Bridging RDF Knowledge Graphs with Graph Neural Networks for Semantically-Rich Recommender Systems

Michael Färber ✉[1], David Lamprecht[2], and Yuni Susanti[3]

[1] ScaDS.AI & TU Dresden, Dresden, Germany
michael.faerber@tu-dresden.de
[2] metaphacts GmbH, Walldorf, Germany
dl@metaphacts.com
[3] Fujitsu Ltd., Japan
susanti.yuni@fujitsu.com

Abstract. Graph Neural Networks (GNNs) have substantially advanced the field of recommender systems. However, despite the creation of more than a thousand knowledge graphs (KGs) under the W3C standard RDF, their rich semantic information has not yet been fully leveraged in GNN-based recommender systems. To address this gap, we propose a comprehensive integration of RDF KGs with GNNs that utilizes both the topological information from RDF *object* properties and the content information from RDF *datatype* properties. Our main focus is an in-depth evaluation of various GNNs, analyzing how different semantic feature initializations and types of graph structure heterogeneity influence their performance in recommendation tasks. Through experiments across multiple recommendation scenarios involving multi-million-node RDF graphs, we demonstrate that harnessing the semantic richness of RDF KGs significantly improves recommender systems and lays the groundwork for GNN-based recommender systems for the Linked Open Data cloud. The code and data are available on our GitHub repository.[4]

Keywords: rdf · knowledge graph · recommender system

1 Introduction

Recommender systems have become essential tools for alleviating information overload and enhancing user experience in a wide range of applications [17,39]. Graph Neural Network (GNN)-based recommendation methods have gained significant attention due to their capability to effectively process structured data and leverage high-order information [14]. Research has shown that GNN-based models consistently outperform traditional approaches across various benchmark datasets [39], highlighting their growing importance in the field.

[4] https://github.com/davidlamprecht/rdf-gnn-recommendation

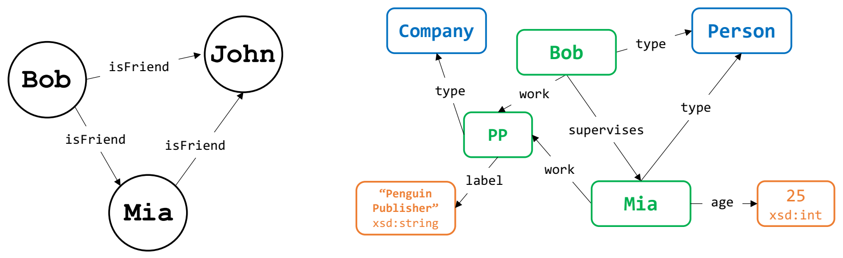

Fig. 1: Illustration of homogeneous (left) vs. heterogeneous (right) graphs (RDF).

However, despite the growing prominence of GNN-based methods in recommender systems, their application has largely focused on *homogeneous* graphs and user-item recommendation scenarios. In a homogeneous graph (see Figure 1), all nodes are of the same type, and all edges represent the same kind of relationship. For instance, in a *social network* graph, nodes represent individuals, and edges represent their connections. Other examples include *product* graphs and *citation networks* [39,49,20]. In contrast, a *heterogeneous* graph, such as knowledge graphs (KGs), includes diverse types of nodes and edges. KGs are typically represented as sets of triples using standard formats like the Resource Description Framework (RDF) [27]. RDF is a W3C standard that serves as the basis for more than thousand knowledge graphs in the Linked Open Data cloud [29], including Wikidata, YAGO, and DBpedia [8], as well as domain-specific ones like SemOpenAlex [10], STRING [31] and ChEMBL [46] in the biomedical domain. These graphs often contain millions or billions of entities and relationships and thus far exceed the order of magnitude that is usually used in evaluation benchmarks for GNNs. Furthermore, these graphs contains not only multiple entity and relation types, but also RDF *datatype* properties (e.g., `xsd:string` in Figure 1). In addition, the rich semantic relationships among nodes in KGs-based recommender systems enhance node representation [34] and improve the interpretability of recommendation results [43].

When dealing with heterogeneous graphs such as RDF KGs, existing approaches often fall short of fully leveraging their semantic potential due to their complex structure with multi-type entities and relations [39]. Thus, current GNN methods struggle to fully process and utilize the comprehensive *semantics* of RDF, which includes the graph topological information and content-based information derived from various types of literals. To address these challenges, we present a GNN-RDF-based recommendation system leveraging both the RDF's topological and content-based information as semantically-rich features, further laying the groundwork for Linked Open Data-based recommendation with many readily-available RDF KGs. Our approach bridges the gap between the Graph Machine Learning community (e.g., ICLR, LOG, ICML) and the Semantic Web community (e.g., ISWC, ESWC, CIKM), eliminating the need for semantic technologies such as SPARQL–the query language for RDF KGs. Our approach utilizes `AutoRDF2GML` [11], a framework to convert RDF data into heterogeneous graph data suitable for GNNs. The main contribution of this work is the evalu-

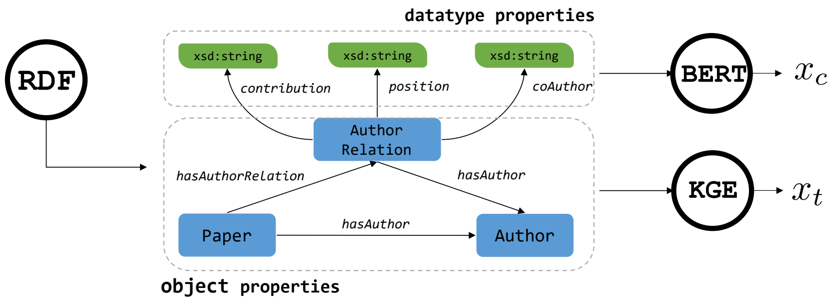

Fig. 2: Illustration of automatic features extraction with `AutoRDF2GML`.

ation of how semantic features and graph structure heterogeneity in RDF KGs affect the performance of GNN architectures across multiple recommendation scenarios. Our evaluation on two large RDF KGs, comprising up to 22 million RDF triples, demonstrates that semantic feature-based initialization techniques and the incorporation of heterogeneous graph structures can significantly boost performance. Our work also provides a solid basis for selecting GNN-based recommender system for RDF data. Overall, we make the following contributions:

1. We present a GNN-based recommendation pipeline aligned with RDF knowledge graphs. Our approach leverages the comprehensive semantics of RDF data, including both the RDF object properties (topological information) and RDF datatype properties (content-based information).
2. We evaluate the impact of various semantic feature initializations and graph structure heterogeneity on the performance of prominent GNN architectures. Our evaluation on two large RDF knowledge graphs across multiple recommendation scenarios shows that leveraging the semantic depth of RDF data significantly improves the performance of GNN-based recommender systems.

2 Framework

Our approach is composed of two main modules: (1) **RDF to Heterogeneous Graphs Transformation** (§2.1) and (2) **GNN-based Recommendation Systems** (§2.2), explained in the following.

2.1 RDF to Heterogeneous Graphs Transformation

To automatically transform large RDF data into a heterogeneous graph dataset, `AutoRDF2GML` [11], a comprehensive framework to convert RDF data into heterogeneous graph datasets suitable for graph-based machine learning methods, has been recently published. `AutoRDF2GML` automatically extracts numeric features from the RDF knowledge graph by utilizing both content-based information and topology-based relationships. A key advantage of `AutoRDF2GML` is its ability to automatically select and transform these semantically-rich content-based features, making it accessible to users without expertise in RDF and SPARQL.

Figure 2 illustrates the features extraction process with `AutoRDF2GML`. Given an RDF knowledge graph $\mathcal{KG}$, we transform it into a heterogeneous graph dataset

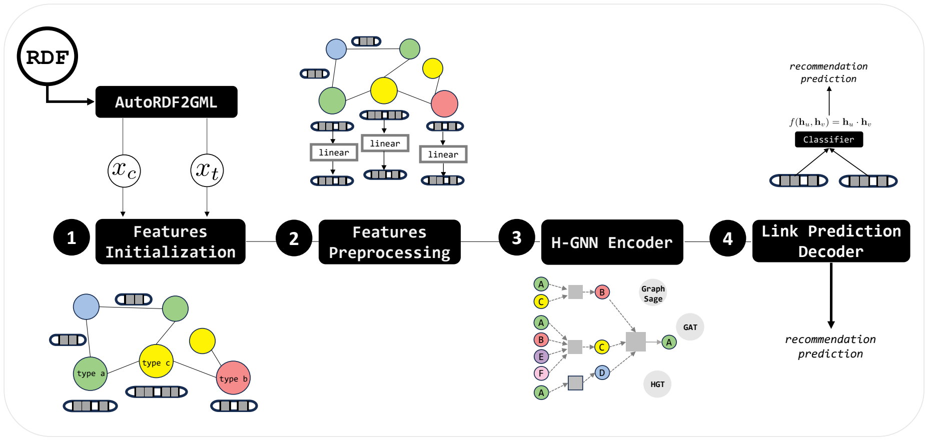

Fig. 3: GNN-based recommendation system with RDF-based features.

$\mathcal{HG}$, with $\mathcal{KG} \mapsto \mathcal{HG}(x_t, x_c)$. This dataset $\mathcal{HG}$ includes topology-based features x_t and content-based features x_c, as follows:

– **Content-based features** x_c : We utilize **RDF datatype properties** to capture the text semantic information of RDF knowledge graphs as content-based features. For example, ``contribution'' and ``coAuthor'' in Figure 2 are RDF datatype properties. Depending on the type of literal, `AutoRDF2GML` applies specific transformation rules. For instance, *string* literals are transformed into numerical features using BERT [6]-based embedding models.
– **Topology-based features** x_t : We leverage the RDF knowledge graph's topological information through its **RDF object properties** as the topology-based features. For example, ``hasAuthor'' in Figure 2 is an RDF object property. Techniques for obtaining these representations include Knowledge Graph Embedding (KGE) techniques such as TransE [3], DistMult [41], ComplEx [32], and RotatE [30], which effectively encode RDF entities into feature vectors. `AutoRDF2GML` automates this process using these techniques.

2.2 GNN-based Recommendation Systems with RDF Features

To demonstrate the benefits of integrating semantic features from RDF knowledge graphs into GNN-based recommendation system, we thoroughly evaluate the heterogeneous graph dataset generated in the previous step by integrating it into GNN-based recommender system, illustrated in Figure 3. Our approach involves an in-depth exploration of various semantic feature initialization methods (§2.2) and types of graph heterogeneity, on the performance of prominent GNN architectures across multiple recommendation scenarios (§3). Our primary goal is to analyse how GNNs can exploit semantic node features, further identify the most effective methods for combining these features having different semantic characteristics to enhance GNN-based recommendation systems. As illustrated in Figure 3, the GNN-based recommendation pipeline follows these steps:

1. **Feature Initialization:** Formally, feature initialization step assigns initial feature vectors x_v^0 to each node v in the heterogeneous graph $\mathcal{HG} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. These initial feature vectors x_v^0 are automatically computed e.g. with `AutoRDF2GML`, correspond to x_c and x_t, as explained in §2.1. In this process, our pipeline addresses both graph structure heterogeneity and feature heterogeneity; including the variations in vector dimensionality and underlying semantics, detailed in §2.2.
2. **Feature Preprocessing:** Due to the heterogeneity of the features extracted from the underlying RDF data, different node types may have features in different spaces. We apply linear transformation $\mathbf{x}_v = \mathbf{W}x_v + \mathbf{b}$ to project the features of varying node types into a unified feature space in a same dimension, following existing works [47,38,21].
3. **H-GNN Encoder:** We implement several heterogeneous GNN architectures to obtain the contextual node representations $\mathbf{h}_v$ of the graph. In a GNN, the contextual representation $\mathbf{h}_v$ of a node v is computed from its features $\mathbf{x}_v$ through the GNN function $\mathbf{h}_v = \text{GNN}(x_v)$ where $\text{GNN}(\cdot)$ denotes the function process the node features and its neighborhood to produce the contextual representation. We detailed our choice of GNN architectures in §3.
4. **Link Prediction Decoder:** Finally, a classifier is used for link prediction, producing the final recommendation. Given a pair of feature vectors h_u and h_v from the encoder, the classifier predicts the probability of existence of an edge by computing a similarity score $\hat{y}_{u\sim v}$ using *dot product* classifier: $\hat{y}_{u\sim v} = f(\mathbf{h}_u, \mathbf{h}_v) = \mathbf{h}_u \cdot \mathbf{h}_v$. We employ dot product due to its simplicity and strong performance, particularly in recommendation scenarios with a large number of potential links [28].

Semantic Feature Initialization. Nodes can be initialized with various feature vectors. In the following, we outline different node feature initializations with different levels of semantic richness. We consider both content-based and topology-based features, as well as their various combinations:

1. **One-hot-encoding (`one-hot`).** As a baseline approach, we employ one-hot encoding for feature initialization, a method commonly used for non-attributed graphs [5,26]. Unlike the other forms of feature initialization, one-hot vectors do not encapsulate any semantic information.
2. **Content-based: Natural language description (NLD, $\texttt{cb}_{\texttt{nld}}$).** This initialization method uses text embeddings, such as BERT [6] embeddings, derived from natural language descriptions of entities (e.g., labels, abstracts).
3. **Content-based: Literals ($\texttt{cb}_{\texttt{Literal}}$).** This initialization leverages all available RDF literals for the feature vectors. Beyond the textual descriptions of entities, these RDF literals can be categorical, numeric, boolean, or of another data type. Thus, in contrast to the *NLD* initialization, the initialization vectors are extended by context-based information of all data types.
4. **Topology-based (`tb`).** This initialization method uses encoded information about the topology of the entities in the input. Such information can be derived using knowledge graph embedding model (e.g., TransE [3]).

5. **NLD if available, otherwise topology-based (`comb`$_{\texttt{nld|tb}}$).** This method initializes with *NLD* vectors when available; otherwise, it uses topology-based embeddings. This approach is commonly used in related works (§5) when content-based node features are not provided for all node types [45,19,42,4].

Additionally, we include the combination of content-based and topology-based features to enhance the semantic richness of the node features, **6. Concatenation** (`comb`$_{\texttt{Concat}}$), **7. Addition** (`comb`$_{\texttt{Addition}}$), **8. Weighted Addition** (`comb`$_{\texttt{WAddition}}$), **9. Average** (`comb`$_{\texttt{Average}}$), and **10. Neural Combinator** (`comb`$_{\texttt{nc}}$) utilizes a feedforward neural network $z = f(W \cdot [a, b] + c)$.

3 Evaluation

Choice of GNN Architectures. We employ the following three widely-used models: **(1) GraphSAGE** [18] samples and aggregates features from node's local neighborhood, integrating topological and feature information using three aggregators: mean, LSTM, and pooling [22,49,12], **(2) Graph Attention Network (GAT)** [33] uses masked self-attention to weigh relevant neighbors, bypassing costly matrix operations and prior graph structure knowledge, while enabling message passing with multidimensional edge features for better node updates [12], and **(3) Heterogeneous Graph Transformer (HGT)** [21] is designed for complex heterogeneous graphs, using type-specific parameters and meta-relation-based attention to handle diverse node and edge types while learning implicit metapaths. While HGT is specifically designed for heterogeneous graphs [21], GraphSAGE and GAT were originally designed for homogeneous graphs. Thus, we applied GraphSAGE and GAT models to heterogeneous graphs by implementing the message and update functions for each edge type [12]. We use default parameters from GNN literature [26,21,36] for training the GNN models. We provide the detailed implementation settings including hyperparameters settings and other technical details for the model training in our Github.

Dataset. We evaluate on real-world recommendation tasks using the heterogeneous graph datasets `SOA-SW` [24] and `LPWC` [23], generated via `AutoRDF2GML` [11] from the RDF knowledge graphs *SemOpenAlex* [10] and *LinkedPapersWithCode* [9]. `SOA-SW` consists of 21.9 million RDF triples and models scientific works and their associated entities from the Semantic Web community. `LPWC` consists of 7.9 million RDF triples and provides comprehensive information about machine learning papers, including the tasks addressed, the datasets utilized and the methods implemented. Compared to other benchmarks that do not utilize RDF data (see Table 4), these datasets are unique in incorporating the semantic features from both topological and content-based information.

Recommendation Scenarios. In GNN-based recommendations, bipartite or homogeneous graphs are often used, which include only the node types relevant to the recommendation. To evaluate the effect of graph heterogeneity, we compare the performance of the recommender system on such graphs with using full heterogeneous graph, on the following recommendation scenarios:

Table 1: Evaluation of GNN models with varying feature initializations and graph heterogeneity for ***paper*** recommendation scenario on `SOA-SW`.

	Feature	GraphSAGE				GAT				HGT			
	Initialization	F1	Pre	Re	AUC	F1	Pre	Re	AUC	F1	Pre	Re	AUC
Full Heterogenous Graph	`one-hot`	0.806	0.899	0.731	0.937	0.875	0.925	0.830	0.962	0.890	0.880	0.901	0.949
	$\texttt{cb}_{\texttt{nld}}$	0.874	0.932	0.823	0.967	0.877	0.924	0.834	0.961	0.886	0.901	0.872	0.957
	$\texttt{cb}_{\texttt{Literal}}$	0.914	0.927	0.901	0.972	0.889	0.919	0.861	0.964	0.887	0.882	0.892	0.945
	`tb`	0.926	0.956	0.899	0.983	0.910	0.942	0.880	0.975	0.915	0.935	0.896	0.976
	$\texttt{comb}_{\texttt{nld\|tb}}$	0.933	0.959	0.908	0.985	0.920	0.929	0.910	0.973	0.906	0.943	0.872	0.976
	$\texttt{comb}_{\texttt{Concat}}$	0.931	0.951	0.911	0.982	0.918	0.948	0.890	0.979	0.925	**0.949**	0.902	**0.982**
	$\texttt{comb}_{\texttt{Addition}}$	0.921	0.951	0.892	0.982	0.922	**0.956**	0.889	0.982	0.882	0.939	0.832	0.970
	$\texttt{comb}_{\texttt{WAddition}}$	**0.940**	0.950	**0.929**	0.984	**0.923**	0.954	0.894	**0.983**	0.885	0.934	0.841	0.968
	$\texttt{comb}_{\texttt{Average}}$	0.926	**0.963**	0.893	**0.987**	0.898	0.932	0.866	0.971	**0.934**	0.937	**0.931**	0.977
	$\texttt{comb}_{\texttt{nc}}$	0.896	0.941	0.855	0.973	0.889	0.867	**0.912**	0.941	0.889	0.913	0.865	0.961
Bipartite Graph	`one-hot`	0.810	0.893	0.740	0.855	0.823	0.894	0.763	0.863	0.824	0.896	0.763	0.850
	$\texttt{cb}_{\texttt{nld}}$	0.855	0.911	0.805	0.942	0.830	0.871	0.793	0.892	0.854	0.836	**0.873**	0.924
	$\texttt{cb}_{\texttt{Literal}}$	0.882	0.910	0.856	0.955	0.846	0.852	0.841	0.903	0.847	0.846	0.848	0.914
	`tb`	**0.936**	**0.969**	0.905	**0.987**	**0.895**	**0.914**	0.877	**0.952**	**0.892**	**0.940**	0.850	**0.967**
	$\texttt{comb}_{\texttt{nld\|tb}}$	0.905	0.904	0.905	0.965	0.872	0.877	0.866	0.928	0.828	0.898	0.768	0.915
	$\texttt{comb}_{\texttt{Concat}}$	0.922	0.936	**0.908**	0.977	0.891	0.890	**0.893**	0.941	0.872	0.902	0.844	0.945
	$\texttt{comb}_{\texttt{Addition}}$	0.904	0.960	0.854	0.978	0.855	0.904	0.810	0.942	0.884	0.937	0.837	0.963
	$\texttt{comb}_{\texttt{WAddition}}$	0.910	0.956	0.869	0.977	0.873	0.902	0.845	0.939	0.876	0.904	0.850	0.949
	$\texttt{comb}_{\texttt{Average}}$	0.906	0.949	0.867	0.973	0.876	0.888	0.865	0.940	0.866	0.875	0.857	0.933
	$\texttt{comb}_{\texttt{nc}}$	0.849	0.906	0.800	0.939	0.846	0.890	0.807	0.924	0.818	0.918	0.738	0.915

1. **Paper Recommendation**: In this scenario, we perform link prediction on *author-work* edge using the `SOA-SW` dataset, experimenting with both full heterogeneous graph setting (6 node types, 7 edge types) and bipartite graph setting containing *author* and *work* nodes and their edges (*author-work*).
2. **Collaboration Recommendation**: In this scenario, we perform link prediction on the *author-author* edge type using the `SOA-SW` dataset. Since this scenario involves only *author* nodes, we experiment with both homogeneous and heterogeneous graph settings: the heterogeneous setting uses the entire graph, while the homogeneous setting focuses on *author* nodes and *author-author* edges forming the *co-author* network.
3. **Task Recommendation**: In this scenario, we perform link prediction on the *dataset-task* edge type using the `LPWC` dataset, on both full heterogeneous graph (4 node types, 6 edge types) and bipartite graph setting focusing on *dataset* and *task* nodes and their connecting edges.

4 Results and Discussions

The evaluation results for *paper*, *collaboration*, and *task* recommendation scenarios are summarized in Table 1, 2, and 3, respectively. Following existing works [39,35], we use the following evaluation metrics: ROC-AUC score (AUC), F1-score (F1), Precision (Pre), and Recall (Re). The best performance on *paper* recommendation scenario was achieved by GraphSAGE using the full heterogeneous graph setting: with feature initialization method $\texttt{comb}_{\texttt{WAddition}}$, it reached an F1-score of 0.940, and with method $\texttt{comb}_{\texttt{Average}}$, it obtained the highest ROC-AUC value of 0.987. Similarly, for *collaboration* recommendation scenario, the

Table 2: Evaluation of GNN models with varying feature initializations and graph heterogeneity for ***collaboration*** recommendation scenario on `SOA-SW`.

	Feature	GraphSAGE				GAT				HGT			
	Initialization	F1	Pre	Re	AUC	F1	Pre	Re	AUC	F1	Pre	Re	AUC
Full Heterogeneous Graph	`one-hot`	0.787	0.661	0.973	0.940	0.790	0.668	0.966	0.938	0.744	0.633	0.903	0.871
	$\mathtt{cb}_{\mathtt{nld}}$	0.793	0.665	0.984	0.957	0.797	0.671	0.983	0.956	0.773	0.654	0.946	0.894
	$\mathtt{cb}_{\mathtt{Literal}}$	0.795	0.666	0.987	0.952	0.744	0.601	0.977	0.861	0.751	0.642	0.905	0.884
	`tb`	0.791	0.663	0.979	0.921	0.800	0.676	0.981	0.947	0.780	0.654	0.966	0.911
	$\mathtt{comb}_{\mathtt{nld\vert tb}}$	0.803	**0.677**	0.989	0.960	**0.803**	0.676	0.988	0.952	0.743	0.613	0.941	0.869
	$\mathtt{comb}_{\mathtt{Concat}}$	0.801	0.676	0.982	0.950	0.798	0.675	0.976	0.941	**0.794**	**0.665**	**0.987**	**0.959**
	$\mathtt{comb}_{\mathtt{Addition}}$	0.795	0.669	0.980	0.946	0.802	0.676	0.985	0.958	0.767	0.645	0.946	0.890
	$\mathtt{comb}_{\mathtt{WAddition}}$	**0.804**	**0.677**	0.989	0.961	**0.803**	0.675	**0.990**	**0.964**	0.760	0.632	0.956	0.886
	$\mathtt{comb}_{\mathtt{Average}}$	0.803	0.675	**0.991**	**0.969**	0.801	0.675	0.984	0.949	0.765	0.641	0.950	0.890
	$\mathtt{comb}_{\mathtt{nc}}$	0.796	0.666	0.989	0.942	0.802	**0.677**	0.985	0.949	0.736	0.611	0.926	0.870
Homogeneous Graph	`one-hot`	0.739	0.644	0.869	0.844	0.755	0.663	0.876	0.868	-	-	-	-
	$\mathtt{cb}_{\mathtt{nld}}$	0.741	0.639	0.881	0.848	0.756	0.663	0.879	0.885	-	-	-	-
	$\mathtt{cb}_{\mathtt{Literal}}$	0.771	**0.672**	0.904	0.906	0.757	0.673	0.865	0.877	-	-	-	-
	`tb`	0.774	0.663	**0.928**	0.909	0.751	0.649	**0.892**	**0.888**	-	-	-	-
	$\mathtt{comb}_{\mathtt{Concat}}$	**0.776**	0.668	0.927	**0.913**	0.757	0.666	0.877	0.885	-	-	-	-
	$\mathtt{comb}_{\mathtt{Addition}}$	0.765	0.659	0.914	0.902	0.757	0.667	0.875	0.879	-	-	-	-
	$\mathtt{comb}_{\mathtt{WAddition}}$	0.762	0.656	0.910	0.898	**0.764**	**0.679**	0.874	0.882	-	-	-	-
	$\mathtt{comb}_{\mathtt{Average}}$	0.762	0.657	0.908	0.898	0.745	0.644	0.882	0.837	-	-	-	-
	$\mathtt{comb}_{\mathtt{nc}}$	0.741	0.641	0.879	0.834	0.736	0.634	0.879	0.825	-	-	-	-

best performances were achieved by GraphSAGE in the full heterogeneous graph setting: with $\mathtt{comb}_{\mathtt{WAddition}}$, it achieved the highest F1-score of **0.804**, and with $\mathtt{comb}_{\mathtt{Average}}$, the highest ROC-AUC value of **0.969**. In *task* recommendation scenario, similar to the other two scenarios, the best performance was achieved by GraphSAGE in the full heterogeneous graph setting (highest F1-score of **0.923** with $\mathtt{comb}_{\mathtt{Average}}$; highest ROC-AUC value of **0.975** with $\mathtt{comb}_{\mathtt{WAddition}}$). In the following, we discuss our results from several perspectives.

Analyis of Semantic Feature Initialization Methods. Across all our experiments (Tables 5-7), feature initialization techniques using semantic node features consistently outperformed those with one-hot vectors. Additionally, our analysis showed that heterogeneous graph structures are superior to both bipartite and homogeneous graph configurations. Our findings highlight the effectiveness of using $\mathtt{comb}_{\mathtt{WAddition}}$ and $\mathtt{comb}_{\mathtt{Average}}$ as the most successful combination methods. These approaches integrate semantic information from both content and topological structure of the underlying RDF data, giving the best results.

Performance across Different GNN Architectures. In all three experiments, GraphSAGE, when initialized with either $\mathtt{comb}_{\mathtt{WAddition}}$ or $\mathtt{comb}_{\mathtt{Average}}$ and applied to the full heterogeneous graph setting, consistently outperforms other models across all evaluation metrics. These results suggest that GraphSAGE is particularly effective at leveraging semantically rich node features, as also shown by [18,22]. Additionally, this result aligns with [26], which showed that GNN architectures designed for homogeneous graphs (e.g., GraphSAGE) can surpass more complex heterogeneous graph neural networks (HGNNs) when provided with appropriate input. This proves the importance of semantically rich

Table 3: Evaluation of GNN models with varying feature initializations and graph heterogeneity for *task* recommendation on `LPWC`.

	Feature	GraphSAGE				GAT				HGT			
	Initialization	F1	Pre	Re	AUC	F1	Pre	Re	AUC	F1	Pre	Re	AUC
Heterogeneous Graph	`one-hot`	0.739	0.791	0.694	0.834	0.748	0.766	0.732	0.794	0.778	0.801	0.756	0.859
	$\texttt{cb}_{\texttt{nld}}$	0.837	0.873	0.803	0.928	0.802	0.742	0.872	0.847	0.783	0.721	0.857	0.862
	$\texttt{cb}_{\texttt{Literal}}$	0.784	0.697	0.895	0.853	0.800	0.834	0.769	0.879	0.820	0.778	0.868	0.894
	`tb`	0.903	0.913	0.893	0.965	0.868	**0.916**	0.826	0.936	0.877	0.837	0.922	0.936
	$\texttt{comb}_{\texttt{Concat}}$	0.907	0.926	0.889	0.970	0.873	0.911	0.839	0.936	0.826	0.779	0.878	0.898
	$\texttt{comb}_{\texttt{Addition}}$	0.912	0.921	0.903	0.967	0.875	0.890	0.860	0.936	**0.885**	**0.845**	**0.930**	**0.943**
	$\texttt{comb}_{\texttt{WAddition}}$	0.918	**0.933**	0.903	**0.975**	0.872	0.896	0.849	0.938	0.875	0.841	0.912	0.936
	$\texttt{comb}_{\texttt{Average}}$	**0.923**	0.920	0.926	0.971	**0.882**	0.882	0.883	**0.942**	0.829	0.767	0.903	0.896
	$\texttt{comb}_{\texttt{nc}}$	0.879	0.832	**0.932**	0.943	0.825	0.766	**0.894**	0.886	0.783	0.811	0.757	0.875
Bipartite Graph	`one-hot`	0.782	0.762	0.803	0.851	0.737	0.601	**0.951**	0.745	0.697	0.541	0.980	0.743
	$\texttt{cb}_{\texttt{nld}}$	0.836	0.800	0.876	0.903	0.806	0.771	0.845	0.852	0.795	0.718	0.891	0.868
	$\texttt{cb}_{\texttt{Literal}}$	0.756	0.739	0.773	0.823	0.815	0.779	0.855	0.859	0.798	0.739	0.868	0.869
	`tb`	**0.915**	0.930	0.901	0.971	**0.830**	**0.809**	0.852	**0.898**	0.819	**0.866**	0.776	**0.912**
	$\texttt{comb}_{\texttt{Concat}}$	0.904	**0.936**	0.873	**0.973**	0.799	0.717	0.903	0.869	**0.845**	0.803	0.892	0.909
	$\texttt{comb}_{\texttt{Addition}}$	0.889	0.902	0.876	0.955	0.803	0.750	0.864	0.878	0.721	0.574	**0.971**	0.772
	$\texttt{comb}_{\texttt{WAddition}}$	0.882	0.836	**0.933**	0.940	0.791	0.717	0.882	0.865	0.794	0.726	0.875	0.854
	$\texttt{comb}_{\texttt{Average}}$	0.862	0.885	0.841	0.940	0.793	0.726	0.872	0.869	0.732	0.599	0.939	0.769
	$\texttt{comb}_{\texttt{nc}}$	0.813	0.746	0.895	0.875	0.740	0.617	0.923	0.725	0.740	0.603	0.958	0.767

node features in the GNN-based recommendation tasks. Conversely, in scenarios where one-hot vectors without semantic information are used as node features, GAT and HGT consistently outperform GraphSAGE in the full heterogeneous graph setting. This is shown in Tables 1–3, where GraphSAGE achieves F1 scores of 0.806, 0.787, and 0.739, while GAT/HGT achieves scores of 0.89, 0.79, and 0.778, respectively. This underscores their effectiveness in leveraging the graph's structure, even when semantically rich node features are not available [21,33,26].

Full Heterogeneous vs. Bipartite Graph Settings. Our findings emphasize that in bipartite graph settings, feature initialization with TransE knowledge graph embeddings–a topology-based feature–consistently delivers the best results, as shown by the performance metrics in the lower half of Tables 1 and 3. Specifically, in the *paper* recommendation scenario on `SOA-SW` (see Table 1), the best-performing full heterogeneous graph seting using $\texttt{comb}_{\texttt{WAddition}}$ feature initialization achieved an F1-score of 0.940. In comparison, the bipartite graph setting initialized with TransE embeddings attained an F1-score of 0.936, showing a minimal difference of just 0.4%. Similarly, for *task* recommendation scenario on `LPWC` (see Table 3), the best full heterogeneous graph seting with $\texttt{comb}_{\texttt{Average}}$ achieved an F1-score of 0.923, while the bipartite graph setting using TransE embeddings resulted in an F1-score of 0.915, a difference of 0.8% score.

5 Related Work

GNNs for Heterogeneous Graphs and the Benchmarks. Heterogeneous graphs are used to model different types of objects and relationships [1,39], offering a more accurate representation of real-world phenomena compared to homogeneous graphs. This structured modeling improves application performance, with GNNs outperforming other methods in recommender systems [14]. In Table 4, we summarized benchmark datasets for heterogeneous graph tasks. Table 4 shows that, apart from `SOA-SW` and `LPWC`, most benchmark datasets do not use RDF data. They often lack node features for all node types, or when present, these features mainly stem from inherent NLD properties. Consequently, performance gains on these benchmarks may result from improved topology-based node features rather than true advancements in GNN models, which reflect optimized feature engineering [25]. As of this writing, `SOA-SW` and `LPWC` are the only large graph datasets that provide both content-based (`NLD` and `Literals\NLD`) and topology-based node features for all node types, ideal for our evaluation.

Table 4: Heterogeneous Graphs

Benchmarks	`NLD`	`Literals\NLD`	`Topology`
AMiner [7]	✗	✗	✗
MovieLens [16]	✗	✗	✗
LastFM [13]	✗	✗	✗
HGB_Freebase [26]	✗	✗	✗
HGB_LastFM [26]	✗	✗	✗
Douban [48]	✗	✗	✗
Flixster [48]	✗	✗	✗
Yahoo-Music [48]	✗	✗	✗
OGB_MAG [20]	✓	✗	✗
DBLP [13]	✓	✗	✗
IMDB [13]	✓	✗	✗
HGB_DBLP [26]	✓	✗	✗
HGB_ACM [26]	✓	✗	✗
HGB_PubMed [26]	✓	✗	✗
HGB_Amazon [26]	✓	✓	✗
HGB_IMDB [26]	✓	✓	✗
SOA-SW [24]	✓	✓	✓
LPWC [23]	✓	✓	✓

GNNs for Recommender Systems. GNNs are widely used in recommender systems because these systems often involve graph-structured data [44,2], and GNNs excel at graph representation learning [39]. Studies show that GNN-based models outperform traditional [17,39] and prior GNN-based methods [37,44]. The advantage of GNNs especially lies in their ability to explore multi-hop relationships, which has been shown to significantly benefit recommender systems [37]. Recent advancements in Large Language Models (LLMs) have introduced their application in recommender systems, with approaches like [15] framing recommendation problems as prompt-based natural language tasks. However, employing prompt-based methods on graph-structured data presents challenges, as it requires transforming complex graph content into textual prompts. In addition, research shows that LLMs and GNNs often complement each other effectively [40]. For that reason, we focus on GNNs-based methods in the current work.

6 Conclusion

In this paper, we integrate RDF knowledge graphs with Graph Neural Networks (GNNs), demonstrating the advantages of using the rich semantics of RDF to improve GNN-based recommendation systems. We conducted an extensive evaluation across multiple recommendation scenarios, analyzing the impact of various semantic node feature initializations and graph structure heterogeneity. Our results demonstrate that these components significantly enhance the performance of GNN-based recommender systems. Additionally, by considering RDF knowledge graphs with their global knowledge, our approach not only uses the node types directly relevant to recommendations, but also auxiliary nodes that provide valuable additional contextual information. This approach effectively facilitates the creation of semantically enriched features from both content and topological information, leading to improved performance of GNN-based recommender systems. In future work, we will consider the combination of several RDF knowledge graphs from the Linked Open Data cloud for GNN-based recommendation.

References

1. Ansarizadeh, F., Tay, D.B.H., Thiruvady, D.R., Robles-Kelly, A.: Deterministic sampling in heterogeneous graph neural networks. Pattern Recognit. Lett. **172**, 74–81 (2023)
2. van den Berg, R., Kipf, T.N., Welling, M.: Graph convolutional matrix completion (2017), https://arxiv.org/abs/1706.02263
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
4. Chi, H., Wang, Y., Hao, Q., Xia, H.: Residual network and embedding usage: New tricks of node classification with graph convolutional networks. In: Journal of Physics. vol. 2171, p. 012011. IOP Publishing (2022)
5. Cui, H., Lu, Z., Li, P., Yang, C.: On positional and structural node features for graph neural networks on non-attributed graphs. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 3898–3902. CIKM'22 (2022)
6. Devlin, J., Chang, M.W., 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 (2019). https://doi.org/10.18653/v1/N19-1423
7. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD international conference on knowledge discovery and data mining. pp. 135–144 (2017)
8. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. Semantic Web **9**(1), 77–129 (2018), https://doi.org/10.3233/SW-170275
9. 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'23 (2023)
10. 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'23, Springer (2023)
11. Färber, M., Lamprecht, D., Susanti, Y.: Autordf2gml: Facilitating rdf integration in graph machine learning. In: Proceedings of the 2024 International Semantic Web Conference. ISWC 2024 (2024)
12. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
13. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In: Proceedings of the 2020 Web Conference. pp. 2331–2341. Web'20 (2020)
14. Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., et al.: A survey of graph neural networks for recommender systems: Challenges, methods, and directions. ACM Transactions on Recommender Systems **1**(1), 1–51 (2023)
15. Geng, S., Liu, S., Fu, Z., Ge, Y., Zhang, Y.: Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In: Proceedings of the 16th ACM Conference on Recommender Systems. p. 299–315. RecSys '22 (2022), https://doi.org/10.1145/3523227.3546767

16. GroupLens Research: MovieLens (2023), https://movielens.org, accessed on: 29.09.2023
17. Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering **34**(8), 3549–3568 (2020)
18. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
19. He, M., Wei, Z., Feng, S., Huang, Z., Li, W., Sun, Y., Yu, D.: Spectral heterogeneous graph convolutions via positive noncommutative polynomials. arXiv preprint arXiv:2305.19872 (2023)
20. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems **33**, 22118–22133 (2020)
21. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous Graph Transformer. In: Proceedings of the 2020 Web Conference. pp. 2704–2710. Web'20 (2020)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
23. Lamprecht, D., Färber, M.: Graph Machine Learning Dataset LPWC. https://doi.org/10.5281/zenodo.10299366 (2023)
24. Lamprecht, D., Färber, M.: Graph Machine Learning Dataset SOA-SW. https://doi.org/10.5281/zenodo.10299428 (2023)
25. 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 (2023)
26. Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., Tang, J.: Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In: ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 1150–1160 (2021)
27. Manola, F., Miller, E.: RDF Primer — w3.org. https://www.w3.org/TR/rdf-primer/ (2004), [Accessed 04-09-2024]
28. Momanyi, B.M., Zhou, Y.W., Grace-Mercure, B.K., Temesgen, S.A., Basharat, A., Ning, L., Tang, L., Gao, H., Lin, H., Tang, H.: SAGESDA: Multi-GraphSAGE networks for predicting SnoRNA-disease associations. Current Research in Structural Biology **7**, 100122 (2024). https://doi.org/10.1016/j.crstbi.2023.100122
29. Polleres, A., Kamdar, M.R., Fernández, J.D., Tudorache, T., Musen, M.A.: A more decentralized vision for linked data. Semantic Web **11**(1), 101–113 (2020). https://doi.org/10.3233/SW-190380, https://doi.org/10.3233/SW-190380
30. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space (2019)
31. Szklarczyk, D., Kirsch, R., Koutrouli, M., Nastou, K., Mehryary, F., Hachilif, R., Gable, A.L., Fang, T., Doncheva, N.T., Pyysalo, S., et al.: The string database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest. Nucleic acids research **51**(D1), D638–D646 (2023)
32. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
33. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)

34. Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., Guo, M.: 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 '18 (2018)
35. Wang, H., Zhao, M., Xie, X., Li, W., Guo, M.: Knowledge graph convolutional networks for recommender systems. In: Proceedings of the World Wide Web Conference. pp. 3307–3313. WWW'19 (2019)
36. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. pp. 165–174 (2019)
37. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 165–174. SIGIR'19 (2019), https://doi.org/10.1145/3331184.3331267
38. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: World Wide Conference. pp. 2022–2032 (2019)
39. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. ACM Computing Surveys **55**(5), 1–37 (2022)
40. Xu, J., Wu, Z., Lin, M., Zhang, X., Wang, S.: LLM and GNN are complementary: Distilling LLM for multimodal graph learning. CoRR **abs/2406.01032** (2024). https://doi.org/10.48550/ARXIV.2406.01032, https://doi.org/10.48550/arXiv.2406.01032
41. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
42. Yang, X., Yan, M., Pan, S., Ye, X., Fan, D.: Simple and efficient heterogeneous graph neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 10816–10824 (2023)
43. Yang, Z., Dong, S.: HAGERec: Hierarchical Attention Graph Convolutional Network Incorporating Knowledge Graph for Explainable Recommendation. Knowledge-Based Systems **204**, 106194 (2020)
44. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 974–983. KDD '18, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3219890, https://doi.org/10.1145/3219819.3219890
45. Yu, L., Shen, J., Li, J., Lerer, A.: Scalable graph neural networks for heterogeneous graphs. arXiv preprint arXiv:2011.09679 (2020)
46. Zdrazil, B., Felix, E., Hunter, F., Manners, E.J., Blackshaw, J., Corbett, S., de Veij, M., Ioannidis, H., Lopez, D.M., Mosquera, J.F., et al.: The chembl database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. Nucleic acids research **52**(D1) (2024)
47. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: ACM SIGKDD international conference on knowledge discovery & data mining. pp. 793–803 (2019)
48. Zhang, M., Chen, Y.: Inductive matrix completion based on graph neural networks. arXiv preprint arXiv:1904.12058 (2019)
49. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI open **1**, 57–81 (2020)