A Graph Language Modeling Framework for the Ontological Enrichment of Conceptual Models

Syed Juned Ali[1][0000−0002−0710−8052] and Dominik Bork[1][0000−0001−8259−2297]

TU Wien, Business Informatics Group, Vienna, Austria
{syed.juned.ali, dominik.bork}@tuwien.ac.at

Abstract. Conceptual models (CMs) offer a structured way to organize and communicate information in information systems. However, current models lack adequate semantics of the terminology of the underlying domain model, leading to inconsistent interpretations and uses of information. Ontology-driven conceptual modeling languages provide primitives for articulating these domain notions based on the ontological categories, i.e., stereotypes put forth by upper-level (or foundational) ontologies. Existing CMs have been created using ontologically-neutral languages (e.g., UML, ER). Enriching these models with ontological categories can better support model evaluation, meaning negotiation, semantic interoperability, and complexity management. However, manual stereotyping is prohibitive, given the sheer size of the legacy base of ontologically-neutral models. In this paper, we present a graph language modeling framework for conceptual models that combines finetuning pre-trained language models to learn the vector representation of OntoUML models' data and then perform a graph neural networks-based node classification that exploits the model's graph structure to predict the stereotype of model classes and relations. We show with an extensive comparative evaluation that our approach significantly outperforms existing stereotype prediction approaches.

Keywords: Ontology-Driven Conceptual Models · Graph Neural Networks · Pre-trained Language Model · Representation Learning.

1 Introduction

Conceptual models were introduced to increase understanding and communication of a system or domain among stakeholders in information systems for design, analysis, and development purposes. Ontologies proved useful in assessing whether different conceptual modeling procedures will likely lead to good representations of real-world phenomena and evaluate the ontological soundness of a conceptual modeling language and its corresponding concepts and grammar. Foundational ontologies emerged to consistently define fundamental concepts in conceptual modeling, e.g., types and taxonomic structures, roles and relational properties, part-whole relations, and multi-level structures. The Unified Foundational Ontology (UFO) was developed to provide foundations for all

these major conceptual modeling constructs. UFO has been applied to design a general-purpose language OntoUML for ontology-driven conceptual modeling (ODCM) as a revised version of UML such that its modeling primitives reflect the ontological distinctions put forth by UFO, and its metamodel includes semantically-motivated syntactic constraints that reflect the axiomatization of UFO. Empirical evidence shows that OntoUML significantly contributes to improving the quality of conceptual models [29,11]. Semantically rich conceptual models created from OntoUML enable various crucial applications such as *i*) better support for semantic interoperability of the systems as shown in [13]; *ii*) for supporting ontological analysis, meaning explication and negotiation, and conceptual clarification using the fundamental ontological distinctions embodied in a foundational ontology as a conceptual toolbox; *iii*) for more sophisticated conceptual model modularization mechanism [14]; and *iv*) database design [4]. Therefore, we notice sufficient value from generating semantically rich models.

Many CMs exist, however, they are often created using ontologically-neutral languages like UML and ER. Enriching the elements of such models with ontological categories would add the aforementioned benefits. However, given the sheer size of these legacy models, manually enriching them is a prohibitive task. For this reason, in our previous work [1], we presented an automated approach using a trained Graph Neural Network (GNN) on a set of OntoUML models to predict the ontological category of classes and relations in a model. In the present work, we aim to resolve the limitations of our and other stereotype prediction approaches and significantly improve the prediction accuracy for a larger set of less frequent stereotypes. We propose a novel framework for CMs that couples the strengths of transformer-based language models [28] with GNNs [15].

We transformed the stereotype prediction of a class or a relation in an OntoUML model into a node classification task. We first transform an OntoUML model into a Knowledge Graph (KG) using a transformation mechanism described in [1]. In particular, the transformation converts an OntoUML model into a type of KG termed as Conceptual Knowledge Graph (CKG) that can comprehensively capture the CM's graph structure and the ontological stereotypes of the model. Next, the node information, which includes node label and meta properties, is encoded as vectors (i.e., node embeddings) using pre-trained language models (PLM) such as GLoVE, BERT [22,8]. The natural language semantics carry significant information about the category of a class or a relation. E.g., a class "Person" is generally of the type *kind* in OntoUML models. Therefore, the vector representation, i.e., the node embeddings of the elements in an OntoUML model that capture such linguistic semantics, can train a neural network to predict a model element's ontological stereotype.

We note three limitations in [1] that we aim to resolve in this work. Firstly, the node embeddings lack relevant contextual information, i.e., the natural language representation of the nodes' data is generated using only the individual node data. It does not use the other nodes of the model. E.g., the node embedding of the class "Person" in Fig. 1 should depend upon the neighboring nodes such as "Patient," "Adult," "Child," or "Physician," which add crucial contextual

information about the *meaning* of the class "Person." We resolve this issue by extending the node data by adding contextual data from the neighboring nodes. Secondly, PLMs such as BERT or GLoVE are trained on generalized natural language text and, therefore, are not particularly aware that the node labels are specific to an OntoUML model. We resolve this limitation by finetuning the PLM for OntoUML models' data and then using the generated node embeddings to classify the stereotype of each node. Finally, [1] misses out on using the stereotype information of the nodes already labeled with a stereotype. E.g., let's consider "Person" and "Adult" as already labeled nodes and "Child" as an unlabeled node in Fig. 1. We can use the stereotype of "Person" and "Adult" to predict the stereotype "Child". In other words, the stereotype information of a node's contextual nodes can be used for the node's stereotype prediction.

In this work, we present our improved solution that tackles the above limitations of the previous work to achieve automated ontological category prediction. Consequently, we developed a **G**raph **La**nguage **M**odeling framework for **C**onceptual **M**odels (**GLaM4CM**) that enables conceptual modeling applications by combining PLMs and GNNs on the KG representation of CMs, i.e., CKGs. GLaM4CM uses the CKGs as an intermediary to achieve conceptual modeling tasks such as model autocompletion, classification, and ontological enrichment using node stereotype classification, which is the task that we use to evaluate our framework in this paper. Since OntoUML and UFO are among the most used modeling languages and foundational ontologies in ODCM, our proposal makes a clear contribution to advancing the state of the art. We present an experimental evaluation to validate our approach and to provide a detailed comparative impact analysis of various design choices in our solution architecture based on the OntoUML FAIR model dataset described in [4]. We make the code available on Github* for (*i*) CKG creation and context generation (*ii*) finetuning PLMs and training GNN on the OntoUML CKGs and, (*iii*) reproducing the results of our work.

In the remainder of this paper, Section 2 introduces relevant background. Section 3 discusses the existing graph language modeling solutions and how our proposal advances the state of the art. Section 4 presents the three contributions of this paper, namely, *i*) a graph language modeling framework for an automated stereotype prediction task; *ii*) an extensive evaluation of the different parameters involved in modeling the nodes' context; and *ii*) a comparative analysis of our approach with existing approaches. Section 5 reports the results of an experimental evaluation of our approach. Section 6 discusses the obtained results, their implications, and threats to validity before we conclude this paper in Section 7.

2 Background

We now provide the relevant background for developing our GLaM4CM framework and its application for ontological stereotype prediction.

*https://github.com/junaidiiith/GLM-Stereotype-Prediction

Conceptual Knowledge Graphs Conceptual models facilitate detection and correction of system development errors [30]. However, error detection techniques are constrained when applied to ontologically-neutral modeling languages like ER and UML as they lack an adequate semantic specification of their terminology, which leads to inconsistent interpretations and uses [21,12]. Ontologies proved useful in assessing whether different conceptual modeling procedures will likely lead to good representations of real-world phenomena and evaluate the ontological soundness of a conceptual modeling language and its corresponding concepts and grammar. ODCM extends or supports conceptual modeling techniques by ontological theories that further formalize the conceptual modeling grammars [29], thereby strengthening the ontological commitment of these languages and thus improving the semantic quality of the conceptual modeling language. Knowledge Graphs represent a collection of interlinked descriptions of entities – e.g., objects, events, and concepts. KGs provide a foundation for data integration, fusion, analytics, and sharing [24] based on linked data and semantic metadata. KGs have been recently used for the representation [25,26] of CMs. Such KG-based representations can act as the intermediary representation of CMs to enable ML-based applications on CMs. *Conceptual Knowledge Graphs* (CKG) are termed as "ontologically enriched KGs representing CMs" [1].

Transformer-based node classification Transformer architecture [28] based language models have proven to be useful in supporting natural language processing (NLP) tasks that operate by leveraging a neural network model that learns a representation of words that captures the contextual relationships between words in a text. BERT, which stands for "Bidirectional Encoder Representations from Transformers", is particularly powerful it captures how each word or token relates to the entire sentence by processing it forward and backward [8]. These transformer-based models are generally trained on a massive text corpus, called the pretraining phase to produce a PLM that provides rich contextual embeddings of text that can be used for various NLP tasks. In the finetuning phase, a PLM is further trained on a smaller dataset and adapted to specific NLP tasks by the model's parameters adjustment to make it proficient for at NLP tasks such as text classification, sentiment analysis, or question answering for a specific dataset.

In the context of graph data, a BERT model can be adapted for node label classification. By representing nodes in a graph as sequences of text or encoding their local neighborhood information into text, we can apply BERT to learn meaningful representations of graph nodes that incorporate local graph structure and semantics of the graph data. Based on the idea of transforming node labels as a sequence of texts, BERT can be finetuned on the graph data. In our work, we finetune pre-trained BERT models for two tasks, i.e., masked language modeling (MLM) and sequence classification (SC). In masked language modeling, some node labels may be masked, i.e., hidden, and the model is tasked with predicting those masked labels based on the contextual text. In sequence classification, a node label is predicted by a classification neural network layer that uses the node embedding produced by the BERT model. Note that the pre-trained embeddings

from the pretraining phase are enriched with dataset (i.e., OntoUML) and task-specific (i.e., MLM and SC) embeddings during finetuning.

Graph Neural Networks are neural models that learn graph representations via *message passing* between graph nodes. A node aggregates information from its neighborhood. GNNs leverage the inherent structure and connectivity of the graph to make informed predictions about the labels or properties of individual nodes. Thereby, GNNs capture the local and global context of each node, allowing them to assign appropriate labels based on the learned representations. This approach is particularly valuable in diverse applications, such as social network analysis, recommendation systems, biology, and knowledge graphs, where nodes represent entities and relationships. Recently, variants of GNNs such as Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Graph Recurrent Networks (GRN) have demonstrated good performance on many deep learning tasks. GraphSAGE [15] generalized the aggregation function (compared to GCN, which uses "mean" as the aggregation function) that generates node embeddings by sampling and aggregating features from a node's local neighborhood. GNNs can be combined with embeddings from pre-trained or language models-based BERT embeddings such that embeddings capture node-specific information, and the GNN can propagate the information by leveraging the graph structure.

3 Related Work

Next, we discuss existing works for ontological stereotype prediction and NLP approaches to support conceptual modeling tasks.

Ontological category prediction Barcelos et al. [3] present a rule-based solution to infer the ontological stereotype of OntoUML classes. They present 37 different rules that use the UFO semantics to predict the ontological stereotype of OntoUML classes. They restrict the inference for eight categories of classes, namely, *kind*, *subkind*, *phase*, *role*, *category*, *mixin*, *roleMixin* and *phaseMixin* [3] and show, that the 37 rules can only predict the stereotype of a class accurately in 15-20% of the cases. However, the rules can predict with good confidence that the correct stereotype is within the top three predicted classes for each node.

Keet et al. [17] proposed a method leveraging a decision tree for DOLCE [5] categories to select the ontological category for their models. A survey by Trojahn et al. [27] indicates that several works enrich the domain ontologies with foundational ontology concepts. Felipe et al. [18] propose mapping rules between the noun synsets of Wordnet and the top-level constructs of UFO. RDF2Vec [23] considers entities' lexical terms for learning node embeddings; however, they treat an entity composed of multiple words as a single entity, which limits generality. OWL2Vec [6] considers word compositions and adds OWL constraints. Junior et al. [16] propose an approach that automatically classifies domain entities into top-level concepts using informal definitions and the term's word embedding.

Word embeddings for Conceptual Modeling A conceptual model's lexical terms contain natural language semantics. Therefore, conceptual modeling tasks can benefit from NLP techniques. Efstathiou et al. [9] released a word2vec

model trained over 15GB of textual data in the software engineering context. Lopez et al. [19] present a word2vec model trained on model-driven engineering (MDE) and conceptual modeling data corpus [19].

Synopsis Considering ontological stereotype prediction, [3] is the closest to our work. However, there are several distinctions. Their approach *i*) is based on a manually created set of non-exhaustive 37 rules compared to our data-driven approach, i.e., it uses deep learning-based patterns to infer the stereotype from the natural language semantics and graph structural information within the OntoUML model; *ii*) is restricted to predicting stereotypes only for classes whereas our approach also considers relations; and *iii*) uses only eight ontological stereotype classes, whereas our approach works for 21 classes, which includes many less frequent classes that are naturally harder to predict. Based on the related work, we present a comparative analysis of our approach with [3] and also show the comparison of using the word embeddings from [9,19] with the embeddings generated by finetuned language models from our framework.

4 The GLaM4CM Framework

We now introduce our end-to-end framework, including its architecture and the steps involved in training and finetuning language models and training GNNs.

4.1 CKG transformation and Context Generation

First, we elaborate on transforming an OntoUML model into a CKG and generating the node data using the neighbors' node data. Fig. 1 shows an example of an OntoUML model of a medical facility (left) and the transformed CKG (right). Note that only the node label and stereotype information are shown in the CKG for simplicity. Other attributes such as node type, i.e., Class or Relation, also form part of the node data (cf. the textual descriptions of the nodes "Person" and "Symptom"). Further information can be similarly added. The CKG represents classes and relations such as "reportedBy" as nodes with a stereotype label. Generalization relations, e.g., "Child" to "Person" relations, do not have a stereotype and are thus not transformed into a CKG node.

We create textual data for each node that captures its neighboring context. Given a directed graph G and a node n within G, the first step is to identify all the nearest neighbors, denoted as N_b, of node n within specified k hops. These neighbors are typically nodes within k hops or edges away from node n in the graph. Next, we create a string of shortest paths from node n to each neighbor N_b as a concatenation of the node labels of all the nodes present in the shortest path. This string effectively captures the sequence of node labels that need to be traversed to reach each neighbor within the specified k hops from node n. Fig. 1 shows paths starting from node "Person" which cover nodes and edges in blue for $k = 1$ and the additional green nodes and edges for $k = 2$. The node description follows a specific format of ⟨ node type ⟩ ⟨ node label ⟩: ⟨ node stereotype ⟩ and the edge description is replaced by the keyword "generalizes" if the edge is between two class nodes or "relates" if the edge is between a class node and

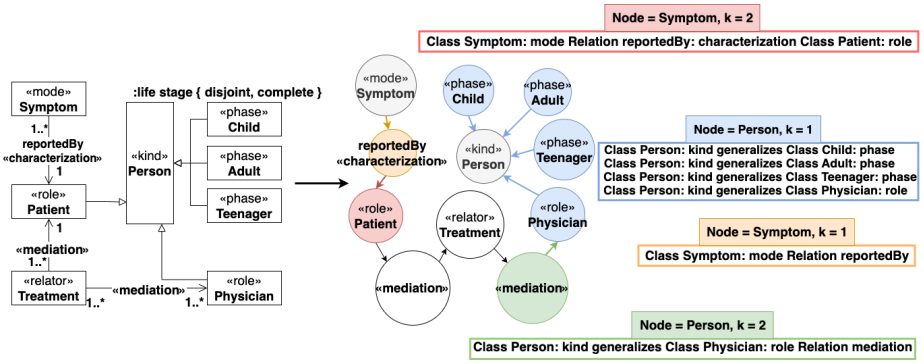

Fig. 1: An OntoUML model CKG transformation and context generation

a relation node. Note that some relation nodes do not have a label and only stereotype information, such as the relation node with "mediation" stereotypes in Fig. 1, which makes predicting the stereotype of such a node difficult. Adding neighboring information can help in predicting the stereotype for such nodes.

4.2 Problem Description

In the following, we formally describe the node stereotype prediction problem for a set of OntoUML models. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ represent a set of CKGs that represent OntoUML models, where $G_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a graph with a set of nodes $\mathcal{V}_i$ and edges $\mathcal{E}_i$. Each node in $\mathcal{V}_i$ is associated with a text content denoted as X_i. The node data vector is obtained from a language model using the node's textual content. To transform the node text labels into node embeddings, a node mapping function $\phi(\cdot)$ is employed. In this context, $\phi(\cdot)$ represents a PLM such as BERT, which generates embeddings for text sequences. A graph nodes' text labels X_i are encoded into a node embedding $\mathbf{H}_i = \{\mathbf{h}_{i1}, \mathbf{h}_{i2}, \ldots, \mathbf{h}_{i|\mathcal{V}_i|}\}$, where $\mathbf{h}_{ij} = \phi(X_{ij})$ for each node j in G_i. Subsequently, a GNN is utilized for node classification on each graph. The objective is to assign a label from a predefined set of labels $\mathcal{Y}$ to each node within a graph. This task can be formally described as finding a function $\sigma(\cdot) : \mathbb{R}^{d_h} \rightarrow \mathcal{Y}$, where k_h is the dimension of the node embeddings, and $\sigma(\mathbf{h}_{ij})$ provides the predicted label for node j in graph G_i. The objective of the problem is i) to learn a node mapping function $\phi(\cdot)$ which maps a node text to a vector by finetuning a pre-trained language model (note that node mapping function is pre-trained the language model itself if no finetuning is required), and ii) the GNN function $\sigma(\cdot)$ to optimize the accuracy of node classification across the set of graphs $\mathcal{G}$.

4.3 GLaM4CM Architecture

We start with a set of Graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ as shown in Fig. 2. Next, we extract the node data in the form of node paths and the node stereotype information as described in Sec. 4.1. Then, we prepare the training and testing dataset from the set of node paths and labels for the MLM and SC tasks. We first divide the nodes of each graph into 80% training and 20% test nodes. Note

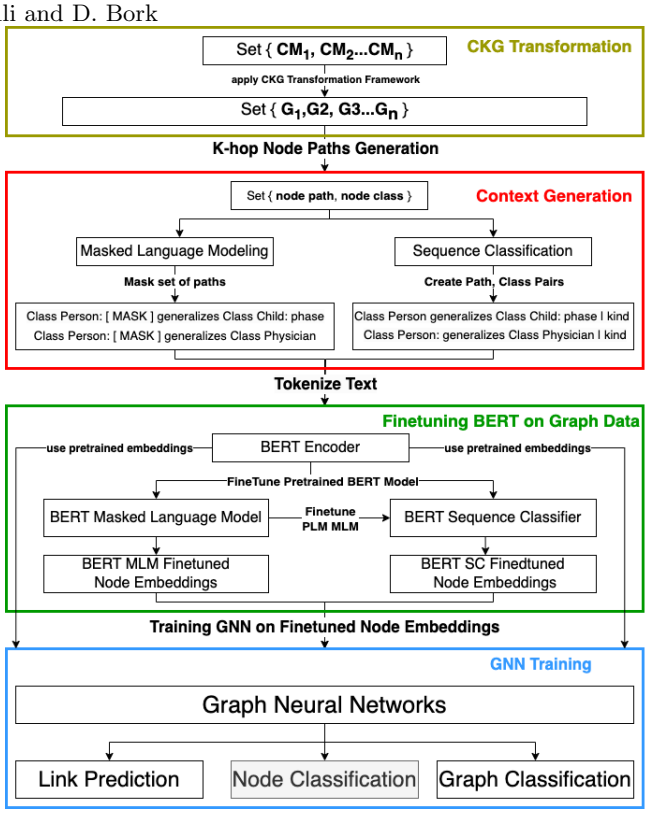

Fig. 2: Graph Language Modeling Framework for Conceptual Models

that the path string consists of node stereotype information of the neighboring nodes. Therefore, we only keep the node stereotype information if the context node is in a training set and remove the information if the context node is in a testing set to simulate a graph where 20% of the stereotype labels of a graph are unknown. For example, if in Fig. 1 we treat the "Child" node as a training node and the "Physician" node as a test node, Fig. 1 shows that the node path string of the "Person" node contains the "phase" information for the "Child" class but does not contain the "phase" information for the "Physician" class. In case of the MLM task, we mask the stereotype label of the training node such that the node data forms the format ⟨ node type ⟩ ⟨ node label ⟩: ⟨ [MASK] ⟩ and in case of sequence classification task, we remove the stereotype label from the node path string such that string follows a format ⟨ node type ⟩ ⟨ node label ⟩. An example of both formats can be seen in Fig. 2 for the class node "Person". The figure shows the expected stereotype "kind" after the node path string which forms the training labels.

Once we have the training and testing dataset, a pre-trained BERT model will be finetuned to the MLM and the SC tasks. After finetuning the BERT model for the MLM task, we can further finetune the BERT MLM model for the same SC task. This can be useful because the MLM task learns the contextual relationships between words in training data sequences and captures how each

Table 1: Stereotypes with a frequency greater than 100

Stereotype	Frequency	Stereotype	Frequency	Stereotype	Frequency
subKind	1522	category	461	derivation	195
kind	1268	event	396	participation	180
mediation	1264	characterization	357	datatype	132
role	1114	roleMixin	346	collective	129
relator	746	mode	333	type	128
material	554	phase	306	quality	128
componentOf	496	formal	272	memberOf	120

word relates to the entire sentence, thereby effectively modeling the nuances and subtleties of sequences in the training dataset. Afterward, another model can use this learned knowledge about contextual relationships in the OntoUML dataset to classify the text sequence. However, it is important to note that the effectiveness of using a two-step approach, i.e., first MLM finetuning and then classification, depends on the task and dataset and, therefore, does not guarantee better performance compared to directly using only a BERT for SC. After the finetuning, we can receive the node embeddings from the BERT models. Till now, BERT models were agnostic to the graph structural information of the CKG. Therefore, we now execute a GNN-based node classification using the finetuned BERT node embeddings.

5 Framework Evaluation

We now elaborate the experimental setup to evaluate the ontological stereotype prediction accuracy of GLaM4CM on a partially complete graph, i.e., when a subset of the CKG nodes do not have an OntoUML stereotype label.

5.1 Experimental Set Up

We use the OntoUML models dataset from [4] with 144 models and test our solution for three cases with respect to the number of classes. We select all the stereotypes that occur more than 1000 and 100 times across all the models, which gives us four and 21 stereotypes, respectively (cf. Table 1). We also select the eight stereotypes used by [3] to compare our approach against theirs. In this work, we show the results for 21 stereotypes classification case. Due to lack of space, we provide the complete set of results separately[†].

Our framework allows the extraction of node paths from the graph based on the hyperparameter k, which is the maximum number of hops between two nodes. We test our framework with the values $k = 1, 2, 3$. Note that the case $k = 0$ does not use any contextual information. Our framework produces node embeddings from four different models, namely, 1) pre-trained BERT model; 2) finetuned MLM model; 3) finetuned SC model; and 4) finetuned for first MLM and then SC model. Furthermore, we compare the quality of the node embeddings produced by the BERT models in our approach compared to three other GLoVE language models, namely, *i*) the GLoVE model from [1] trained

[†]https://bit.ly/onto-stp-cls

on OntoUML data; *ii*) GLoVE model from [19] trained on MDE and conceptual modeling research papers data; and 3) GLoVE model from [9] trained on data from the software engineeering domain. We use the generated node embeddings to train a GraphSage GNN model for the node classification task. We test our framework with different hyperparameter values, such as the type of GNN model, the number of GNN layers, and the learning rate (lr). After experimentation, we eventually chose the GraphSage model from [1], two GNN layers, and a $lr = 0.01$ as the best hyperparameter values for training. We perform a five-fold cross-validation to provide robust results.

In our experiments, we test the prediction accuracy of our ML models on the nodes in the test set in each graph for a set of graphs on which ML models are trained. In this configuration, the ML model has seen the graphs during training. However, we also test how well the trained ML model transfers its learning to unseen graphs because even though different OntoUML models may differ in their graph structure and the nodes' data, they may still share some common patterns that an ML model can learn to generalize even to unseen graphs. For example, given a generalization with two specializations like "MedicalProcedure" as a generalization of "RemoteMedicalProdure" and "PresenceMedicalProduce," the generalization can be a "kind" and the specialization can be a "subkind." Therefore, we split our set of 144 graphs into 90% seen and 10% unseen graphs and then calculate the prediction accuracy of the ML model on the test nodes in both seen and unseen graphs. It is important to note that the nomenclature "seen" does not mean the ML model has seen these graphs previously before training. In our experiments, "seen" simply means the graphs used to train the ML models with 80% of nodes as training nodes and 20% nodes as testing nodes.

Lastly, we compare our approach with the most recent existing rule-based inferencing approach [3] which infers the stereotype of the classes in an OntoUML model, given a percentage of classes already labeled. We compare our approach with the same setting of 80% of the nodes already labeled. They present their results in the form of *accuracy@n* metric that evaluates the average number of times the actual class y_i is present in the top-n predicted classes out of all predicted classes $\hat{y}$ for all N nodes defined as $\text{Accuracy@n} = \frac{1}{N}\sum_{i=1}^{N}\delta(y_i, \hat{y})$ where $\delta(y_i, \hat{y}) = 1$ if $y_i \in \hat{y}$ and 0 otherwise.

We note that even though we use the same dataset, stereotypes, and percentage of already labeled nodes (80%), a direct comparison has a shortcoming. Due to the unavailability of the information to reproduce the same graphs used in [3], our results hint at the comparative performance of the two approaches. Furthermore, they divide their model files into two sets—one that follows an Open World Assumption (OWA) and the other that follows a Closed World Assumption (CWA)—and provide the accuracy scores for both these approaches. Our approach is independent of such assumptions, and we compare our results with the average score of the two scores provided for both cases in [3].

5.2 Results

Performance on seen graphs For seen graphs, our approaches involving first getting embeddings from MLM and SC finetuning stages and then training a

Table 2: Different models accuracy comparison for top 21 stereotypes

Approach	Test Accuracy Seen				Test Accuracy Unseen			
	k=0	k=1	k=2	k=3	k=0	k=1	k=2	k=3
SE [9] + GNN	0.498	0.555	0.553	0.537	0.363	0.435	0.397	0.375
Onto [1] + GNN	0.574	0.634	0.618	0.599	0.356	0.422	0.379	0.356
MDE [19] + GNN	0.563	0.644	0.624	0.597	0.374	0.489	0.446	0.435
BERT + GNN	-	0.401	0.384	0.356	-	0.390	0.363	0.336
SC	-	0.801	0.816	0.792	-	0.646	0.625	0.609
MLM + SC	-	0.814	0.810	0.799	-	**0.656**	**0.636**	**0.634**
SC + GNN	-	**0.815**	**0.831**	**0.817**	-	0.634	0.625	0.621
MLM + SC + GNN	-	**0.827**	**0.822**	**0.813**	-	**0.661**	**0.643**	**0.627**

Table 3: Accuracy comparison with rule-based approach

Approach	accuracy@1	accuracy@2	accuracy@3
Rule-based [3]	0.181	0.303	0.917
SC + GNN	**0.820**	0.917	0.956
MLM + SC + GNN	**0.820**	**0.920**	**0.957**

GNN model using these embeddings outperforms the other approaches that involve non-finetuned embeddings (cf. Table 2). The results clearly show the positive impact of using finetuned embeddings on the contextual information available in the node paths. We show the combined importance of contextual information and pre-trained language model finetuning. Adding contextual information in the form of node paths increases the prediction accuracy for all the approaches, including those that do not involve finetuned embeddings—i.e., GLoVE and pre-trained BERT. Moreover, the accuracy of using finetuned embeddings is significantly higher than pre-trained embeddings. Interestingly, the BERT embeddings without finetuning (BERT + GNN) perform worse than embeddings from GLoVE models (SE, Onto, MDE). This result is very much in line with the fact that the pre-trained BERT model does not have any knowledge of OntoUML models' specific knowledge as it is trained on general natural language text. However, we see that finetuning the BERT model for MLM and SC produces rich contextualized embeddings that provide significantly higher accuracy, thereby outperforming the state-of-the-art [1] by more than **25%** (from 57.4% with Onto + GNN and $k = 0$ to 83.1% with SC + GNN and $k = 2$).

Next, we see that using GNNs improves the classification accuracy (cf. Table 2). These results are consistent with the idea that BERT language models learn contextualized node embeddings but do not directly or only implicitly capture graph structural information. In contrast, GNNs can better capture the graph's structural information. Finally, we also note the impact of contextual information in the paths and find that using $k = 2$ provides the best results. The accuracy decreases with $k = 3$, which indicates that for larger values of k, the neighborhood of the majority of the nodes ends up having a lot of common nodes, thereby reducing the distinguishing characteristic of a neighborhood.

Performance on unseen graphs We now evaluate how well the ML models transfer the learning to unseen graphs. In Table 2, the GNN models trained using node embeddings from pre-trained embeddings or GLoVE model embeddings do

not generalize well with a maximum accuracy of 48.9%. However, using BERT-based finetuning models improves the prediction accuracy by more than 20%, indicating that the finetuned embeddings generalize much better and capture latent features that support the ML model in predicting the node stereotype. We note that using GNN models improves the prediction accuracy for unseen graphs. However, the prediction accuracy does not improve as well as they did for seen graphs. Using GNNs with finetuned embeddings for seen graphs increased the prediction accuracy score by almost 2% (from 81.6% by SC and $k = 2$ to 83.1% by SC + GNN and $k = 2$). In contrast, in case of unseen graphs, the prediction accuracy increased by less than 1% in case of $k = 1, 2$ (from 65.6% by MLM + SC and $k = 1$ to 66.1% by MLM + SC + GNN and $k = 1$) and in fact decreased in case of $k = 3$. This result may be because the linguistic semantics are shared more across OntoUML graphs than the graph structural information.

Comparative evaluation with Rule-based Inference Lastly, we compare our approach with the rule-based approach in [3]. We choose the top two best results from all the configurations, and use the GNN-based classification on finetuned embeddings for seen graphs. Our approach significantly outperforms the rule-based approach that predicts the exact stereotype class in 18.1% and has the correct class as one of the top two predicted classes in 30% of the cases. We see that the rule-based approach provides a quite good accuracy**@3** score, i.e., the correct stereotype is amongst the top three predicted ones. Overall, we see that our approach outperforms the rule-based approach by more than **60%** in case of accuracy**@1** and accuracy**@2** and by around 4% in case of accuracy**@3**.

6 Discussion

In the following, we aim to discuss our results, the degree of automation provided by our work for modeling, the transferability aspects, i.e., how can a model trained for one domain transfer the learning for a different domain, and finally discuss some further applications of our framework.

Illustrative Example-based Results Analysis In Fig. 3, we show how GLaM4CM learned to correctly predict the stereotype from the data itself without the need for any explicit rules to infer the stereotype. The example uses the ontology from [10] that provides an ontological analysis of cyber-security cases. GLaM4CM was able to predict stereotypes accurately, notably identifying the "Upload Private Media Object to Digital Platform" as an "event" and "Accessible Digital Platform" as a "role." The correct "event" prediction could be attributed to the model's understanding of the term "Upload," highlighting the model's capability to recognize patterns beyond frequent stereotypes. Similarly, it distinguished "Accessible Digital Platform" as "role" instead of "subKind" by considering contextual clues from related classes. This indicates the model's ability to learn ontological rules, such as those from OntoUML. The shortcomings of a rule-based approach, as can be seen from our results, are that the rules are hardly exhaustive and, even if the rule-based approach does provide good top-k suggestions, such a benefit is not useful enough. Even if a rule-based approach

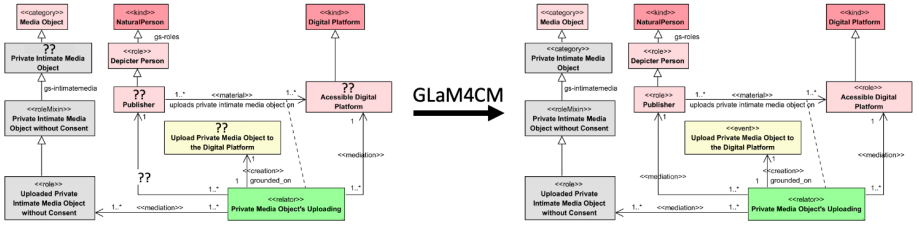

Fig. 3: GLaM4CM-based Ontological Stereotype Prediction Example

provides the suggestion for the top 3 possible stereotypes with high confidence, if the confidence on top 1 or top 2 is poor, these suggestions are not quite useful for the domain expert because the expert can, given adequate modeling experience, narrow down on the top-3 with some effort by herself.

Degree of Automation for ODCM The paper presents a modeling assistant, not for fully automating ODCM research, but as an ontological concept recommender aiding UML modelers in transitioning to OntoUML and assisting OntoUML modelers during model creation. Our approach does not replace the need for domain experts in UML structural class diagram modeling but supports them with recommendations based on data-driven patterns linking natural language to ontological semantics—for instance, identifying "Person" as "kind" and "Student" as "role" relative to "Person." While one could use rules to define these relationships, our comparative evaluation indicates that rule-based methods are outperformed by our data-driven approach.

Cross-domain Learning Transfer Despite having only 144 OntoUML models for training, our model's ability to make correct predictions in a zero-shot scenario—where it has not seen an example of the test input—is notably promising, with over 65% accuracy. This is particularly significant given that current research shows industry-standard large language models average 60-75% accuracy in zero-shot generalization across different domains [33,32]. Our approach achieving 65% accuracy aligns with these findings and signals a positive outlook on transferability rather than poor performance. The field is actively developing methods such as fine-tuning [31] and in-context learning [7] to enhance zero-shot generalization, and we plan to integrate these advancements into our approach. We are also confident that, as the adoption of OCDM increases and we get access to more ODCM models, the performance of our approach will even further improve.

Overall, we showed that our GLaM4CM framework combines pre-trained language models to learn contextually rich embeddings of OntoUML model nodes' data. It is important to note that our framework provides flexibility by being usable with *i*) different PLMs like BERT, distill-bert, LSTM, or GPT, among others, *ii*) different GNN models like GCN, GATConv, or GINConv among others, and *iii*) different modeling languages provided a CKG transformation exists as proposed in [2,25].

Further Applications Although this paper focuses on the accuracy of predicting OntoUML stereotypes using node classification, we want to elaborate on several application areas enabled by our GLaM4CM framework.

AI-based ODCM. GLaM4CM can be used to predict an ontologically sound metamodel element recommendations given a partially constructed metamodel.

Metamodel domain classification. CKG embeddings capture domain information using domain ontologies; therefore, these embeddings can be finetuned for classifying the domain of the metamodel. Metamodel domain classification can further support domain-based clustering of conceptual models.

Semantic Search. Learned representations of conceptual models can directly use the trained embeddings in semantic search for models. Semantic search can provide an efficient way of accessing and searching on and within these models, considering the ontological semantics of the queried model's semantics in search results. This would enable queries like *search all models that use UFO and that have a concept 'professor' with the stereotype 'role' assigned.*

Threats to Validity Our research is not exempted from the following threats to validity: *Conclusion Validity: Dataset size.* The training dataset for the experiment consists only of 144 OntoUML models. Each model belongs to a specific domain, and the labels consist of domain-specific information, which makes it difficult for the model to learn generalized patterns. We mitigated this by performing a five-fold cross-validation so that the ML models learn general patterns and not domain-specific ones. *Construct Validity: Node Path String Content.* The node path string of a node n contains stereotype information of its neighboring nodes. If the neighboring node of n is a testing node, then the stereotype information of the testing node should not be present in the node path string. We mitigated this threat by adding the stereotype label only if the neighboring node is a training node. *Internal Validity.* The risk of overfitting or suboptimal configuration can impact the internal validity of the findings. Therefore, we performed an exhaustive search for different GNN models, language models, context path lengths, and hyperparameters of GNN models to mitigate these issues. *External Validity.* To enhance the generalizability of the results beyond the specific dataset we used, the approach should be tested on a more diverse set of datasets, potentially including larger graphs with varying characteristics. We aim to explore a more diverse dataset in our future work.

7 Conclusion

In this paper, we presented GLaM4CM, a Graph Language Modeling framework for Conceptual Models and showed its genericity by using different ML models, different extents of contextual information during learning, and combining finetuned BERT models with GNN. We used the ontological stereotype prediction task for OntoUML models to experimentally evaluate the prediction accuracy for different configurations that analyze the impact of *i*) adding contextual information, i.e., neighboring nodes data, *ii*) the quantity of contextual information, i.e., experimenting with different path lengths with k hops, *iii*) the language modeling architecture used to capture the natural language features, i.e., pre-trained BERT-based language models and GLoVE models, *iv*) using GNNs on the learned linguistic features, and, finally, *v*) using the learned ML models on unseen graphs to evaluate the learning transferred to new graphs. Our extensive evaluation showed that GLaM4CM significantly outperforms other

GNN-based and rule-based approaches. In our future work, we aim to explore possibilities to integrate our stereotype predictor into either an existing OntoUML tool (`https://ontouml.org/` or a new web-based OntoUML modeling editor by extending the currently developed GLSP-based UML editor [20]. Furthermore, concerning GLaM4CM itself, we aim to apply GLaM4CM on several applications like (meta-)model classification or model completion. We also aim to create an accessible user interface for modelers to plug and play different modeling languages, PLMs or GNN models for different conceptual modeling tasks.

References

1. Ali, S.J., Guizzardi, G., Bork, D.: Enabling representation learning in ontology-driven conceptual modeling using graph neural networks. In: International Conference on Advanced Information Systems Engineering. pp. 278–294. Springer (2023)
2. Ali, S.J., Michael Laranjo, J., Bork, D.: A generic and customizable genetic algorithms-based conceptual model modularization framework. In: Int. Conference on Enterprise Design, Operations, and Computing. pp. 39–57. Springer (2023)
3. Barcelos, P., Prince Sales, T., Romanenko, E., Almeida, J., Engelberg, G., Klein, D., Guizzardi, G.: Inferring ontological categories of owl classes using foundational rules. In: Proceedings of the 13th International Conference on Formal Ontology in Information Systems (2023)
4. Barcelos, P.P.F., Sales, T.P., Fumagalli, M., Fonseca, C.M., Sousa, I.V., Romanenko, E., Kritz, J., Guizzardi, G.: A fair model catalog for ontology-driven conceptual modeling research. In: Int. Conf. on Conceptual Modeling. pp. 3–17 (2022)
5. Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E.M., Vieu, L.: Dolce: A descriptive ontology for linguistic and cognitive engineering. Applied ontology **17**(1), 45–69 (2022)
6. Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O.M., Antonyrajah, D., Horrocks, I.: Owl2vec*: Embedding of owl ontologies. Machine Learning **110**, 1813–1845 (2021)
7. Coda-Forno, J., Binz, M., Akata, Z., Botvinick, M., Wang, J., Schulz, E.: Meta-in-context learning in large language models. Advances in Neural Information Processing Systems **36** (2024)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint:1810.04805 (2018)
9. Efstathiou, V., Chatzilenas, C., Spinellis, D.: Word embeddings for the software engineering domain. In: int. conf. on mining software repositories. pp. 38–41 (2018)
10. Falduti, M., Griffo, C.: Modeling cybercrime with ufo: An ontological analysis of non-consensual pornography cases. In: International Conference on Conceptual Modeling. pp. 380–394. Springer (2022)
11. Fonseca, C.M., Porello, D., Guizzardi, G., Almeida, J.P.A., Guarino, N.: Relations in ontology-driven conceptual modeling. In: 38th International Conference Conceptual Modeling. pp. 28–42 (2019)
12. Grüninger, M., Atefi, K., Fox, M.S.: Ontologies to support process integration in enterprise engineering. Comput. Math. Organ. Theory **6**, 381–394 (2000)
13. Guizzardi, G.: The role of foundational ontologies for conceptual modeling and domain ontology representation. In: 2006 7th International Baltic conference on databases and information systems. pp. 17–25. IEEE (2006)

14. Guizzardi, G., Prince Sales, T., Almeida, J.P.A., Poels, G.: Relational contexts and conceptual model clustering. In: IFIP Working Conference on The Practice of Enterprise Modeling. pp. 211–227. Springer (2020)
15. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
16. Junior, A.G.L., Carbonera, J.L., Schimidt, D., Abel, M.: Predicting the top-level ontological concepts of domain entities using word embeddings, informal definitions, and deep learning. Expert Systems with Applications **203**, 117291 (2022)
17. Keet, C.M., Khan, M.T., Ghidini, C.: Ontology authoring with forza. In: Proceedings of the 22nd ACM international conference on Information & Knowledge Management. pp. 569–578 (2013)
18. Leão, F., Revoredo, K., Baião, F.: Extending wordnet with ufo foundational ontology. Journal of Web Semantics **57**, 100499 (2019)
19. López, J.A.H., Durá, C., Cuadrado, J.S.: Word embeddings for model-driven engineering. In: ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS) (2023)
20. Metin, H., Bork, D.: Introducing bigUML: A flexible open-source glsp-based web modeling tool for uml. In: Companion Proceedings of MODELS 2023. IEEE (2023)
21. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data & Knowledge Engineering **55**(3), 243–276 (2005)
22. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: empirical methods in NLP. pp. 1532–1543 (2014)
23. Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., Paulheim, H.: Rdf2vec: Rdf graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019)
24. Sequeda, J., Lassila, O.: Designing and building enterprise knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge **11**(1), 1–165 (2021)
25. Smajevic, M., Bork, D.: Towards graph-based analysis of enterprise architecture models. In: Int. Conference on Conceptual Modeling. pp. 199–209. Springer (2021)
26. Sun, S., Meng, F., Chu, D.: A model driven approach to constructing knowledge graph from relational database. In: Journal of Physics: Conference Series. vol. 1584, p. 012073. IOP Publishing (2020)
27. Trojahn, C., Vieira, R., Schmidt, D., Pease, A., Guizzardi, G.: Foundational ontologies meet ontology matching: A survey. Semantic Web **13**(4), 685–704 (2022)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
29. Verdonck, M., Gailly, F., Pergl, R., Guizzardi, G., Martins, B., Pastor, O.: Comparing traditional conceptual modeling with ontology-driven conceptual modeling: An empirical study. Information Systems **81**, 92–103 (2019)
30. Wand, Y., Weber, R.: Research commentary: information systems and conceptual modeling—a research agenda. Information systems research **13**(4), 363–376 (2002)
31. Wei, J., Bosma, M., Zhao, V.Y., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652 (2021)
32. Yang, H., Zhang, Y., Xu, J., Lu, H., Heng, P.A., Lam, W.: Unveiling the generalization power of fine-tuned large language models. arXiv preprint arXiv:2403.09162 (2024)
33. Zhang, X., Li, J., Chu, W., Hai, J., Xu, R., Yang, Y., Guan, S., Xu, J., Cui, P.: On the out-of-distribution generalization of multimodal large language models. arXiv preprint arXiv:2402.06599 (2024)