Accepted for CAiSE 2026 and publication in Springer LNCS.
This is the camera-ready author version of the paper, the final version will be made accessible via Springer.

Shortcut or Understanding? Diagnosing LLM Type Prediction in Conceptual Models

Syed Juned Ali[1 [0000−0002−0710−8052]], Zhuoxun Zheng[1 [0000−0002−9644−1219]], and Dominik Bork[1 [0000−0001−8259−2297]]

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

Abstract. Automated type prediction in conceptual models—such as ArchiMate and OntoUML—has emerged as a core capability of modern intelligent modeling assistants. Yet it remains unclear whether these systems genuinely leverage the semantic and structural richness of conceptual models or merely exploit surface-level regularities in modeling repositories. This paper presents a systematic investigation into how natural language labels, structural context, and different learning paradigms shape type-semantics prediction. Using a controlled dataset-generation pipeline, we create parametrized text datasets from large collections of enterprise architecture and OntoUML models by manipulating label semantics and structural context. We evaluate two contrasting families of approaches—pretrained BERT models finetuned for masked and supervised classification, and prompting-based LLMs using zero-shot and few-shot strategies. Our results indicate that natural language semantics are the primary driver of predictive performance, with structural cues providing complementary support only when meaningful labels are intact. Finetuned BERT models outperform LLM prompting when label quality degrades but structural regularities remain available, whereas prompting with local few-shot examples becomes highly competitive under severe label sparsity. These findings offer empirical guidance for future intelligent modeling assistants that would utilize hybrid architectures that combine structurally trained encoders with LLM-based semantic reasoning.

Keywords: Semantics · Finetuning · LLMs · OntoUML · ArchiMate.

1 Introduction

Conceptual modeling remains a cornerstone of information systems engineering, providing structured and semantically rich representations of business domains, organizational processes, and technological architectures. To support such models for downstream tasks including modularization, interoperability, and database design [1,2], modeling languages such as ArchiMate and OntoUML define semantically meaningful element types, such as *Business Process*, *Application Component*, or *Role*, that encode constraints and analytic properties. These types enhance model interpretation, determine valid connections, support viewpoints and simulations, and underpin reasoning tasks such as consistency

checking or compliance analysis [3,1]. In many real-world settings, these types are incomplete, inconsistently assigned, or only partially specified. As a result, semantic type prediction is a key capability in data-driven modeling assistants, supporting model completion and ensuring semantic consistency [4].

At the same time, large language models (LLMs), trained on vast amounts of text, have demonstrated impressive capabilities in natural-language understanding, contextual inference, and semantic classification [5,6]. This makes them suitable candidates for assisting with type semantics prediction in conceptual models [7,8]. These LLM-based models infer element types normally from element labels, descriptions, and surrounding structures, enabling modeling assistance with model completion. For example, existing works like [9,10] utilize the text labels in models for model classification and partial domain model autocompletion. The works reported in [11,10] further utilize the tree-based structure of model elements for model transformation and model elements recommendation.

LLMs have demonstrated strong performance and have become widely adopted for automated type prediction, however, despite their rapid adoption, there is fundamental uncertainty as to what LLMs actually learn from conceptual models. These models may succeed either because they genuinely understand the meaning of model elements or because they rely on superficial cues such as recurring graph patterns, repository-specific regularities, or memorized vocabulary. For example, an LLM may over-rely on structural information by classifying elements purely based on familiar subgraphs, such as assuming that any node embedded in a pattern typical of ArchiMate's Application Layer must be an Application Component. Conversely, it may over-rely on textual information by assigning types solely from surface keywords in labels, inferring that any element containing the word *process* must belong to the corresponding category regardless of its actual role in the model. These behaviors raise an important concern: ***Do LLMs truly understand conceptual models, or do their predictions instead reflect shortcut learning?*** Clarifying this distinction is crucial for guiding the use of LLMs in practice—if their performance depends on superficial cues, we must invest in better data quality and structural consistency, whereas if they capture deeper conceptual distinctions, they can be trusted to operate effectively even on noisy or partially specified models.

This paper presents a systematic investigation into the respective contributions of natural language labels and structural context to ontological type prediction. In particular, we focus on the following five research questions:

- **[RQ1] Text Quality**: To what extent do LLMs rely on the natural-language quality of element labels when predicting ontological types?
- **[RQ2] Structure Quantity**: How does the graph structure, in terms of the edges linking model elements, affect type prediction performance?
- **[RQ3] LLM Usage Paradigm**: How consistent are the effects across different LLM usage paradigms, namely, finetuning smaller LLMs like BERT versus prompting LLMs like GPT-4o?
- **[RQ4] LLM Prompting Effect**: How do different prompting techniques affect the prediction performance?

Our comprehensive evaluations reveal several consistent patterns. Lexical semantics are the primary signal for type prediction, with structural information offering only limited complementary value when meaningful labels are present. Structural sparsity alone has little independent effect, suggesting that many structural patterns in conceptual models are highly redundant. We also observe a clear distinction between LLM usage paradigms: fine-tuned BERT models perform best when some type information remains available, whereas prompting large LLMs becomes competitive only when models are heavily incomplete and label signals deteriorate. These observations highlight that different LLM strategies excel under different conditions and motivate a more nuanced understanding of how text and structure jointly shape type prediction.

Building on these observations, this paper makes the following contributions to the understanding and application of LLMs in conceptual modeling:

- A controlled dataset-generation framework, that systematically varies textual and structural conditions across large ArchiMate and OntoUML repositories.
- A comprehensive empirical evaluation, that isolates the effects of label quality, neighboring information and graph completeness on LLM behavior.
- A comparative analysis of LLM usage paradigms, contrasting fine-tuned pretrained models (BERT) with LLM Prompting
- A comparative analysis of LLM prompting techniques to show how sensitivity to textual and structural signals varies across these paradigms.

It is important to clarify the notion of "understanding" used by our study as there exists a longstanding question whether highly accurate statistical predictors can be said to "understand" what they predict. In a recent debate Mitchell et al. [12] argue that accurate statistical prediction alone should not be equated with understanding. Therefore, in context of this work, we define *understanding* by LLMs as the ability of a system to produce contextually appropriate outputs by internalizing large-scale statistical regularities in data and learning extensive statistical correlations between linguistic tokens. This notion differs from human understanding, which is typically associated with causal models, grounded experiences, and the ability to reason and explain beyond observed correlations.

The paper is organized as follows. Sec. 2 reviews prior work on machine learning for model-driven engineering (ML4MDE) emphasizing explainability, which highlights the gap in understanding why these methods work and motivating the need for our systematic analysis. Sec. 3 and Sec. 4 present our methodology and empirical results for addressing the research questions. Sec. 5 discusses the implications of our findings, and Sec. 6 concludes the paper.

2 Related Work

Research in ML4MDE has grown substantially in recent years [13,14]. Existing works use a wide variety of signals contained in conceptual models—textual labels, modeling-language types, structural patterns, or combinations thereof to develop techniques for modeling tasks such as model clustering, classification, recommendation, or completion. However, while these works utilize both natural language semantics and structural information, none of them examines how the

Table 1: Semantic and Structural Signals in existing ML4MDE without LLMs

Paper	NLS	MLS	GS	Modeling Task
Strüber et al. [15]	✗	❖	✓	Model Clustering
Elkamel et al. [16]	✓	❖	✗	Model Clustering
Basciani et al. [17]	✗	✓	✗	Model Clustering
Babur et al. [18,19]	✓	✓	✗	Model Clustering
Rubei et al. [20]	✗	✓	✗	Model Classification
Nguyen et al. [21]	✗	❖	✗	Model Classification
Khalilipour et al. [22]	✗	✗	✓	Model Classification
Lopez et al. [23]	✓	✗	✗	Model Classification
Di Rocco et al. [24]	✗	❖	✗	Partial Model Completion
Di Rocco et al. [25]	✗	✗	✓	Partial Model Completion
Burgueno et al. [11]	✓	❖	❖	Model Transformation

quality of these signals or their independent contribution affects performance. This gap is central to the motivation of this paper.

To contextualize our investigation, we analyze the semantic and structural signals used across ML4MDE research. We extract three dimensions from the literature— (*i*) **Natural-Language Semantics (NLS)** captured from textual labels through pretrained word embeddings; (*ii*) **Modeling-Language Semantics (MLS)** such as element types or relationship types; (*iii*) **Graph Structure (GS)** referring to structural information from the model graph;

We first summarize the non-LLM ML4MDE approaches, then the LLM-based ML4MDE approaches involving training, and finally cover the recent LLM4MDE approaches that do not involve model training but instead use prompting techniques to realize LLM-assisted modeling.

2.1 Non-LLM ML4MDE Approaches

Table 1 summarizes approaches that use classical or small ML models without relying on a pretrained LLM. These works collectively demonstrate that ML4MDE has explored natural language, modeling language primitives, and graph structure, but rarely in conjunction. Across several tasks like model clustering, classification, model completion, and model transformation, non-LLM works reveal, *i*) **Natural language semantics (NLS) are often underused.** Most works rely on static embeddings (tf-idf, word2vec) rather than contextual semantics from pretrained language models like BERT, *ii*) **Modeling-language semantics (MLS) are inconsistently incorporated.** Only a subset uses type information, despite its crucial role in defining the semantics of conceptual models, and *iii*) **Graph structure (GS) is used sparsely** and **never in combination** with NLS and MLS.

2.2 Pretrained LLM-based ML4MDE Approaches

Table 2 shows works that explicitly use LLMs in the learning pipeline (finetuning LLMs or using LLM embeddings). These works exploit the semantic capacity of the pretrained LLMs but do not analyze the semantic or structural signals encoded in the models' data. Table 2 further shows that LLM-based ML4MDE works *i*) primarily rely on **natural-language semantics**, *ii*) often underuse

Table 2: Semantic and Structural Signals in existing ML4MDE involving LLMs

Paper	NLS	MLS	GS	Modeling Task
Lopez et al. [26]	✓	✗	✗	Model Classification
Lopez et al. [23]	✓	✗	✗	Model Classification
Weyssow et al. [10]	✓	✓	❖	Partial Model Completion
Burgueno et al. [27]	✓	❖	✗	Partial Model Completion
Ali et al. [4]	✓	✓	✓	Partial Model Completion

or ignore **modeling-language semantics**, and *iii*) rarely incorporate **graph structure**. More importantly, these works treat the training data as static, without investigating the role of data quality or how degrading or enriching label quality, neighborhood size, or graph completeness affects predictions. Thus, despite using the natural language understanding capabilities of pretrained LLMs, existing works do not explain why these models perform well (or fail).

2.3 Generative LLM-based ML4MDE approaches

A separate line of work uses LLMs purely in a generative/prompting mode (without pretraining) to create or complete domain models from textual descriptions [7,8]. These works demonstrate the feasibility of using LLMs for domain modeling, but significantly differ from ML4MDE. They rely on the LLM's pre-existing knowledge of well-known modeling languages (usually UML or ER). They do not incorporate modeling-language semantics beyond what the LLM already knows. They assume simple metamodels (classes, attributes, relations) without rich type semantics. Thus, while generative LLM4MDE works demonstrate the usability of prompting with LLMs, they do not address learning from conceptual models, nor do they investigate how semantic signals impact modeling tasks.

2.4 Synthesis and Research Gap

Across both traditional ML4MDE and LLM-based ML4MDE, the literature demonstrates that natural language, modeling language types, and structural patterns are all useful signals, and different works utilize different combinations of these signals. However, a lack of focus on data quality and over-reliance on the technique used obscures the source of predictive power, limitations, and potential of these techniques. Our work addresses this gap through a controlled experimental framework that disentangles semantic and structural signals for modeling-language semantics prediction in ArchiMate and OntoUML.

3 Methodology

Next, we elaborate our experimental setup with the pipeline illustrated in Fig. 1.

3.1 Model-to-Knowledge-Graph Transformation

All conceptual models in the corpus are first converted into typed knowledge graphs (KGs), providing a uniform abstraction across heterogeneous source formats such as XML, XMI, or proprietary model files. The KG representation

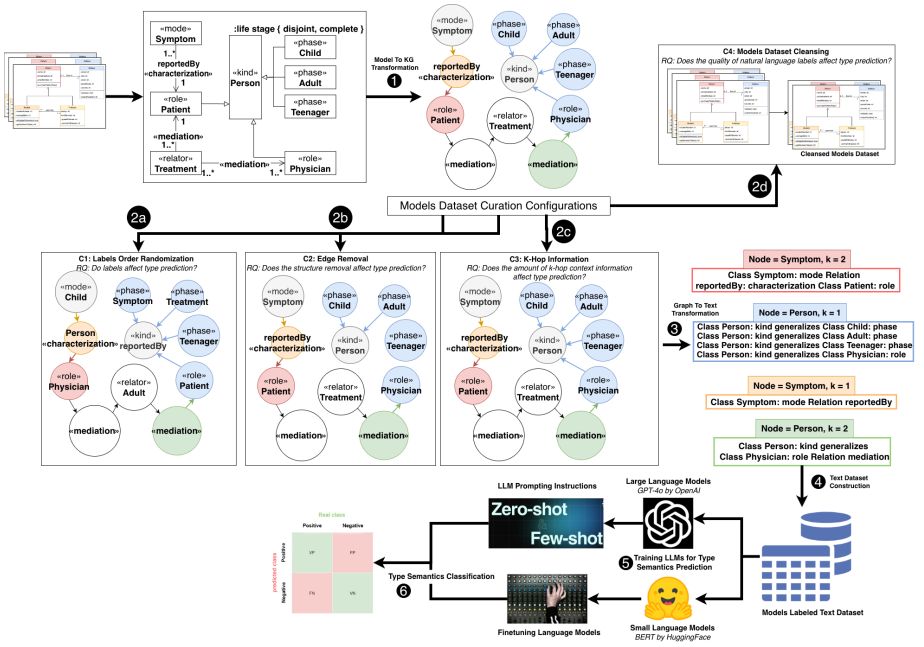


Fig. 1: Methodology for Type Semantics Prediction Across Configurations: 1. Extracting KG representations of models; 2. KG curation (a-d indicate different curation methods); 3. KG to text transformation; 4. Text construction; 5. LLM-based prediction in finetuned BERT and promted paradigms; 6. Prediction evaluation.

captures the core modeling constructs—nodes, edges, labels, types, and stereotypes—in a consistent, machine-readable form. Each model element becomes a node annotated with its name, modeling-language type (when available), and relevant metadata. Relationships are mapped to directed, typed edges that retain ArchiMate or OntoUML semantics. This transformation preserves the model's topology while enabling algorithmic access to neighborhoods, paths, and other structural features. Working at the KG level exposes the two key aspects for type semantics prediction: natural-language labels and structural context (node and edge types, graph topology) that we can manipulate independently—for example, randomizing labels while keeping structure intact, or removing edges without altering linguistic attributes. This separation is crucial for evaluating whether predictive models rely primarily on natural-language semantics, structural properties, or their interaction.

3.2 Controlled KG Dataset Curation

After converting models into KGs, we apply controlled manipulations that selectively modify linguistic or structural signals. These produce parallel variants of each model representing distinct experimental conditions.

Manipulating Label Semantics: Cleansing and Randomization. Label quality varies significantly across repositories, so we address this issue in two ways.

First, *label cleansing* removes models with low-quality names (e.g., placeholders or dummies) using the cleansing pipeline of [28], yielding a subset with reliable natural-language semantics. Second, *label randomization* shuffles labels across elements within or across models, preserving the global word distribution while destroying true associations. This eliminates linguistic cues without altering graph structure, isolating the contribution of natural-language semantics.

Varying Neighborhood Radius. Modeling semantics often arise from local context rather than isolated elements (e.g., an Abstract class is always connected to an edge of "generalization" type). Therefore, we manipulate the neighborhood radius used for constructing textual descriptions. For a given node, a radius of $k = 0$ produces a textual sample consisting solely of the node's label. At $k = 1$, the sample includes the node's label plus the labels and relationship types of immediate neighbors. Increasing the radius captures more structural and semantic information, allowing neighboring elements to clarify the role of the focal element. For instance, an element labeled "Register Patient" may be ambiguous when isolated. When surrounded by "Admission Service" or "Patient Record," its type becomes more predictable.

Structural Perturbation through Edge Removal. To evaluate the role of structure independently of text, we deliberately perturb the graph topology by removing a fraction of edges. Edge removal weakens the coherence of element neighborhoods, reduces the density of structural properties, and diminishes the availability of relational cues. This serves as a controlled stress test, specifically to determine whether a model relies heavily on structural patterns; if so, performance should degrade under edge removal. Conversely, models capable of interpreting natural language semantics may remain robust even as the structure degrades.

3.3 Graph-to-Text Conversion

Once the curated KG variants are produced, each element is transformed into a textual description suitable for input into a language model. This ensures that finetuned BERT and prompted LLMs can operate on text with structural information encoded with the k-hop neighboring information. The textual description of an element at radius k includes—*i*) the element's own label, *ii*) labels of adjacent nodes up to k-hops, and *iii*) relationship types between the element and its neighbors. These descriptions are rendered as simple declarative sentences as shown in Fig. 1. This approach transforms graph-derived structure and textual attributes into a unified linguistic format that can be processed by both BERT encoders and Generative LLMs. Because all dataset variants follow the same representation template, differences in model performance can be attributed solely to differences in available semantic or structural cues.

3.4 Large Language Models for Type Prediction

We evaluate two LLM-based paradigms for predicting ArchiMate nodes and edges types and OntoUML stereotypes—(*i*) finetuning pretrained encoders and (*ii*) prompt-based classification.

(*i*) *Finetuning Pretrained Encoders.* Encoder-only transformers, such as BERT [29,30], learn contextual linguistic representations that can be adapted to domain-specific tasks through supervised fine-tuning. In our setting, node labels and their local neighborhoods are linearized into textual sequences and fed into a pretrained BERT encoder. A classification head is added to predict the correct modeling language type, and the model is trained using cross-entropy loss with early stopping. Finetuning enables the encoder to learn repository-specific terminology and structural regularities, providing strong performance when sufficient labeled examples are available.

(*ii*) *Prompting Large Language Models.* Large LLMs such as GPT-4o perform type prediction through natural language prompts rather than parameter updates [31]. Prompting treats the task as a semantic inference problem grounded in the model's broad linguistic priors. We evaluate three prompting modes:

Zero-shot prompting. The LLM receives the element description and natural language definitions of all candidate types, without labeled examples [32]. This test determines whether the model can infer modeling semantics solely from general linguistic knowledge. *Global few-shot prompting.* The prompt includes very few labeled examples sampled across the repository [5]. These examples expose the LLM to common naming practices and text–type correlations, enabling a repository-wide mapping. *Local few-shot prompting.* Examples are drawn from the same model as the test instance [33]. This reflects real modeling settings where teams use project-specific terminology. Prior results [34,35] show that local demonstrations significantly improve domain adaptation; our experiments test whether similar benefits arise for type-semantics prediction. Together, these two paradigms contrast parameter-efficient specialization (finetuning) with flexible, inference-only adaptation (prompting), allowing us to evaluate how LLMs exploit linguistic and structural cues under varying levels of supervision.

The mapping between experimental conditions and research questions is explicit. Label cleansing and randomization address RQ1, while neighborhood radius and structural completeness address RQ2. The comparison of BERT Encoder finetuning and Generative LLMs, and global versus local prompting, addresses RQ3 and RQ4. Observed performance deltas provide empirical evidence about each of the configurations.

4 Evaluation

Next, we provide details of the datasets used and the experimental results of our evaluation across all fine-tuning configurations and prompting configurations. We organize the findings around the RQs and report descriptive statistics, inferential tests, and qualitative observations to provide a comprehensive picture of how semantic and structural signals affect type prediction performance.

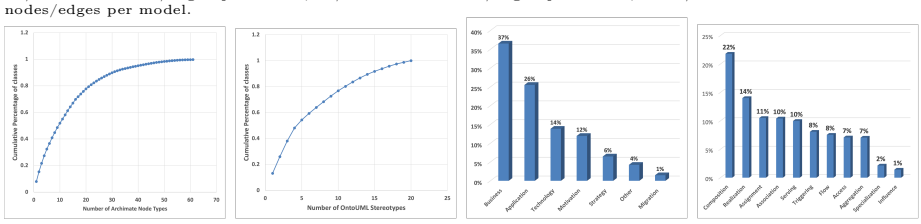
4.1 Datasets and Experimental Setup

We use 175 OntoUML models from the FAIR OntoUML dataset [36] and 936 ArchiMate models from the EAModelSet [37] to conduct our empirical evalu-

Table 3: Summary of Dataset Statistics After Filtering

Dataset	Models	Nodes	Edges	$\overline{N}$	$\tilde{N}$	$\overline{E}$	$\tilde{E}$	StdN / StdE
EAModelSet [37]	574	58,174	70,831	101	43	123	55	252.23 / 302.06
OntoUML [36]	174	15,815	20,100	91	65	115	79	96.60 / 140.37

$\overline{N}/\overline{E}$: mean nodes/edges per model; $\tilde{N}/\tilde{E}$: median nodes/edges per model; StdN/StdE: standard deviation of nodes/edges per model.



(a) Archi Node Type (b) OntoUML Type (c) Archi Layer (d) Archi Edge Type

Fig. 2: Type distributions across ArchiMate and OntoUML datasets.

ations. Before constructing CKG representations, we remove exact duplicates and exclude models with fewer than five edges. This decision follows manual inspection, indicating that very small models rarely encode meaningful structural or semantic content and tend to introduce noise in downstream learning tasks. While more advanced filtering criteria could be employed using automated pipelines for conceptual model cleansing [28][1], such measures were not required for the present study. Table 3 summarises the resulting datasets after filtering.

Both datasets exhibit substantial skew in their type distributions, a characteristic that significantly shapes the difficulty of the type–semantics prediction task. In ArchiMate, edge types such as *Composition*, *Realization*, and *Assignment* predominate, whereas relations like *Influence* and *Specialization* are relatively rare. Similarly, node occurrences are concentrated in the *Business* and *Application* layers, forming a frequency profile in which a handful of element types are highly prevalent while many appear sparsely. A cumulative frequency plot of ArchiMate node types further highlights the long-tail nature typical of enterprise architecture models. OntoUML displays a comparable pattern: stereotypes including *Subkind*, *Kind*, *Mediation*, and *Role* constitute the majority of class occurrences, while more specialized UFO categories—such as *Participation*, *Derivation*, and *Collective*—occur infrequently. Fig. 2 provides an overview of the key type distributions across the ArchiMate and OntoUML datasets.

We used BERT encoder from Google for the small LLM experimentation and GPT-4o as the prompting-based LLM. The code for the detailed experiments, together with the prompts used in our study, is open-sourced in our GitHub repository. [2]. It is essential to note that the primary objective of this study was not to develop the highest-performing type-prediction system, but to identify and understand the causal impact of specific semantic and structural factors. The experimental manipulations were designed to function as independent variables, not as optimization techniques. This focus on explanatory insight rather than performance maximization enables the study to disentangle how natural

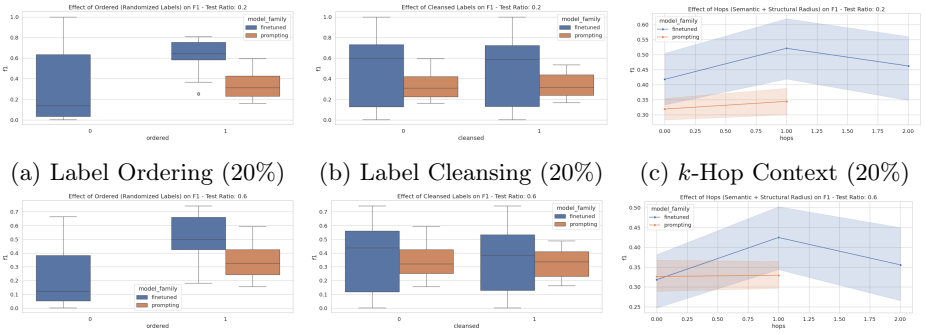
[1] Model Cleansing GitHub Repository
[2] Archi Type Prediction GitHub Repository

language signals, structural context, and learning paradigms contribute to the prediction of type semantics in conceptual models.

4.2 Results

This section reports the empirical findings of our controlled study across two masking regimes: a *low masking* setting with 20% test nodes (80% labeled nodes) and a *high masking* setting with 60% test nodes. These regimes allow us to systematically vary the amount of semantic and structural information available to the models. The 20% condition mirrors BERT-style masked learning [29], whereas the 60% condition induces severe information sparsity. We report aggregated metrics, inferential statistics, and cross-model comparisons to address each research question. We use Analysis of Variance (ANOVA) [38] tests to examine how much variance is explained by experimental factors versus random noise. ANOVA testing produces an F-statistic, which measures how strongly a given experimental factor (e.g., label ordering) affects the F1-score.

RQ1 — To What Extent Do Natural-Language Labels Influence Performance?
Effect of Label Randomization. The results indicate that label order plays a major role in semantics prediction. ANOVA shows a strong main effect of `ordered` under both masking levels: $F = 12.93, p < 0.001$ (20%) and $F = 45.41, p < 10^{-9}$ (60%). F1-scores drop substantially when labels are randomized, suggesting that models depend heavily on lexical regularities (naming patterns, syntactic forms, and implicit ontological cues). The effect becomes stronger under high masking, where the loss of structure makes natural-language cues more influential. Fig. 3 shows that finetuned models degrade sharply when label order is destroyed, while prompting exhibits a smaller but still noticeable decline.
Effect of Label Cleansing. Across both masking regimes, the `cleansed` factor has non-significant ANOVA effects ($F = 0.067, p = 0.796$ at 20%; $F = 0.453, p = 0.501$ at 60%). Boxplots in Fig. 3b–3e are nearly identical. This outcome does not imply that cleansing is irrelevant; rather, the filtering step removed only 36 models, 780 nodes, and 806 edges—too small a change to shift the overall distribution. The datasets appear already relatively clean, with stable naming conventions and few placeholder or malformed labels, meaning the cleansing step did not noticeably alter the lexical signal.
Effect of k-Hop Expansion. Expanding the semantic–structural radius from $k = 0$ to $k = 1$ yields significant improvements ($F = 6.91, p < 0.01$ at 20%; $F = 5.84, p < 0.02$ at 60%), as shown in Fig. 3. Immediate neighbors provide useful contextual cues—layer hints, decomposition structures, and mediation patterns—that benefit both finetuned models and prompting-based LLMs. However, $k = 2$ introduces semantic dilution as additional text and structure become less specific, reducing clarity and lowering performance (Fig. 3c–3f).
Interaction Between Ordered Labels and Hops. A strong interaction effect appears in both regimes ($F = 11.27, p < 0.001$ at 20%; $F = 17.65, p < 10^{-4}$ at 60%). With intact lexical semantics (`ordered`=1), increasing k noticeably



(a) Label Ordering (20%) (b) Label Cleansing (20%) (c) k-Hop Context (20%)

(d) Label Ordering (60%) (e) Label Cleansing (60%) (f) k-Hop Context (60%)

Fig. 3: RQ1 results across semantic and structural manipulations. Top: 20% masking. Bottom: 60% masking.

improves F1 up to $k = 1$. With randomized labels (`ordered`=0), the same expansion offers little benefit and can even reduce performance. Figs. 3c–3f clearly show this pattern: ordered labels produce rising trajectories, whereas randomized labels flatten or decline. These findings suggest that structural context is most helpful when meaningful text anchors are preserved; without them, additional structure may amplify noise. Finetuned models show the largest interaction effect due to learned label–structure correlations, while prompting follows the same trend with a smaller magnitude.

Response to RQ1 LLMs rely strongly on the natural-language quality of element labels for type prediction. Poor label semantics significantly drop performance, whereas label cleansing has a little effect if the dataset are already relatively clean.

RQ2 — What Is the Impact of Graph Structure? In the following, we show the impact of two structural manipulations: *edge removal* (sparsifying the topology) and *radius expansion* (capturing wider neighborhoods).

Effect of Edge Removal. Removing 20% of edges consistently lowers performance, but only slightly. ANOVA finds no significant main effect at either masking level (20%: $F = 0.05, p = 0.82$; 60%: $F = 0.16, p = 0.68$). This indicates that modest sparsification is insufficient to disrupt the models' ability to exploit remaining structural cues—particularly since many ArchiMate and OntoUML relations are redundant or hierarchically constrained.

Interaction Between Edge Removal and Hops. Fig. 4 shows that once $k = 1$ context is provided, performance becomes nearly insensitive to missing edges. Expanded semantic–structural neighborhoods compensate for the removed topology because enough relational evidence remains accessible. The interaction is statistically non-significant (20%: $F = 0.02, p = 0.89$; 60%: $F = 0.07, p = 0.79$), confirming that edge sparsity and neighborhood radius operate largely independently.

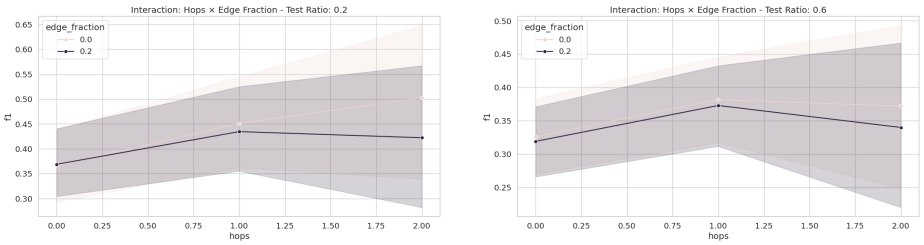


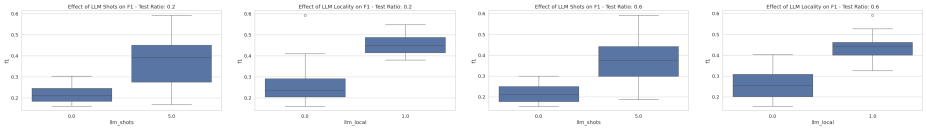
Fig. 4: RQ2: Interaction of edge sparsity with neighborhood radius at 20% (left) and 60% (right).

Answer to RQ2 Graph structure alone provides only limited complementary value for type prediction. Immediate neighborhood context improves performance when meaningful labels are present, but structural sparsity alone has little independent effect.

RQ3 — How Does Finetuning Compare to Prompting? We compare finetuned BERT models with prompting-based LLMs using aligned configurations across dataset, target type, masking regime, edge removal, and k-hop radius, and using each prompting configuration's best-performing variant. When 80% of nodes are labeled, finetuning outperforms prompting: the mean difference is -0.105, with a significant Wilcoxon test ($p = 0.011$). Under 40% supervision, the gap collapses: ΔF1 $= -0.008$ with $p = 0.79$. Prompting becomes competitive and occasionally surpasses finetuning, indicating that *finetuning is highly dependent on abundant labeled examples*, while prompting retains robustness under label scarcity.

Answer to RQ3 Finetuned BERT performs better when sufficient labeled information is available, whereas prompting-based LLMs become competitive when supervision is sparse and type information is heavily missing.

RQ4 — How Do Prompting Configurations Influence Performance? Fig. 5 shows that prompting is highly sensitive to configuration. The results show that providing five in-context examples yields significant gains: $F = 9.47, p = 0.003$ (20% masking) and $F = 16.53, p < 0.001$ (60%). Improvements range from 5–15 F1 points, demonstrating that LLMs rely on concrete examples to ground modeling language semantics. Local few-shot prompting. Examples drawn from the same model repository provide the strongest boost: $F = 78.80,\ p < 10^{-11}$ for 20% masking and $F = 71.18,\ p < 10^{-11}$ for 60% masking. Local examples capture organization-specific and repository-specific naming patterns, allowing prompting to sometimes match–or exceed–the corresponding fine-tuned models. This suggests that large LLMs possess general linguistic priors but still require domain-calibrated exemplars to effectively interpret modeling semantics.



(a) Shots (20%) (b) Locality (20%) (c) Shots (60%) (d) Locality (60%)

Fig. 5: Prompting configurations for zero-shot and few-shots prompting and global and local prompting

Answer to RQ4 Prompting configuration matters substantially for prediction quality. Few-shot prompting consistently improves performance over zero-shot, and local few-shot examples from the same repository provide the strongest gains by better capturing domain-specific naming patterns.

Across all experimental configurations, our results show that type semantics prediction in conceptual models is driven most strongly by the quality of natural language labels, moderately by structural context, and only weakly by graph sparsity. Label randomization produces the largest performance drop in both finetuning and prompting settings, and the effect grows under high masking, demonstrating that linguistic regularities are the primary semantic signal. k-hop expansion improves performance only when labels remain intact, revealing a strong ordered×hops interaction where structural cues become informative only when anchored in meaningful text. Edge removal alone has no significant effect, as neighborhood expansion compensates for missing links. Finetuning outperforms prompting when 80% of labels are available, but loses its advantage when supervision is scarce, with prompting becoming statistically indistinguishable at 60% masking. Within prompting, few-shot examples substantially enhance F1, and local few-shot examples—drawn from the same repository—produce the largest gains, sometimes matching or even exceeding those of finetuned models. Collectively, the findings show that (*i*) natural language is the dominant carrier of type semantics, (*ii*) structure amplifies but cannot replace semantic cues, and (*iii*) prompting with domain-specific examples offers a robust alternative to finetuning under limited supervision.

5 Discussion

This section synthesizes the findings, highlights their implications, and discusses their relevance for ML4MDE and the design of semantics-aware modeling tools.

The results indicate that natural language labels—particularly domain nouns, layered terminology, and implicit ontological cues—carry most of the discriminative signal for type prediction. The text provides the semantic anchor, while structural cues enrich it but rarely replace it. Both channels contribute, but neither is sufficient on its own. This suggests that ML4MDE workflows should prioritize maintaining natural language labels.

Structural manipulations (edge removal and k-hop expansion) show that graph topology supports prediction but plays a secondary role. Removing 20%

of edges caused limited degradation, suggesting structural redundancy in conceptual models. This is consistent with prior work on recurring modeling patterns [39,40,41,42]. Structure contributes meaningfully only when semantic anchors remain intact. Structural reasoning helps only when semantics are interpretable, challenging approaches that rely primarily on topology (cf. Table 1)

Finetuning performs best with ample supervision, learning repository-specific naming habits and structural conventions. Its advantage diminishes as labels become scarce. By contrast, prompting remains more stable under low-label conditions, benefiting from broad linguistic priors and contextual demonstrations. RQ4 shows that LLMs do not infer modeling semantics purely from pretraining; they require targeted examples to align to modeling language types. Prompting thus acts as an adaptive mechanism, while finetuning offers specialization.

Do LLMs truly understand conceptual models? Our findings suggest that current finetuned BERT encoders and LLMs with prompt instructions exhibit a partial understanding of conceptual models, strong enough to exploit well-formed natural language labels and familiar structural patterns, but not yet robust enough to generalize when these cues are degraded. Their predictions reflect a semantic sensitivity (e.g., recognizing layered terminology, utilizing contextual 1-hop neighborhoods semantics, leveraging domain nouns) and shortcut behavior (e.g., over-reliance on naming conventions, dependence on repository-specific regularities, difficulties interpreting structure without lexical anchors). In other words, LLMs appear to form shallow but useful approximations of modeling language semantics, rather than internalizing deeper conceptual distinctions or rule-based constraints. This view does not diminish the practical value of LLMs; instead, it clarifies where their strengths can be leveraged and where they are likely to fail. More importantly, the results point to concrete research directions for advancing ML4MDE toward genuine conceptual understanding—curating datasets that reduce lexical shortcuts and encourage models to rely on deeper semantic distinctions, developing training approaches that explicitly disentangle textual and structural cues, and designing evaluation setups that deliberately break surface regularities to test whether models have learned the underlying modeling constructs. Together, these directions encourage a shift from shortcut-driven performance to approaches that systematically verify and enable true semantics-aware behavior in ML-based modeling assistants.

Role of Pragmatics. While our study isolates lexical and structural cues, the meaning of a modeling element in practice is often also shaped by pragmatics, that is, by the modeling purpose, project scope, organizational conventions, and stakeholder assumptions under which the model was created. For example, the same or very similar label may be assigned different types depending on whether the model is intended for business analysis, application landscape planning, or implementation-oriented documentation. Likewise, repository-specific conventions may encode tacit assumptions that are not visible in the label text or immediate graph neighborhood alone. Our results therefore should not be interpreted as showing that type semantics are fully recoverable from lexical and

structural signals alone; rather, they show how strongly current approaches depend on these observable cues when pragmatic context is absent. Incorporating such context in future work would help approximate more faithfully how type assignments are made in real modeling practice.

Threats to Validity Following Wohlin et al. [43], we outline key validity considerations. *Internal validity:* Artificial manipulations (label shuffling, edge removal) isolate causal effects but may not fully reflect real-world model degradation. Richer datasets with organic inconsistencies (e.g., [44]) can improve ecological realism. *External validity:* The curated datasets limit generalizability to noisier industrial repositories, where labels are often abbreviated, cryptic, or organization-specific. Because our repositories are comparatively clean, the reported scores may overestimate performance in such environments; however, the strong degradation under label randomization suggests that the main conclusion—namely, that lexical quality is a dominant driver of type prediction—would likely still hold. However, we do aim to conduct experimentation in industrial settings in the future extension of this analysis. *Construct validity:* Our operationalization focuses on lexical and structural cues, omitting pragmatic and requirements-driven semantics. *Conclusion validity:* Statistical tests (ANOVA, Wilcoxon, mixed-effects models) support robustness, but subtle interactions may require larger and more heterogeneous datasets.

6 Conclusion

In this paper, we examined whether LLM-based type prediction reflects genuine conceptual understanding or is a shortcut based on surface-level cues. Across extensive experiments on ArchiMate and OntoUML models, we find that natural-language labels overwhelmingly drive predictive accuracy, while structural cues provide only limited support—and only when meaningful labels remain intact. Moderate graph sparsity has little effect, reflecting redundancy in recurring modeling motifs. Finetuned BERT models outperform prompting-based LLMs when some semantic signal is preserved, but under severe label loss, prompting with local few-shot examples closes the gap, albeit at uniformly low accuracy. These patterns consistently indicate that current models rely on lexical shortcuts and repository regularities rather than internalizing the semantics of modeling languages. In the future, we aim to further explore the effect of long-tails in the *type* distribution on the nature of the shortcuts learned. Future ML4MDE work should incorporate pragmatic, requirements-driven context and evaluate systems on larger, more heterogeneous enterprise datasets to better understand—and ultimately overcome—the shortcut behaviors observed in this study.

Acknowledgments. This study was partially funded by the FFG-funded projects Smart GLSP – Facilitating Large Language Models for Smart GLSP-based Modeling (Grant number: FO999925707) and EAGLE – Enterprise Architecture Knowledge Graph for Learning and Exploration (Grant number: FO999925702).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. G. Guizzardi, T. Prince Sales, J. P. A. Almeida, G. Poels, Relational contexts and conceptual model clustering, in: IFIP Working Conference on The Practice of Enterprise Modeling, Springer, 2020, pp. 211–227.
2. P. P. F. Barcelos, T. P. Sales, M. Fumagalli, C. M. Fonseca, I. V. Sousa, E. Romanenko, J. Kritz, G. Guizzardi, A fair model catalog for ontology-driven conceptual modeling research, in: Int. Conf. on Conceptual Modeling, 2022, pp. 3–17.
3. M. M. Lankhorst, H. A. Proper, H. Jonkers, The anatomy of the archimate language, International Journal of Information System Modeling and Design (IJISMD) 1 (1) (2010) 1–32.
4. S. J. Ali, G. Guizzardi, D. Bork, Enabling representation learning in ontology-driven conceptual modeling using graph neural networks, in: International Conference on Advanced Information Systems Engineering, Springer, 2023, pp. 278–294.
5. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
6. X. Liu, D. McDuff, G. Kovacs, I. Galatzer-Levy, J. Sunshine, J. Zhan, M.-Z. Poh, S. Liao, P. Di Achille, S. Patel, Large language models are few-shot health learners, arXiv preprint arXiv:2305.15525 (2023).
7. M. B. Chaaben, L. Burgueño, I. David, H. Sahraoui, On the utility of domain modeling assistance with large language models, arXiv preprint arXiv:2410.12577 (2024).
8. K. Chen, Y. Yang, B. Chen, J. A. H. López, G. Mussbacher, D. Varró, Automated domain modeling with large language models: A comparative study, in: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2023, pp. 162–172.
9. J. A. H. López, R. Rubei, J. S. Cuadrado, D. Di Ruscio, Machine learning methods for model classification: a comparative study, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, 2022, pp. 165–175.
10. M. Weyssow, H. Sahraoui, E. Syriani, Recommending metamodel concepts during modeling activities with pre-trained language models, Software and Systems Modeling 21 (3) (2022) 1071–1089.
11. L. Burgueño, J. Cabot, S. Gérard, An lstm-based neural network architecture for model transformations, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2019, pp. 294–299.
12. M. Mitchell, D. C. Krakauer, The debate over understanding in ai's large language models, Proceedings of the National Academy of Sciences 120 (13) (2023) e2215907120.
13. A. C. Marcén, A. Iglesias, R. Lapeña, F. Pérez, C. Cetina, A systematic literature review of model-driven engineering using machine learning, IEEE Transactions on Software Engineering (2024).
14. D. Bork, S. J. Ali, B. Roelens, Conceptual modeling and artificial intelligence: A systematic mapping study, CoRR abs/2303.06758 (2023).
15. D. Strüber, M. Selter, G. Taentzer, Tool support for clustering large meta-models, in: Proceedings of the workshop on scalability in model driven engineering, 2013, pp. 1–4.

16. A. Elkamel, M. Gzara, H. Ben-Abdallah, An uml class recommender system for software design, in: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), IEEE, 2016, pp. 1–8.
17. F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, A. Pierantonio, Automated clustering of metamodel repositories, in: Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings 28, Springer, 2016, pp. 342–358.
18. Ö. Babur, L. Cleophas, M. van den Brand, Hierarchical clustering of metamodels for comparative analysis and visualization, in: European conference on modelling foundations and applications, Springer, 2016, pp. 3–18.
19. Ö. Babur, L. Cleophas, Using n-grams for the automated clustering of structural models, in: International Conference on Current Trends in Theory and Practice of Informatics, Springer, 2017, pp. 510–524.
20. R. Rubei, J. Di Rocco, D. Di Ruscio, P. T. Nguyen, A. Pierantonio, A lightweight approach for the automated classification and clustering of metamodels, in: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2021, pp. 477–482.
21. P. T. Nguyen, J. Di Rocco, L. Iovino, D. Di Ruscio, A. Pierantonio, Evaluation of a machine learning classifier for metamodels, Software and Systems Modeling 20 (6) (2021) 1797–1821.
22. A. Khalilipour, F. Bozyigit, C. Utku, M. Challenger, Categorization of the models based on structural information extraction and machine learning, in: International Conference on Intelligent and Fuzzy Systems, Springer, 2022, pp. 173–181.
23. J. A. H. López, C. Durá, J. S. Cuadrado, Experimenting with modeling-specific word embeddings, Software and Systems Modeling (2024) 1–23.
24. J. Di Rocco, D. Di Ruscio, C. Di Sipio, P. T. Nguyen, A. Pierantonio, Memorec: a recommender system for assisting modelers in specifying metamodels, Software and Systems Modeling 22 (1) (2022) 203–223.
25. C. Di Sipio, J. Di Rocco, D. Di Ruscio, P. T. Nguyen, Morgan: a modeling recommender system based on graph kernel, Software and Systems Modeling 22 (5) (2023) 1427–1449.
26. J. A. H. López, J. S. Cuadrado, R. Rubei, D. Di Ruscio, Modelxglue: a benchmarking framework for ml tools in mde, Software and Systems Modeling (2024) 1–24.
27. L. Burgueño, R. Clarisó, S. Gérard, S. Li, J. Cabot, An nlp-based architecture for the autocompletion of partial domain models, in: International Conference on Advanced Information Systems Engineering, Springer, 2021, pp. 91–106.
28. Djelic, S. J. Ali, C. Verbruggen, J. Neidhardt, D. Bork, A model cleansing pipeline for model-driven engineering: Mitigating the garbage in, garbage out problem for open model repositories, in: 2025 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS), 2025.
29. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint:1810.04805 (2018).
30. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
31. A. Kostina, M. D. Dikaiakos, D. Stefanidis, G. Pallis, Large language models for text classification: Case study and comprehensive review, arXiv preprint arXiv:2501.08457 (2025).

32. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, Advances in neural information processing systems 35 (2022) 22199–22213.
33. X. Tang, H. Zhai, C. Belwal, V. Thayanithi, P. Baumann, Y. K. Roy, Dynamic context-aware prompt recommendation for domain-specific ai applications, arXiv preprint arXiv:2506.20815 (2025).
34. A. Zhao, M. Ding, Z. Lu, T. Xiang, Y. Niu, J. Guan, J.-R. Wen, Domain-adaptive few-shot learning, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 1390–1399.
35. D. Pal, D. More, S. Bhargav, D. Tamboli, V. Aggarwal, B. Banerjee, Domain adaptive few-shot open-set learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 18831–18840.
36. T. P. Sales, P. P. F. Barcelos, C. M. Fonseca, I. V. Souza, E. Romanenko, C. H. Bernabé, L. O. B. da Silva Santos, M. Fumagalli, J. Kritz, J. P. A. Almeida, et al., A fair catalog of ontology-driven conceptual models, Data & Knowledge Engineering 147 (2023) 102210.
37. P.-L. Glaser, E. Sallinger, D. Bork, Ea modelset–a fair dataset for machine learning in enterprise modeling, in: IFIP Working Conference on The Practice of Enterprise Modeling, Springer, 2023, pp. 19–36.
38. D. C. Montgomery, Design and analysis of experiments, John wiley & sons, 2017.
39. M. Fumagalli, T. P. Sales, P. P. F. Barcelos, G. Micale, P.-L. Glaser, D. Bork, V. Zaytsev, D. Calvanese, G. Guizzardi, Mining frequent structures in conceptual models: M. fumagalli et al., Software and systems modeling (2025) 1–29.
40. M. Fumagalli, T. P. Sales, G. Guizzardi, Pattern discovery in conceptual models using frequent itemset mining, in: International Conference on Conceptual Modeling, Springer, 2022, pp. 52–62.
41. T. P. Sales, G. Guizzardi, Anti-patterns in ontology-driven conceptual modeling: the case of role modeling in ontouml, in: Ontology Engineering with Ontology Design Patterns, IOS press, 2016, pp. 161–187.
42. M. Skouradaki, V. Andrikopoulos, O. Kopp, F. Leymann, Rose: reoccurring structures detection in bpmn 2.0 process model collections, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2016, pp. 263–281.
43. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Second Edition, Springer, 2024.
44. C. Verbruggen, L. Netz, P.-L. Glaser, M. Scholz, C. Huemer, M. Calamo, B. Rumpe, M. Snoeck, D. Bork, Toward a community-curated golden dataset of uml models, in: MODELS 2025 Educator Symposium - Int. Conf. on Model Driven Engineering Languages and Systems, 2025.