systems for products [27], scientific papers [2], datasets [15], and movies [29]. These examples not only

demonstrate the availability of knowledge graphs but also an industry demand for such resources.

**Scalability and Big Data Benchmarking.** So far, most benchmarks have been considerably small, often consisting of only a few thousand nodes and edges (see Table 2). In contrast, AUTORDF2GML has been applied to large RDF knowledge graphs, such as LPWC with 8 million RDF triples and LinkedMDB with 6.1 million RDF triples. These benchmarks, along with others easily generated using AUTORDF2GML, are crucial for advancing the field and providing standardized datasets for real-world Machine Learning applications. There is a particular need for large benchmarks that include both content (node features) and structural information (topological features) to enhance AI-based systems.

**Enhancing Recommendation Systems.** Graph Machine Learning datasets are increasingly used in various applications, including deep learning-based search and recommender systems. Unlike systems limited to topological data and, thus, collaborative filtering approaches, the graph datasets we have developed enable more precise and higher-performing systems. Initial evaluations of recommender systems using heterogeneous graph neural networks and datasets generated with AUTORDF2GML have demonstrated an improvement in F1 score (see our GitHub repository). In addition, knowledge graph-based recommender systems, as summarized in [52], offer several benefits. For instance, the rich semantic relationships among items in knowledge graphs helps improving item representation [47], and further enhances the interpretability of the recommendation results [55].

**Foundation for Neurosymbolic AI.** AUTORDF2GML is well positioned to contribute to the field of neurosymbolic AI and language models. While large language models (LLMs) have been widely developed and utilized, they come with limitations such as knowledge cutoffs and significant hardware requirements. An emerging alternative involves leveraging language models, such as BERT [10] and T5 [38] integrated with knowledge graphs [1, 45, 54]—an approach sometimes referred as *knowledge-guided* language models. For instance, [45] introduces approach using smaller LMs combined with KGs that achieve results comparable to or even surpass those of LLM-based methods. Such approach provides capabilities in explaining the model outputs, as well, such as in recommendation systems [55], by linking to KGs as explicit knowledge representations.

## 6 Conclusion

In this paper, we introduced AUTORDF2GML, a novel framework designed to efficiently convert RDF data into benchmarks tailored for graph-based machine learning applications, potentially bridging the gap between the Graph Machine Learning and Semantic Web communities. AUTORDF2GML framework is characterized by its modular design, automated feature extraction, and one-file configuration design, making it accessible even to users who may not be familiar with semantic technologies such as SPARQL. Furthermore, we demonstrated

the utility of AUTO RDF2GML by applying it to large RDF knowledge graphs, successfully transforming them into heterogeneous graph datasets, each enriched with unique semantic features.

In the future, we plan to enhance our framework to operate across multiple RDF knowledge graphs within the Linked Open Data cloud in parallel and to incorporate reasoning through OWL concepts. This enhancement will include mechanisms for handling ontological relationships across different knowledge graphs, such as `equivalentClass` links.

**Resource Availability Statement:** All resources are accessible through the URLs provided on page 1.

## References

1. Baek, J., Aji, A.F., Saffari, A.: Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In: Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations, pp. 78–106. NLRSE 2023, Toronto, Canada
2. Bai, X., Wang, M., Lee, I., Yang, Z., Kong, X., Xia, F.: Scientific paper recommendation: a survey. *IEEE Access* **7**, 9324–9339 (2019)
3. Bakhshandegan Moghaddam, F., Draschner, C., Lehmann, J., Jabeen, H.: Literal2Feature: an automatic scalable RDF graph feature extractor. In: Further with Knowledge Graphs, pp. 74–88. IOS Press (2021)
4. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 3613–3618. EMNLP-IJCNLP 2019 (2019)
5. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 58–71. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-76298-0\\_5](https://doi.org/10.1007/978-3-540-76298-0_5)
6. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems, pp. 2787–2795. NIPS 2013 (2013)
7. Cheng, W., Kasneci, G., Graepel, T., Stern, D.H., Herbrich, R.: Automated feature generation from structured knowledge. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, pp. 1395–1404. CIKM 2011 (2011)
8. Chennupati, N.: Recommending collaborations using link prediction. Ph.D. thesis, Wright State University (2021)
9. Chuan, P.M., Son, L.H., Ali, M., Khang, T.D., Huong, L.T., Dey, N.: Link prediction in co-authorship networks based on hybrid content similarity metric. *Appl. Intell.* **48**, 2470–2486 (2018)
10. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186. NAACL-HLT 2019 (2019)

11. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144. KDD 2017 (2017)
12. Draschner, C.F., Stadler, C., Moghaddam, F.B., Lehmann, J., Jabeen, H.: DistRDF2ML - scalable distributed in-memory machine learning pipelines for RDF knowledge graphs. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, pp. 4465–4474. CIKM 2021 (2021)
13. Färber, M., Lamprecht, D.: Linked papers with code: the latest in machine learning as an RDF knowledge graph. In: Proceedings of the 22nd International Semantic Web Conference. ISWC 2023 (2023)
14. Färber, M., Lamprecht, D., Krause, J., Aung, L., Haase, P.: SemOpenAlex: the scientific landscape in 26 billion RDF triples. In: Proceedings of the 22nd International Semantic Web Conference, pp. 94–112. ISWC 2023 (2023)
15. Färber, M., Leisinger, A.: Recommending datasets for scientific problem descriptions. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, pp. 3014–3018. CIKM 2021 (2021)
16. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch geometric. arXiv preprint [arXiv:1903.02428](https://arxiv.org/abs/1903.02428) (2019)
17. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of the Web Conference, pp. 2331–2341. WWW 2020 (2020)
18. Gesese, G.A., Biswas, R., Alam, M., Sack, H.: A survey on knowledge graph embeddings with literals: which model links better literal-ly? Semantic Web **12**(4), 617–647 (2021)
19. Giunti, M., Sergioli, G., Vivanet, G., Pinna, S.: Representing N-ary relations in the Semantic Web. Logic J. IGPL **29**(4), 697–717 (2021)
20. GroupLens research: MovieLens (2023). <https://movielens.org>, accessed on: 29.09.2023
21. Guyon, I., Elisseeff, A.: An introduction to feature extraction. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (eds.) Feature Extraction: Foundations and Applications, vol. 207, pp. 1–25. Springer, Heidelberg (2006). [https://doi.org/10.1007/978-3-540-35488-8\\_1](https://doi.org/10.1007/978-3-540-35488-8_1)
22. Hassanzadeh, O., Consens, M.P.: Linked movie data base. In: Proceedings of the WWW2009 Workshop on Linked Data on the Web. LDOW 2009 (2009)
23. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 639–648. SIGIR 2020 (2020)
24. H&M Group: H&M Personalized Fashion Recommendations. <https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations> (2023). Accessed 29 Sept 2023
25. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs (2020)
26. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. In: Proceedings of the Web Conference, pp. 2704–2710. WWW 2020 (2020)
27. Huang, Z., Chung, W., Chen, H.: A graph model for E-commerce recommender systems. J. Am. Soc. Inform. Sci. Technol. **55**(3), 259–274 (2004)
28. Hubert, N., Monnin, P., Brun, A., Monticolo, D.: Knowledge graph embeddings for link prediction: beware of semantics! In: Proceedings of the Workshop on Deep Learning for Knowledge Graphs. DL4KG@ISWC 2022 (2022)

29. Jung, J.J.: Attribute selection-based recommendation framework for short-head user group: an empirical study by MovieLens and IMDB. *Expert Syst. Appl.* **39**(4), 4049–4054 (2012)
30. Kappara, V.N.P., Ichise, R., Vyas, O.P.: LiDDM: a data mining system for linked data. In: Proceedings of the WWW2011 Workshop on Linked Data on the Web. WWW 2011 (2011)
31. Khan, M.A., Grimes, G.A., Dengel, A.: Two pre-processing operators for improved learning from Semantic Web data. In: Proceedings of the First Rapid-Miner Community Meeting And Conference. RCOMM 2010 (2010)
32. Lavrač, N., Škrlj, B., Robnik-Šikonja, M.: Propositionalization and embeddings: two sides of the same coin. *Mach. Learn.* **109**, 1465–1507 (2020)
33. Lehmann, J., et al.: Distributed semantic analytics using the SANSA stack. In: Proceedings of the 16th International Semantic Web Conference, pp. 147–155. ISWC 2017 (2017)
34. Li, C., Ling, T.W.: OWL-based semantic conflicts detection and resolution for data interoperability. In: Proceedings of the ER 2004 Workshops, pp. 266–277 (2004)
35. Li, C., Guo, Z., He, Q., Xu, H., He, K.: Long-range dependency based multi-layer perceptron for heterogeneous information networks. arXiv preprint [arXiv:2307.08430](https://arxiv.org/abs/2307.08430) (2023)
36. Lv, Q., et al.: Are we really making much progress?: revisiting, benchmarking and refining heterogeneous graph neural networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1150–1160. KDD 2021 (2021)
37. Paulheim, H., Fürnkranz, J.: Unsupervised generation of data mining features from linked open data. In: Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, pp. 31:1–31:12. WIMS 2012 (2012)
38. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**(140), 1–67 (2020). <http://jmlr.org/papers/v21/20-074.html>
39. Ristoski, P., Bizer, C., Paulheim, H.: Mining the Web of linked data with Rapid-Miner. *J. Web Semant.* **35**, 142–151 (2015)
40. Ristoski, P., Paulheim, H.: A comparison of propositionalization strategies for creating features from linked open data. In: Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. LDKD@ECML-PKDD 2014 (2014)
41. Ristoski, P., Paulheim, H.: RDF2Vec: RDF graph embeddings for data mining. In: Proceedings of the 15th International Semantic Web Conference, pp. 498–514. ISWC 2016 (2016)
42. Ristoski, P., Paulheim, H.: Semantic Web in data mining and knowledge discovery: a comprehensive survey. *J. Web Semant.* **36**, 1–22 (2016)
43. Ruffinelli, D., Broscheit, S., Gemulla, R.: You CAN teach an old dog new tricks! on training knowledge graph embeddings. In: Proceedings of the 8th International Conference on Learning Representations. ICLR 2020 (2020)
44. Sun, Z., Deng, Z., Nie, J., Tang, J.: RotateE: knowledge graph embedding by relational rotation in complex space. In: Proceedings of the 7th International Conference on Learning Representations. ICLR 2019 (2019)
45. Susanti, Y., Färber, M.: Knowledge graph structure as prompt: improving small language models capabilities for knowledge-based causal discovery. In: Proceedings of the 2024 International Semantic Web Conference. ISWC 2024 (2024)

46. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on Machine Learning. ICML 2016, vol. 48, pp. 2071–2080 (2016)
47. Wang, H., et al.: RippleNet: propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. CIKM 2018 (2018)
48. Wang, M., et al.: Deep graph library: towards efficient and scalable deep learning on graphs. arXiv preprint [arXiv:1909.01315](https://arxiv.org/abs/1909.01315) (2019)
49. World Wide Web Consortium: defining N-ary relations on the semantic web (2006). <https://www.w3.org/TR/swbp-n-aryRelations/>
50. World Wide Web Consortium: RDF 1.1 concepts and abstract syntax (2014). <https://www.w3.org/TR/rdf11-concepts/>
51. World Wide Web Consortium: RDF schema 1.1 (2014). <https://www.w3.org/TR/rdf-schema/>
52. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. ACM Comput. Surv. **55**(5), 1–37 (2022)
53. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the 3rd International Conference on Learning Representations. ICLR 2015 (2015)
54. Yang, L., Chen, H., Li, Z., Ding, X., Wu, X.: ChatGPT is not enough: enhancing large language models with knowledge graphs for fact-aware language modeling. arXiv preprint [arXiv:2306.11489](https://arxiv.org/abs/2306.11489) (2023)
55. Yang, Z., Dong, S.: HAGERec: hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. Knowl. Based Syst. **204**, 106194 (2020)
56. Zhang, M., Chen, Y.: Inductive matrix completion based on graph neural networks. In: Proceedings of the 8th International Conference on Learning Representations. ICLR 2020 (2020)
57. Zhu, J., Yaseen, A.: A recommender for research collaborators using graph neural networks. Front. Artif. Intell. **5**, 881704 (2022)