Detecting Semantic Correspondences in Ecore Models via Contrastive Learning and LLM-Based Approach

Monalisha Ojha[1,*,†], Shilpi Gupta[1,†] and Rahul Sharma[2,†]

[1]University of Mannheim, Germany
[2]Karlsruhe Institute of Technology, Germany

Abstract

Detecting semantic correspondences between elements of independently developed Ecore models is a fundamental challenge in model-driven engineering, underpinning tasks such as model integration, consistency checking, and transformation reuse. Heterogeneity in naming, typing, and structure makes automated detection difficult, while manual inspection is impractical at scale. Because model elements encode semantics across multiple interdependent dimensions, approaches that rely solely on textual similarity are insufficient. We propose a hybrid approach that jointly models textual and structural semantics by combining contrastive learning with a large language model (LLM). Our pipeline employs Contrastive Learning (SimCLR)-based encoders to capture textual descriptions and the topology of inheritance and reference structures, integrated through a learned gating mechanism. The fused embeddings enable efficient candidate retrieval, and an LLM verifies the top-ranked candidates through pairwise semantic reasoning with contextual justification. We evaluate the approach on a cross-domain industrial scenario involving an Ecore-based brake system model and a CAD parameter model. Empirical results indicate that the hybrid method outperforms contrastive learning alone and LLM-only matching in precision, recall, and F1, highlighting the complementary strengths of representation learning and LLM-based reasoning.

Keywords

Semantic Correspondences, Large Language Models, Model-Driven Engineering, Contrastive Learning

CEUR-WS.org/Vol-4338/LLM4SE-paper4.pdf

1. Introduction

Model-driven engineering (MDE) relies on structured models—often expressed in the Eclipse Modeling Framework (EMF) Ecore format—to capture the design of software-intensive systems [1]. As organizations scale, independently developed models that describe overlapping aspects of a system inevitably diverge in naming, type representations, and structural organization. A brake system model, for example, may represent a property as an integer attribute `length`, while a CAD parameter model encodes the same quantity as a `NumericParameter` object with a separate `unit` field [2]. Identifying that the two elements denote the same concept is essential for model integration and transformation reuse, yet the heterogeneity makes automated detection challenging.

We refer to this challenge as the *Structural-Semantic Gap*: the disconnect between the surface-level syntactic representation of model elements (names, types, containment hierarchies) and their intended domain semantics. Two elements may share identical real-world meaning yet differ in naming (`ABSSensor` vs. `WheelSpeedSensor`), typing (an `EInt` attribute vs. a `NumericParameter` object), or structural role (inherited attribute vs. composed parameter). Bridging this gap requires approaches that jointly reason about textual and structural dimensions.

Classical model matching approaches [3, 4] rely on syntactic similarity measures such as string edit distance or token overlap, which break down under domain-specific synonyms, divergent capitalization, or structurally different encodings. More recent work leverages word embeddings [5] or pre-trained

Joint Proceedings of the STAF 2026 Workshops: AgileMDE, GCM, ICMM, LLM4SE, TTC. Rennes, France, June 29- July 3, 2026
*Corresponding author.
†These authors contributed equally.
monalisha.ojha@uni-mannheim.de (M. Ojha); shilpi.gupta@uni-mannheim.de (S. Gupta); rahul.sharma@kit.edu (R. Sharma)
0009-0007-6333-711X (M. Ojha); 0009-0003-3689-2223 (S. Gupta); 0000-0002-9024-8768 (R. Sharma)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-26

language models [6], but treating model elements as flat text discards the structural information encoded in inheritance and reference relationships. Large language models (LLMs) show strong semantic-reasoning capabilities [7], yet evaluating all $O(n^2)$ element pairs between two models is cost-prohibitive at scale—matching two models of 500 classes each would require 250,000 LLM calls.

In this paper, we propose a hybrid pipeline that addresses these limitations by combining the efficiency of contrastive representation learning with the reasoning capability of an LLM. Our approach consists of four stages:

1. A **text encoder** trained with SimCLR [8] on domain-specific augmentations to produce semantically meaningful embeddings of model element descriptions.
2. A **graph encoder** trained with SimCLR on Ecore inheritance and reference graphs to capture structural relationships.
3. A **gated fusion module** that learns to combine textual and structural representations via an element-wise gating mechanism.
4. An **LLM verification stage** that performs pairwise semantic reasoning on the top-ranked candidates retrieved by the fused embeddings.

We evaluate the approach on a large-scale benchmark of 3,589 Ecore metamodels (89,128 EClasses) drawn from the ModelSet repository [9], and demonstrate its application on a cross-domain industrial case study involving a brake system specification model and a CAD parameter model [2]. Experimental results show that the proposed hybrid pipeline achieves an F1 score of 0.975, outperforming both SimCLR-only (F1 = 0.946) and LLM-only (F1 = 0.969) baselines, while the contrastive retrieval stage reduces the number of required LLM calls from $O(n^2)$ to $O(nk)$, where k is a small constant.

The remainder of the paper covers related work (§2), the proposed methodology (§3), an industrial case study (§4), evaluation (§5), discussion (§6), threats to validity (§7), and concluding remarks (§8).

2. Background and Related Work

We review prior work in model matching, categorized as *syntactic*, *semantic*, and *hybrid*.

2.1. Syntactic Matching Approaches

Syntactic approaches rely on surface-level similarity between model elements using string metrics such as Levenshtein distance and Jaccard coefficient, type-based comparison of attributes and multiplicities, and structure-based analysis of containment and reference topologies [4]. Tools such as EMFCompare [10] combine these heuristics with configurable strategies. While computationally efficient, syntactic methods struggle to capture semantic equivalence when naming conventions or structural representations diverge significantly across models.

2.2. Semantic Matching Approaches

Semantic approaches extend syntactic techniques by incorporating domain knowledge. Ontology-based methods map elements to shared concepts [11], while embedding-based methods learn vector representations using word vectors [5] or sentence transformers [12], with recent work leveraging pre-trained language models for software artifacts [6]. However, these approaches typically treat model elements as isolated textual units, overlooking structural dependencies such as inheritance hierarchies central to Ecore models.

Contrastive learning has proven effective for learning invariant embeddings. SimCLR [8] pulls augmented views of an instance together while pushing different instances apart via NT-Xent loss, and has been extended to text [13], graphs [14], and code [15]. We adopt SimCLR over alternatives such as MoCo for its conceptual simplicity and natural compatibility with semantics-preserving Ecore transformations; its in-batch negatives avoid the momentum encoders and memory queues required by MoCo. Separately, LLMs have shown strong capabilities in software engineering [16, 17], MDE tasks [18, 19], and entity matching [20].

2.3. Hybrid Matching Approaches

Hybrid approaches combine efficient similarity retrieval with higher-level reasoning. However, applying LLMs directly to model matching requires $O(n^2)$ pairwise calls, which is cost-prohibitive at scale. Our approach follows a retrieve-then-verify paradigm: contrastive embeddings enable efficient candidate retrieval, reducing comparisons to $O(nk)$ where k is a small constant, and LLM verification is applied only to the top-ranked candidates.

3. Methodology

Our methodology detects semantic correspondences through a three-phase hybrid pipeline (Figure 1): Phase 1 transforms raw Ecore models into structured representations and derives training supervision; Phase 2 trains dual contrastive encoders unified by a learned fusion module; Phase 3 deploys the trained models in a real-time inference service that pairs structural retrieval with LLM-driven semantic verification.

3.1. Data Preparation and Ground Truth Construction

The pipeline ingests the ModelSet v0.9.2 corpus [9] (5,443 raw Ecore metamodels). After filtering models with fewer than 2 or more than 500 EClasses and removing duplicates, we retain 3,589 unique metamodels spanning diverse SE domains (web frameworks, database schemas, DSLs, UML profiles, automotive, enterprise integration). A data parser extracts all 89,128 EClass definitions—names, supertypes, attributes (with types and multiplicities), and references (with target types, multiplicities, and containment flags)—into a structured JSON representation. From this corpus we construct a three-tier ground truth of 25,000 pairs following established synthetic practices [3]. Each EClass is also serialized into a canonical text string that encodes its full structural signature; this serialization is both the input to the text encoder and the descriptive context supplied to the LLM at inference.

Training pairs are generated automatically through category-consistent synthetic transformations applied to the serialized EClass descriptions. Six semantics-preserving augmentation types are employed:

1. **Synonym replacement** – class and attribute names are substituted with domain-specific synonyms drawn from a curated dictionary of 30 modeling-concept families (e.g., `Person` → `Individual`, `Customer` → `Client`).
2. **Attribute reordering** – the order of attributes and references within an EClass is randomly permuted to simulate varying developer conventions.
3. **Type abstraction** – Ecore types are mapped to equivalent representations (e.g., `EString` → `String`, `EInt` → `Integer`).
4. **Case switching** – identifiers are converted between `camelCase` and `snake_case`.
5. **Optional feature addition** – plausible optional attributes are appended to the EClass.
6. **Inheritance flattening** – supertype attributes are inlined into the subclass, removing the inheritance reference.

These augmentations eliminate the need for manual annotation. Hard negatives (structurally proximate but semantically distinct classes from the same model) and easy negatives (classes from unrelated models) are also sampled to provide balanced contrastive supervision.

3.2. Offline Training

Contrastive learning proceeds along two complementary modalities, each trained under the SimCLR contrastive framework [8], before being unified by a gated fusion module.

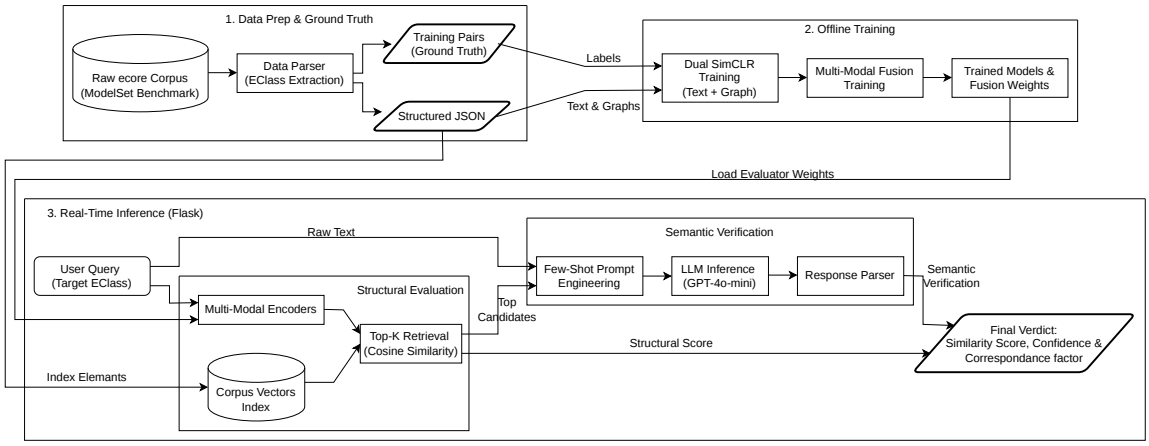


Figure 1: Overview of the hybrid semantic correspondence detection pipeline.

3.2.1. Text SimCLR

The text encoder produces semantically meaningful embeddings of serialized model element descriptions. For each training sample, two stochastically augmented views are generated using the text-level transformations described in Section 3.1 (synonym replacement, attribute reordering, type abstraction, case switching, and optional feature addition). Both views are encoded by a frozen sentence transformer (all-MiniLM-L6-v2 [12]) and passed through a trainable 3-layer projection head ($384 \to 512 \to 512 \to 256$) with batch normalization, ReLU activations, and dropout. The projection head is optimized with the NT-Xent contrastive loss [8] at temperature $\tau = 0.1$.

3.2.2. Graph SimCLR

Inheritance hierarchies and cross-references in Ecore metamodels encode semantic information that text alone cannot capture. For each Ecore package, a graph is constructed with EClasses as nodes and bidirectional edges for inheritance and reference relations; node features are initialized with sentence embeddings of the serialized descriptions. The encoder is a 3-layer GCN (hidden dim. 128, batch normalization, dropout) with a dual-pooling readout (mean + max) yielding a 256-dim graph representation, followed by a 2-layer projection head. Augmentations are edge dropping and node feature masking; training uses NT-Xent loss at $\tau = 0.1$.

3.2.3. Multi-Modal Fusion

The text and graph encoders capture complementary facets of model semantics. A learned gate adaptively calibrates each modality per element: it takes the concatenation of the text embedding $\mathbf{h}_t$ and graph embedding $\mathbf{h}_g$ and produces $\alpha \in [0, 1]^{256}$ via a two-layer network with sigmoid activation. The fused representation is

$$\mathbf{h}_f = \alpha \odot f_t(\mathbf{h}_t) + (1 - \alpha) \odot f_g(\mathbf{h}_g) \tag{1}$$

where f_t, f_g are learned projections and $\odot$ is element-wise multiplication. This lets the model rely more on textual cues when naming is informative and more on structural cues when it is ambiguous. The fusion module is trained end-to-end with NT-Xent loss while the pretrained encoders remain frozen.

3.3. Real-Time Inference

The trained models are deployed within a Flask-based web application that accepts user queries and returns ranked semantic correspondences.

3.3.1. Structural Evaluation

When a user submits a target EClass as a query, the multi-modal encoders produce its fused embedding. This query embedding is compared against a pre-computed corpus vectors index containing the fused embeddings of all EClasses in the repository. Top-k candidates are retrieved via cosine similarity.

3.3.2. Semantic Verification

The top-k candidates and the raw query text are forwarded to a semantic verification module powered by GPT-4o-mini. A few-shot prompt includes three exemplar pairs covering synonym recognition (e.g., "Person" vs. "Individual"), structural role matching (inheritance vs. composition), and domain-boundary distinction; each exemplar gives an expected score and a one-sentence justification. The LLM returns a similarity score in $[0, 1]$ with a natural-language justification in structured JSON, which a response parser consumes downstream.

3.3.3. Final Output

The pipeline produces a consolidated verdict for each accepted correspondence, comprising three components:

- **Similarity score**: a combined measure computed as $s_{\text{comb}} = \lambda \cdot s_{\text{struct}} + (1 - \lambda) \cdot s_{\text{llm}}$, where s_{struct} is the cosine similarity from contrastive retrieval, s_{llm} is the LLM's semantic assessment score, and $\lambda = 0.4$ is tuned on the validation set to maximize F1.
- **Heuristic confidence estimate**: an LLM-reported confidence indicator that accompanies the score; we treat it as a heuristic signal rather than a calibrated probability.
- **Semantic correspondences and divergence factors**: the set of model elements deemed semantically equivalent, together with the divergence factors that distinguish near-misses from genuine correspondences.

By restricting LLM invocations to only the highest-confidence candidates from the structural stage, the pipeline achieves substantial cost and latency savings while preserving matching quality and providing explainable results.[1]

4. Case Study

We apply the hybrid pipeline to two independently developed Ecore metamodels for an automotive disc brake assembly, adapted from Hagel et al. [2]. Figure 2 shows both: the **brake system specification metamodel** (left) uses an inheritance-based design, with an abstract `BrakeComponent` (`id`, `OEM_number`) extended by five concrete subclasses (`ABSSensor`, `BrakeCaliper`, `BrakeHose`, `BrakeDisk`, `BrakePad`), each carrying component-specific attributes with units embedded in names (e.g., `diameterInMM` of type `EInt`). The **CAD parameter metamodel** (right) is a flat composition-based structure: a `CAD_Model` contains `Namespace` objects grouping typed `Parameter` instances (`NumericParameter`, `StringParameter`, `BooleanParameter`) with units captured via an explicit `Unit` enumeration.

The two models exhibit heterogeneity along four dimensions: *naming divergence* (embedded units vs. separate unit fields), *structural divergence* (inheritance vs. composition), *type representation divergence* (`EInt` attributes vs. `EFloat` parameter objects), and *granularity divergence* (domain-specific EAttributes vs. generic parameter EAttributes). We evaluate correspondences at **two levels of the MOF hierarchy**:

- **Metamodel level (M2):** Matching `EAttribute` definitions from `brakesystem.ecore` against `cad.ecore`—comparing structural type definitions (names, types, EClass context) without concrete values.
- **Instance level (M1):** Matching concrete attribute values from `brakesystem.model` against parameter instances from `example.cad`—comparing runtime data with value-based confirmation.

[1]The replication package (source code, data, trained models, and evaluation results) is publicly available at https://doi.org/10.5281/zenodo.20157003.

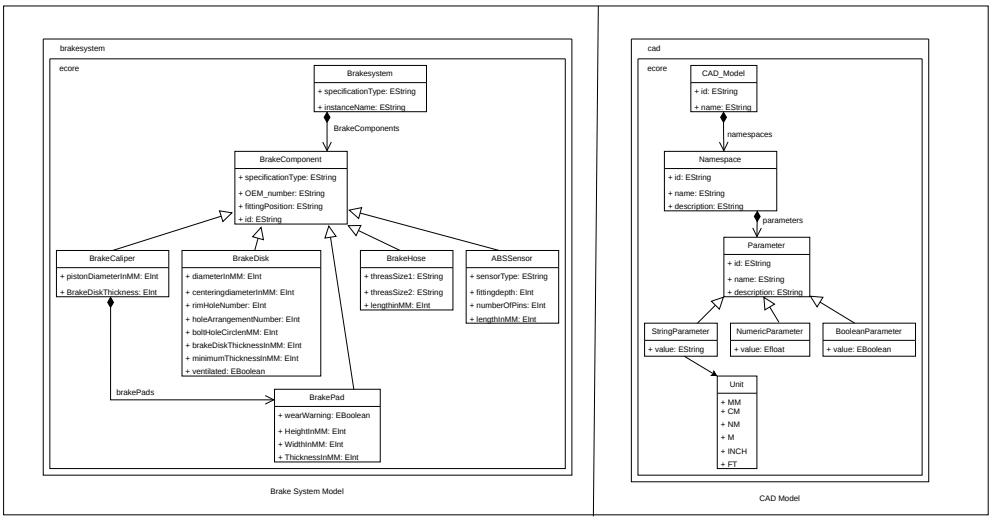


Figure 2: The BrakeSystem specification metamodel (left) and the CAD parameter metamodel (right).

Table 1
Metamodel-level (M2) attribute-to-attribute semantic correspondences. The "Reasoning" column contains justifications generated by GPT-4o-mini during the semantic verification stage.

#	BrakeSystem (M2)	CAD (M2)	Sim	LLM	Comb.	Reasoning
M2 Attribute-to-Attribute Correspondences						
1	*BrakeComponent.id* (*EString*)	*Namespace.id* (*EString*)	0.72	1.00	0.89	Identical name, type, and role as identifier.
2	*BrakeComponent.specType* (*EString*)	*StringParameter.value*, name = "Spec Type"	0.61	0.90	0.78	Same concept; typed attribute vs. named parameter.
3	*BrakeComponent.OEM_number* (*EString*)	*StringParameter.value*, name = "OEM Number"	0.68	0.95	0.84	Same identifier; naming differs (underscore vs. label).
4	*BrakeDisk.diameterInMM* (*EInt*)	*NumericParameter.value*, name = "Diameter", unit = *MM*	0.58	0.90	0.77	Unit embedded in the source attribute name vs. explicit unit enumeration in the target model.
5	*BrakeDisk.ventilated* (*EBoolean*)	*BooleanParameter.value*, name = "Ventilated"	0.52	0.95	0.78	Same boolean flag and semantics.
Representative M2 Attribute-Level Divergence						
—	*BrakeDisk.diameterInMM* (*EInt*)	*NumericParameter.value*, name = "Disk Thickness"	0.55	0.10	0.28	Both are numeric attributes, but they denote different physical quantities.

M2 Results (Metamodel Level). Class-to-class matching yields no correspondences above the acceptance threshold, as the two metamodels follow incompatible design paradigms. However, meaningful correspondences emerge at the *attribute-to-attribute* granularity. Table 1 presents the 12 correspondences and 2 representative divergences detected.

M1 Results (Instance Level). At the instance level, the BrakeDisk instance is matched against 11 CAD parameter instances. Unlike M2, instance matching benefits from concrete value equality as

Table 2
Instance-level (M1) semantic correspondences. The "Reasoning" column contains justifications generated by GPT-4o-mini during the semantic verification stage.

#	BrakeSystem (M1)	CAD (M1)	Sim	LLM	Comb.	Reasoning
Instance-Level Correspondences						
1	*BrakeDisk.OEM_number = "VW123456"*	*OEM Number (String) = "VW123456"*	0.69	1.00	0.87	Identical string values confirm the match.
2	*BrakeDisk.thickness = 30*	*Disk Thickness (Numeric) = 30.0*	0.60	1.00	0.84	Same thickness; integer vs. float representation (*30 = 30.0*).
3	*BrakeDisk.ventilated = true*	*Ventilated (Boolean) = true*	0.43	1.00	0.77	Same boolean flag with matching runtime value.
Representative Instance-Level Divergence						
—	*BrakeDisk.centDiam = 20*	*Hole Number = 20.0*	0.45	0.50	0.48	Same numeric value, but the properties denote different concepts.

Table 3
Two-level case study summary.

Metric	M2 (Metamodel)	M1 (Instance)
Source elements	26 EAttributes	10 attribute-value pairs
Target elements	12 EAttributes	11 parameters
Total pairs evaluated	312	110
Correspondences (score ≥ 0.5)	12	9
Mean combined score (matches)	0.776	0.805

additional evidence. The pipeline detects 9 correspondences, all confirmed by the LLM with score 1.0. Table 2 presents these results.

Cross-Level Observations. Comparing the M2 and M1 results (Table 3) yields four observations. First, attribute-to-attribute granularity is essential: class-level matching finds zero correspondences, whereas attribute-level matching reveals valid correspondences. Second, the LLM adapts its reasoning to the available evidence—using type signatures at M2 and exploiting value equality at M1. Third, the SimCLR encoder shows a moderate transfer gap at M1, confirming that LLM verification is essential. Fourth, divergence detection is robust: near-misses are correctly rejected at both levels despite shared numeric types or values.

5. Evaluation

We evaluate the hybrid pipeline against two baselines on three dimensions: classification effectiveness, score quality, and computational cost.

5.1. Dataset and Setup

The benchmark draws 89,128 EClasses from 3,589 Ecore metamodels in ModelSet v0.9.2 [9]. We generate 25,000 pairs following synthetic practices [3]: 10,000 positive pairs (label 1.0) via the six semantics-preserving transformations defined in Section 3.1 (synonym replacement, attribute reordering, type abstraction, case switching, optional feature addition, and inheritance flattening), 10,000 hard negatives

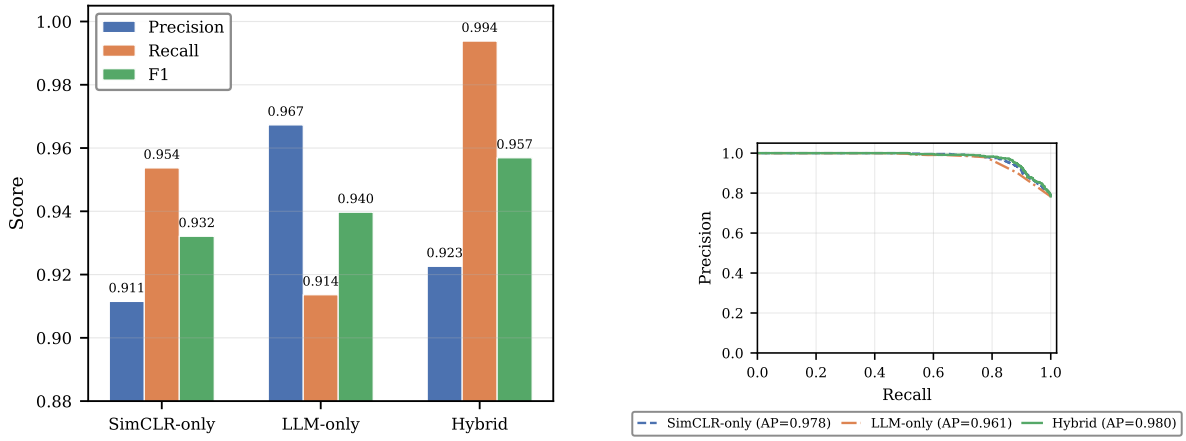


Figure 3: Left: Precision, Recall, and F1 comparison across the three methods. Right: Precision-recall curves with average precision (AP) values.

(label 0.5) from same-model EClasses, and 5,000 easy negatives (label 0.0) from cross-model EClasses. The split is 70/10/20% (train/val/test). We compare **SimCLR-only** (fused contrastive cosine similarity), **LLM-only** (GPT-4o-mini pairwise scoring), and **Hybrid** (contrastive retrieval + LLM verification with score blending). Thresholds are tuned on validation to maximize F1. SimCLR-only is evaluated on all 5,000 test pairs; LLM-only and Hybrid are evaluated on a stratified 500-pair subset (200 positive, 200 hard negative, 100 easy negative) due to API cost constraints. This subset preserves the original label distribution, and the evaluation-size difference is acknowledged as a limitation in Section 7.

5.2. Results

Table 4 reports test-set performance. SimCLR-only achieves F1 = 0.946, confirming that contrastive encoders capture substantial semantic information. LLM-only reaches the highest precision (0.990, F1 = 0.969) but at lower recall. The Hybrid attains the best F1 (0.975) with near-perfect recall (0.990) and competitive precision (0.961)—a practically significant advantage since missing a true correspondence is typically costlier than surfacing a false positive. Figure 3 visualizes the metric comparison and precision-recall curves.

Table 4
Classification performance on the test set

Method	Prec.	Rec.	F1	Acc.	Spear.
SimCLR-only	0.944	0.949	0.946	0.957	0.868
LLM-only	**0.990**	0.950	0.969	0.976	0.873
Fusion (Hybrid)	0.961	**0.990**	**0.975**	**0.980**	**0.886**

Spearman's ρ confirms Hybrid achieves the best ordinal ranking ($\rho = 0.886$), followed by LLM-only (0.873) and SimCLR-only (0.868), indicating that both contrastive fine-tuning and LLM verification contribute to score quality.

5.3. Computational Cost

Table 5 reports inference latency. SimCLR-only processes all 5,000 test pairs in 44.4 s (~9 ms/pair) via batch matrix multiplication. LLM-only requires ~14.5 min for 500 pairs, dominated by sequential API round trips. The Hybrid approach reduces this to ~3.3 min for the same 500 pairs by leveraging contrastive retrieval with top-$k = 5$ filtering: only 110 of the 500 candidate pairs are forwarded to the

LLM, yielding a 4.4× speedup over LLM-only while achieving the highest F1. More generally, the hybrid pipeline reduces LLM calls from n^2 to kn, making it viable for industrial-scale models.

Table 5
Inference latency comparison.

Method	Pairs	Total (s)	Per pair (s)
SimCLR-only	5,000	44.4	0.009
LLM-only	500	869.3	1.739
Fusion (Hybrid)	500	196.7	0.393

6. Discussion

Complementarity. The contrastive stage achieves strong matching (F1 = 0.946) through distributional learning, with the gated fusion adaptively weighting textual and structural signals. The LLM attains the highest precision (0.990) via contextual reasoning. Combining both in the Hybrid yields the best F1 (0.975) with near-perfect recall (0.990), suggesting complementary behavior between retrieval and semantic verification: contrastive retrieval maximizes coverage while the LLM applies targeted semantic scrutiny. While the numerical improvements are modest, the hybrid pipeline consistently improves both ranking quality and recall while substantially reducing LLM calls. The LLM's natural language justifications additionally address explainability—a critical concern for AI-assisted MDE tools.

Scalability. Training on the full corpus of 3,589 models (89,128 classes) yields a contrastive F1 of 0.946, indicating that the contrastive approach benefits from large and diverse training data. The contrastive stage processes 5,000 pairs in ∼50 s on CPU, enabling interactive use. In deployment, contrastive retrieval reduces LLM calls from $O(n^2)$ to $O(nk)$—a 20-fold reduction for $n = 100$, $k = 5$. When LLM access is unavailable, the contrastive stage alone (F1 = 0.946) provides graceful degradation without architectural changes.

Implications for MDE. The case study confirms that in heterogeneous environments semantic correspondences reside at the *attribute-to-attribute* level rather than the class level targeted by traditional tools [3, 4], and our pipeline addresses both MOF levels without modification.

Broader Applicability to Software Engineering. Although our evaluation targets Ecore metamodels, the retrieve-then-verify architecture generalizes to other SE contexts where heterogeneous artifacts must be aligned—e.g., schema matching, API compatibility checking, and ontology alignment—by substituting domain-appropriate augmentations and serialization strategies.

Comparison with Existing Tools. Our experimental comparison focuses on ablation of the pipeline components rather than on established tools such as EMFCompare [10], which operates on syntactic similarity at the class level, whereas our approach targets semantic correspondences at the attribute level across structurally heterogeneous models. A shared attribute-level benchmark with manually curated ground truth would enable a direct comparison and is left as future work.

7. Threats to Validity

Construct validity. The combined score ($s_{\text{comb}} = 0.4\, s_{\text{struct}} + 0.6\, s_{\text{llm}}$) uses a fixed weighting tuned on validation, which may not generalize to all heterogeneity patterns. The LLM's self-reported confidence is used without external calibration, and its mapping to true correspondence strength has not been independently validated.

Internal validity. The synthetic ground truth may bias results toward trained transformation types; we mitigate this with six diverse augmentation categories, stratified held-out splits, and the cross-domain case study (Section 4) on real-world metamodels not seen during training. SimCLR-only is evaluated on 5,000 test pairs while LLM-only and Hybrid use a stratified 500-pair subset due to API cost, limiting direct comparability; the subset preserves the original label distribution.

External validity. The evaluation covers 3,589 Ecore metamodels from ModelSet v0.9.2 and may not generalize to other modeling languages (UML, SysML) or proprietary models. Near-perfect accuracy across methods (> 0.95) suggests the synthetic test set may be easier than real-world matching; a manually curated benchmark and a human-in-the-loop study are needed for stronger evidence of real-world effectiveness.

8. Conclusion

We presented a hybrid approach combining SimCLR-based contrastive encoders with LLM verification for detecting semantic correspondences between Ecore model elements. Trained on 3,589 metamodels (89,128 EClasses), the Hybrid achieves F1 = 0.975, outperforming SimCLR-only (0.946) and LLM-only (0.969), while reducing LLM calls from $O(n^2)$ to $O(nk)$. The industrial case study demonstrates dual-level operation across M2 and M1. Future work will conduct human-in-the-loop evaluations, compare against established tools such as EMFCompare on a curated benchmark, and explore open-source LLMs for cost and reproducibility.

9. Acknowledgments

This work was funded by the Deutsche Forschungsgemein- schaft (DFG, German Research Foundation) – CRC 1608 – 501798263.

10. Declaration on Generative AI

During the preparation of this work, the authors used GPT-4o-mini in order to: (1) serve as the LLM verification component within the proposed hybrid pipeline (part of the research methodology), and (2) assist with grammar and spelling checking during manuscript preparation. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks, EMF: Eclipse Modeling Framework, 2 ed., Addison-Wesley, 2008.

[2] N. Hagel, J. Mäkelburg, C. Hammann, T. Weber, T. A. Völk, F. P. Urbano, et al., Explainability in automated cross-domain model-driven brake system development, in: Proc. 40th IEEE/ACM Int. Conf. on Automated Software Engineering Workshops (ASEW), 2025, pp. 204–211. doi:10.1109/ASEW67777.2025.00047.

[3] D. S. Kolovos, R. F. Paige, F. A. C. Polack, Model comparison: A foundation for model composition and model transformation testing, in: Proceedings of the International Workshop on Global Integrated Model Management (GaMM), 2006, pp. 13–20.

[4] J.-R. Falleri, M. Huchard, M. Lafourcade, C. Nebut, Metamodel matching for automatic model transformation generation, in: Proceedings of the International Conference on Model Driven Engineering Languages and Systems (MoDELS), Springer, 2008, pp. 326–340.

[5] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Proceedings of the ICLR Workshop, 2013.

[6] Z. Chen, D. Kolovos, R. Paige, Using pre-trained language models for model matching, in: Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2024.

[7] T. B. Brown, et al., Language models are few-shot learners, in: Proceedings of NeurIPS, 2020, pp. 1877–1901.

[8] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR, 2020, pp. 1597–1607.
[9] J. A. H. López, J. L. C. Izquierdo, J. S. Cuadrado, Modelset: A dataset for machine learning in model-driven engineering, in: Proc. 25th Int. Conf. on MODELS, ACM, 2022, pp. 98–108. doi:10.1145/3550355.3552452.
[10] Eclipse Foundation, Emf compare, n.d. URL: https://www.eclipse.org/emf/compare/.
[11] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007.
[12] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference (EMNLP-IJCNLP), Association for Computational Linguistics, 2019, pp. 3982–3992.
[13] T. Gao, X. Yao, D. Chen, Simcse: Simple contrastive learning of sentence embeddings, in: Proceedings of EMNLP, 2021, pp. 6894–6910.
[14] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, Y. Shen, Graph contrastive learning with augmentations, in: Proceedings of NeurIPS, 2020, pp. 5812–5823.
[15] P. Jain, A. Jain, T. Zhang, P. Abbeel, J. E. Gonzalez, I. Stoica, Contrastive code representation learning, in: Proceedings of EMNLP, 2021, pp. 5954–5971.
[16] M. Chen, et al., Evaluating large language models trained on code, arXiv:2107.03374v1, 2021.
[17] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, R. Karri, Examining zero-shot vulnerability repair with large language models, in: Proceedings of IEEE Symposium on Security and Privacy, 2023, pp. 2339–2356.
[18] M. Chaaben, L. Burgueño, J. Cabot, On the assessment of generative ai in modeling tasks, in: Proceedings of ACM/IEEE MODELS Companion, 2023.
[19] H.-G. Fill, F. Fettke, K. Hinkelmann, Conceptual modeling and large language models: Impressions from first experiments, Enterprise Modelling and Information Systems Architectures 18 (2023).
[20] R. Peeters, Entity Matching using Deep Neural Networks: From Discriminative Pre-trained Language Models to Generative Large Language Models, Ph.D. thesis, University of Mannheim, Mannheim, Germany, 2025. URL: https://madoc.bib.uni-mannheim.de/69425/.