Evaluating Embedding Models and Preprocessing for Retrieval of MDE Elements in RAG Systems

Julian Roßkothen
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Vector Informatik GmbH
Karlsruhe, Germany
rosskothen@kit.edu

David Inca Pilco
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
david.pilco@student.kit.edu

Tobias Hey
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
hey@kit.edu

Clemens Reichmann
Vector Informatik GmbH
Karlsruhe, Germany
clemens.reichmann@vector.com

Ralf Reussner
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
reussner@kit.edu

Abstract

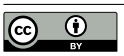
Model-driven engineering tools manage large, interconnected models that may contain millions of model elements, making it increasingly difficult for engineers to locate relevant information using classical search mechanisms. Retrieval-Augmented Generation is a promising approach for enabling natural language question answering for large knowledge bases, but the applicability of text-based embedding models to highly structured, graph-shaped model artifacts has not been systematically evaluated.

This paper presents an empirical study comparing 13 embedding models, 7 serialization formats, and 7 data preprocessing strategies for the retrieval of model elements. In an evaluation on two PREEvision Electric/Electronic-architecture models using automatically derived questions, we find that text embedding models are capable of capturing the semantics of model elements. However, they often only find anchor points in the model and struggle to retrieve all relevant model elements.

The choice of embedding model is the dominant factor, followed by data preprocessing. Serialization has a smaller but model-dependent effect; compact, semi-structured formats such as Markdown key-value pairs perform most robustly. Embedding model retrieval-benchmark rankings transfer well to structured model element retrieval.

CCS Concepts

• **Computing methodologies** → *Information extraction*; • **Software and its engineering** → **Model-driven software engineering**; • **Information systems** → *Data encoding and canonicalization*; Top-k retrieval in databases; *Question answering*.



MODELS Companion 2026, Málaga, Spain

ACM ISBN 979-8-4007-2903-4/26/10
https://doi.org/10.1145/3837062.3839371

Keywords

Retrieval-augmented Generation, Embedding Models, Model-driven Engineering

ACM Reference Format:
Julian Roßkothen, David Inca Pilco, Tobias Hey, Clemens Reichmann, and Ralf Reussner. 2026. Evaluating Embedding Models and Preprocessing for Retrieval of MDE Elements in RAG Systems. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3837062.3839371

1 Introduction

Model-driven engineering (MDE) tools manage large, interconnected models that may contain millions of artifacts. As MDE models grow, it becomes increasingly difficult for engineers to locate relevant elements, understand cross-component dependencies, or navigate structures using classical search mechanisms.

Conversational AI systems based on Large Language Models (LLMs) offer a natural interface for querying such models. However, directly embedding entire model repositories in an LLM's context window is infeasible due to context-length limits and high inference costs. Training or fine-tuning LLMs with model-specific data is infeasible due to update frequencies and training costs.

Retrieval-Augmented Generation (RAG) addresses these limitations by combining a retrieval component with an LLM that generates answers from the retrieved context.

While embedding models and RAG systems have been extensively evaluated on natural language text, their suitability for structured, graph-shaped model-driven engineering (MDE) models remains largely unexplored. Key open questions concern: which embedding models transfer best from text to model artifacts; how artifacts should be serialized from graph to string; and how preprocessing can compensate for the structural mismatch between graph-based representations and the sequential token inputs expected by transformer models.

In this paper, we test different embedding models and preprocessing strategies for their suitability for use in RAG systems for PREEvision Electric/Electronic-architecture (EEA) [22] models. Specifically, we address the following research questions:

- **RQ1:** Can embedding models be used to identify relevant information in models?
- **RQ2:** Which embedding models are most effective for retrieving structured model artifacts?
- **RQ3:** How do different serialization formats affect retrieval quality?
- **RQ4:** Which data preprocessing strategies most improve retrieval over complex model hierarchies?

2 Related Work

PREEvision. PREEvision is a model-based development environment by Vector Informatik GmbH for complex software-defined systems, with a focus on electrical/electronic development [1, 22]. It follows a model-driven approach in which system aspects—functional, logical, and physical—are described graphically at a high abstraction level, independent of implementation details [22]. PREEvision supports the full development process from requirements definition through software behavior and hardware configuration, with trace links connecting all layers so that relationships between requirements, architecture, software, and hardware remain explicitly traceable [22]. The tool's domain-specific metamodel is the EEA metamodel [22], which is based on the Meta-Object Facility (MOF) standard. Each model element (also called an artifact) is identified by a unique ID (also called XMIID) and carries attribute values as well as explicit links to related artifacts.

Retrieval-Augmented Generation. RAG systems extend large language models by incorporating external knowledge sources into the generation process [16]. Unlike standard LLMs that rely solely on pretrained parameters, RAG combines language models with retrieval techniques to fetch relevant information from a database and use it as context for answer generation [16]. The approach consists of two phases: (1) *indexing*, where data is preprocessed, embedded into high-dimensional vectors, and stored in a vector database with references to the original documents; and (2) *retrieval*, where a user query is embedded and compared against stored vectors via similarity search to identify the most relevant information [16]. The retrieved context is then passed to the LLM along with the original query, enabling the system to generate accurate, up-to-date responses without requiring LLM retraining, while maintaining traceability to the underlying data sources [16]. During indexing, the data is split into semantic snippets, called chunks. Each chunk is transformed into a high-dimensional vector using an embedding model.

Text retrieval benchmarks. Several benchmarks have been developed to systematically evaluate retrieval and embedding models. BEIR compares information retrieval models across diverse text retrieval tasks and has significantly improved the comparability of retrieval approaches [23]. The Massive Text Embedding Benchmark (MTEB) [19] and its extension MMTEB [6] provide the most comprehensive framework for comparing text embedding models, with MMTEB extending MTEB to multilingual scenarios and more challenging novel tasks [6]. These benchmarks systematically evaluate hundreds of embedding models across tasks such as retrieval, classification, clustering, and re-ranking [6, 19]. However, all these benchmarks focus on unstructured text data rather than MDE models. This work builds on these benchmarks but explicitly investigates the extent to which their findings transfer to structured MDE model elements, additionally examining the influence of different serializations and data preprocessing strategies.

Retrieval over structured and code data. Embedding-based retrieval has been studied for various structured data formats beyond natural language text [9]. In software engineering, natural language queries are matched against code snippets by representing text and code in a shared vector space [26], with specialized benchmarks such as CoIR [17] for evaluation. However, these approaches are limited in transferability since code snippets preserve meaning, whereas model elements are graph-based with structural relationships encoded via explicit references rather than implicit token order.

Similarly, table retrieval poses challenges for embedding models due to the need to capture explicit two-dimensional structural relationships [11]. Dense table retrievers [11] and serialization strategies with metadata [8] have been proposed to address this.

For knowledge graphs, specialized embedding models focus on link prediction [7] or entity matching with different serializations [27], but do not address general retrieval contexts for model artifacts. Recent graph-aware methods like GraphRAG [5] and HubLink [13] explicitly model relationships during retrieval, but require structurally modified architectures. In contrast, this work evaluates standard text embedding models with different serializations and preprocessing strategies for retrieval over MDE model elements.

Preprocessing effects on retrieval. The impact of data preprocessing on retrieval system performance has been studied across domains and is considered an important factor for similarity search quality. Early work showed that preprocessing steps can affect ranking behavior independently of the embedding model, such as lossy compression in image-based retrieval [21].

For text-based retrieval, studies report that basic preprocessing (such as lowercasing, punctuation and stopword removal, stemming/lemmatization) yields measurable but task- and model-dependent effects, with simple steps often providing robust improvements while complex procedures show inconsistent gains [2, 3, 12, 15]. Our work evaluates preprocessing strategies adapted to MDE model elements, such as filtering unused attributes and enriching with metadata, to assess how different strategies affect the retrieval performance of existing text embedding models.

Specialized embeddings for structured data. As an alternative to universal text models, approaches like SANTA [18] and WordE4MDE [10] develop specialized embeddings that directly account for structural properties of specific data formats. In contrast, this work does not pursue a model-centric approach but instead evaluates existing, generally available text embedding models in combination with different preprocessing and serialization strategies.

3 Approach

Our Approach is divided into two main components: the artifact embedding component and the evaluation component. The artifact embedding component creates test-based chunks from MDE models

and is designed for use in production systems (see subsection 3.1). We describe the evaluated preprocessing, serializations, and embedding models in subsection 3.2, subsection 3.3, and subsection 3.4, respectively.

3.1 Ingestion Pipeline Architecture

The MDE model element embedding component is the evaluated component of this work, responsible for transforming model data from a PREEvision model into vector representations. The component's modules are organized as sequential processing steps in a pipeline. This modular architecture enables independent execution of individual processing steps, as well as targeted replacement of modules. The pipeline operates in parallel, with logically decoupled modules exchanging data via queues. The resulting workflow is illustrated in Figure 1. (1) All model elements are retrieved from a PREEvision model; (2) The preprocessing module performs chunk-specific content aggregation from raw model elements; (3) The serialization module transforms preprocessed model elements into string representations serving as input to the embedding model; (4) The embedding module passes these representations to the respective embedding model, which generates multi-dimensional vectors for each MDE model element; (5) The vector database module stores the resulting vectors. For each model element, we create a single chunk. Each model element's ID serves as the identifier in the vector database for storage and later retrieval result association.

While modern vector databases typically employ Approximate Nearest Neighbor (ANN) algorithms for efficient similarity search at scale [4, 25], this work deliberately disables approximate search in favor of exact neighborhood search. This ensures deterministic and reproducible retrieval results, excluding the possibility that differences in retrieval performance stem from search algorithm approximations rather than the evaluated preprocessing, serialization, or embedding strategies. For every query, a complete comparison against all stored vectors is performed.

3.2 Data Preprocessings

The data preprocessing module prepares the concrete data used when embedding a single model element. The evaluated preprocessing strategies are summarized in Table 1. An exemplary example of the impact of each data processing step is shown in Figure 2. The evaluation focuses on two central strategies: *plain* (DV1) as a baseline with no changes compared to the raw data from the PREEvision model, and *mopilot* (DV2) combining steps DV3–DV7. *filterAttributes* (DV3) removes semantically irrelevant attributes like redundant IDs and temporal metadata. *addLayer* (DV4) appends PREEvision layer information describing each model element's abstraction level. *convertMofType* (DV5) normalizes verbose MOF type names to concise forms. *processLinks* (DV6) dereferences model element links by inlining the referenced element's name, type, and ID. *extendFileContent* (DV7) incorporates the contents of externally linked files into the chunk. The file contents are mostly descriptive texts about a specific model element.

3.3 Serializations

The serialization module transforms preprocessed model elements into linear string representations. The selected formats (Table 2) cover a broad spectrum of established representations for structured data. We include widely adopted, formally structured formats such as JSON (SF1), XML (SF2), and YAML (SF3), as well as more minimalist or specialized approaches like TOML (SF4) and TOON (SF5). The latter formats are evaluated for their potential to better preserve structural neighborhood through minimal syntactic overhead in the token sequence [27]. Less strictly structured representations were considered to examine the influence of structuring degree on embedding quality. These include natural language prose (SF6) and a simple key-value representation in Markdown style (SF7).

Structured serialization formats (SF1–SF5) were implemented using established Python libraries. For SF6, model element contents were embedded into predefined textual templates to ensure consistent representation (see Listing 1). SF7 was implemented analogously, with attribute names and their corresponding values explicitly represented as formatted pairs (see Listing 2).

3.4 Embedding Models

We evaluate 13 embedding models (Table 3), covering a wide range of sizes, architectures, and benchmark backgrounds. BM25 (EM02) serves as a sparse retrieval baseline.

Model selection was based on performance in established benchmarks. We considered both text embedding benchmarks such as MTEB [19] and MMTEB [6], as well as code embedding benchmarks including CoIR [17]. Models of varying sizes and embedding dimensions were selected.

During the embedding process, each model element was individually transformed into a vector representation. When a serialized representation exceeded a model's maximum context length, the text was truncated accordingly.

4 Evaluation Setup

The evaluation component serves to quantitatively assess the retrieval results generated by the artifact embedding component.

4.1 Dataset

The approach was evaluated on two PREEvision models (see Table 4). The Demo Model, a model incorporating practical examples of many different PREEvision features, and the vCANdrive model, a small model for fair demonstrations of the whole PREEvision toolchain. Both models are not real-world models, but they are realistic in the sense that they were created by PREEvision users for demonstration purposes, and they contain a variety of different model elements and relationships. For our tests, we only embed a subset of the model elements, namely the EEA artifacts, which are the model elements that hold semantic information about the system being developed. Other model elements, such as folders or diagrams, are not embedded directly. For our data preprocessing, we include relations to non-EEA model elements. Most configurations were only evaluated for the vCANdrive model because the combinatorial complexity, in addition to the number of model elements, made it impractical to evaluate all combinations on the bigger Demo Model.

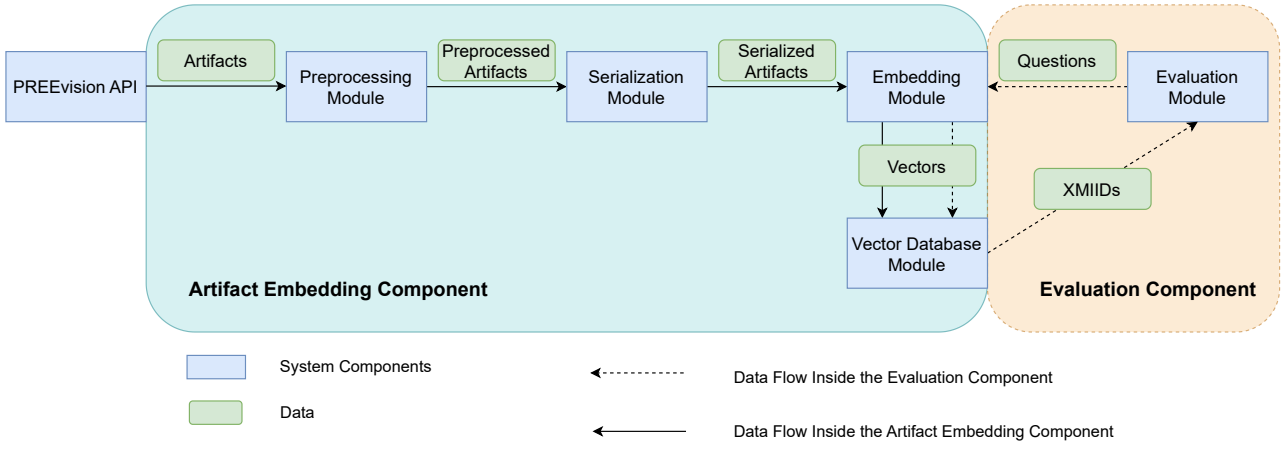


Figure 1: Ingestion pipeline architecture.

Figure 2: Data preprocessing strategies.

Table 1: Preprocessing strategies. Metrics are on average for EM10. Best values are in bold, non-significant worse values are underlined ($\alpha = 0.05$). Token count with Qwen3 tokenizer.

ID	Strategy	Description	nDCG@10	Recall@20	Recall@100	MRR	Mean Tokens/Model element
DV1	*plain*	No preprocessing (baseline)	0.39	0.36	0.61	0.88	755.37
DV2	*mopilot*	Combination of DV3–DV7	0.52	**0.72**	**0.92**	0.72	**301.99**
DV3	filterAttributes	Remove generic/technical metadata	0.36	0.36	0.61	0.80	389.80
DV4	addLayer	Append artifact layer label	0.38	0.36	0.61	0.85	759.04
DV5	convertMofType	Normalize MOF type names	0.40	0.36	0.62	**0.90**	720.92
DV6	processLinks	Dereference links	**0.52**	0.66	0.88	0.81	698.62
DV7	extendFileContent	Include external file contents	0.38	0.35	0.61	0.86	755.01

Table 2: Serialization formats. Metrics are averaged across all embedding models and data preprocessing configurations. Best values are in bold, non-significant worse values are underlined ($\alpha = 0.05$). Token count with Qwen3 tokenizer.

ID	Format	Description	nDCG@10	Recall@20	Recall@100	MRR	Mean Tokens/Model element
SF1	JSON	JavaScript Object Notation	0.32	0.39	0.60	0.57	668.54
SF2	XML	eXtensible Markup Language	0.29	0.36	0.55	0.51	813.02
SF3	YAML	YAML Ain't Markup Language	0.32	0.40	0.59	0.58	**568.75**
SF4	TOML	Tom's Obvious. Minimal Language	0.33	0.41	0.62	0.60	575.57
SF5	TOON	Token-Oriented Object Notation	0.34	0.42	**0.63**	0.61	871.96
SF6	Natural text	Template-based prose	0.33	0.40	0.61	0.61	594.73
SF7	Markdown KV	Markdown-style key-value pairs	**0.36**	**0.43**	0.62	**0.67**	588.22

Table 3: Evaluated embedding models with MTEB Retrieval Score and performance metrics. For performance metrics, best values are in bold, non-significant worse values are underlined. ($\alpha = 0.05$)

ID	Model	Params	nDCG@10	Recall@20	Recall@100	MRR	MMTEB Retrieval Score [6]
EM01	all-MiniLM-L6-v2	22M	0.24	0.35	0.61	0.35	32.51
EM02	BM25 (sparse baseline)	—	0.38	0.41	0.41	**0.95**	-
EM03	bge-large-en	335M	0.38	0.54	0.82	0.60	-
EM04	bge-m3	569M	0.35	0.50	0.78	0.58	54.6
EM05	inf-retriever-v1	7.1B	0.51	0.72	0.91	0.77	66.48
EM06	KaLM-Embedding-Gemma3-12B	11.8B	**0.57**	0.73	0.94	0.87	75.66
EM07	nomic-embed-code	7.0B	0.43	0.58	0.89	0.80	-
EM08	Octen-Embedding-8B	7.6B	0.49	0.64	0.87	0.90	71.68
EM09	Qwen3-Embedding-0.6B	596M	0.51	0.69	0.92	0.87	64.65
EM10	Qwen3-Embedding-4B	4.0B	0.56	**0.80**	**0.95**	0.94	69.6
EM11	Qwen3-Embedding-8B	7.6B	0.53	0.71	0.93	0.91	70.88
EM12	snowflake-arctic-embed-l-v2	568M	0.42	0.60	0.83	0.68	58.36
EM13	text-embedding-3-large	unknown	0.34	0.51	0.83	0.62	59.32

```
The following document describes an artifact with its attributes:
The XMIID is 791.
The name is Front_0.
There are the following compositeRelations:
  [...]
  - The artifact is connected with the relation componentPackage
to the attribute ECUs (eea.componentslayer.ComponentPackage) -
xmiid=697.
  [...]
There are the following associations:
  - The artifact is connected with the relation referencedBySMB to
1 artifact Reference SMB81 (metricmm.ReferenceSMB) - xmiid=2509.
  [...]
The mofType has the name ECU and the fullname
eea.componentslayer.ECU.
```

Listing 1: Example of SF6 - Natural text serialization

Table 4: PREEvision models. Evaluated PREEvision models with information

Name	# Elements	# EEA Elements	Q-A Pairs
Demo Model	1,477,124	63,914	750
vCANdrive	1,299,469	11,941	62

```
# 791 (Front_0):
```
XMIID: 791
name: Front_0
  [...]
compositeRelations:
  - rel: componentPackage
    artifacts:
      ECUs (eea.componentslayer.ComponentPackage) - xmiid=697
  [...]
associations:
  - rel: referencedBySMB
    size: 1
    artifacts:
      Reference SMB81 (metricmm.ReferenceSMB) - xmiid=2509
  [...]
mofType:
  name: ECU
  fullname: eea.componentslayer.ECU
```
```

Listing 2: Example of SF7 - Markdown KV serialization

4.2 Question Set

The benchmark question set comprises questions, answers, and the artifact IDs required for answering. Although the structure of a PREEvision model's element tree may vary, certain structural patterns are likely to recur across different models. To generate evaluation inputs, we apply model-to-model transformations to match predefined search patterns within the models. The identified

model element patterns are then automatically transformed into corresponding question-answer pairs, with each pair explicitly referencing the artifacts from which it was derived. A key advantage of this approach lies in its model-agnostic nature, as the transformation rules are defined in terms of the metamodel's classes and can be applied uniformly to any model that conforms to the PREEvision metamodel without requiring model-specific adaptations.

The questions operationalize six specific structural patterns of the PREEvision model. They evaluate the embedding models' ability to retrieve relevant information even when answers depend on referenced artifacts or are distributed across multiple artifacts. Answering individual questions requires identifying chains of connected artifacts. For example, the question "Which ECU is connected to the actuator LedMatrixFrontLeft?" requires identifying five artifacts (actuator, two connectors, cable, and ECU) to generate the answer "Front_0". Other question categories have different structures and may have multiple solutions, typically requiring shorter connected paths. The number of relevant model elements varies by question, ranging from 3 to 16.

4.3 Evaluation Module

In the evaluation step, the questions are transformed into vector representations using the same embedding model as the model elements. Depending on the specific embedding model, a model-specific prompt may be prepended to the question. If any, we followed the embedding model's recommended prompt for retrieval tasks. Subsequently, the most similar model element IDs to the question vector are retrieved from the vector database using cosine similarity.

The determined retrieval results are compared with the IDs marked as relevant for each respective question. Based on this comparison, evaluation metrics are calculated. This is done first at the level of individual questions and then aggregated across all questions. The flow of this evaluation process is illustrated in Figure 1.

4.4 Metrics

We follow the MTEB [19] and BEIR [23] conventions, reporting: **nDCG@10** (primary metric, ranks-aware quality of the top-10 results), **Recall@20**, **Recall@100** (coverage of all relevant artifacts at practical LLM context sizes), and **MRR** (mean reciprocal rank of the first relevant artifact). We calculate statistical significance for these metrics only. For the significance test, we use a paired t-test as proposed by Urbano et al. [24].

5 Results

We present our results by our research questions (see section 1). In addition, we also evaluate the factor importance in subsection 5.5.

5.1 RQ1: Model-RAG feasability

Embedding-based retrieval of single model elements is a feasible approach for RAG systems. Table 5 shows the 20 best evaluated combinations of data preprocessing, serialization, and embedding model sorted by nDCG@10. It also includes the best combination of plain data preprocessing and the one with the highest MRR.

Table 5: Excerpt of the 20 best evaluated combinations sorted by nDCG@10 on the PREEvision model vCANdrive. Additionally, the first combination with *DV1* data preprocessing and the one with the highest MRR are shown. Best values per metric are in bold, non-significantly lower values ($\alpha = 0.05$) are underlined.

#	Model	DV	SF	$nDCG_{10}$	$Recall_{20}$	$Recall_{100}$	MRR
1	EM10	DV6	SF5	**0.59**	0.74	0.91	0.82
2	EM06	DV2	SF3	0.57	0.73	0.94	0.82
3	EM10	DV2	SF1	0.56	0.76	0.93	0.78
4	EM06	DV2	SF4	0.55	0.70	0.92	0.87
5	EM10	DV2	SF5	0.55	**0.80**	0.93	0.70
6	EM10	DV6	SF4	0.54	0.67	0.91	0.86
7	EM10	DV2	SF4	0.54	0.73	0.94	0.76
8	EM06	DV2	SF5	0.54	0.71	0.93	0.81
9	EM11	DV2	SF2	0.53	0.71	0.91	0.74
10	EM10	DV2	SF6	0.53	0.72	**0.95**	0.75
11	EM10	DV6	SF3	0.52	0.72	0.90	0.78
12	EM10	DV6	SF1	0.52	0.64	0.86	0.84
13	EM11	DV2	SF3	0.52	0.68	0.90	0.77
14	EM10	DV2	SF7	0.52	0.67	0.91	0.76
15	EM10	DV6	SF2	0.51	0.63	0.84	0.86
16	EM09	DV2	SF1	0.51	0.67	0.90	0.69
17	EM05	DV2	SF5	0.51	0.72	0.91	0.61
18	EM05	DV2	SF6	0.51	0.72	0.89	0.67
19	EM06	DV2	SF1	0.50	0.62	0.87	0.85
20	EM10	DV2	SF3	0.50	0.73	0.91	0.67
52	EM11	DV1	SF3	0.42	0.46	0.64	0.89
91	BM25	DV1	SF6	0.38	0.26	0.26	**0.95**

Table 6: Selected configurations on the bigger Demo Model evaluation set with performance metrics. Best values are in bold; non-significant worse values are underlined. ($\alpha = 0.05$)

EM	DV	SF	$nDCG_{10}$	$Recall_{20}$	$Recall_{100}$	MRR
EM05	DV2	SF5	0.44	0.60	0.74	0.66
EM06	DV2	SF4	**0.49**	0.59	0.71	**0.78**
EM06	DV2	SF5	0.48	0.59	0.71	0.73
EM06	DV2	SF3	0.45	0.55	0.70	0.76
EM09	DV2	SF1	0.48	0.59	0.71	0.72
EM10	DV2	SF1	0.49	0.62	0.77	0.75
EM10	DV2	SF6	0.48	0.60	0.77	0.76
EM10	DV2	SF4	0.44	0.58	0.76	0.69
EM10	DV2	SF5	0.48	**0.62**	**0.78**	0.70
EM11	DV2	SF2	0.48	0.60	0.74	0.73

BM25 serves as the classical sparse retrieval baseline, considered in the literature as a robust reference method for retrieval tasks [23]. Good-performing configurations achieve nDCG@10 above 0.5, which is a significant improvement over the BM25 baseline with around 0.38 (see Table 5). We also retrieve more than 70% of relevant model elements in the top 20 retrieved model elements, and more than 90% in the top 100 retrieved model elements. The highest MRR value of 0.95 was achieved by BM25, which is a side effect of our evaluation setup with automatically generated questions that contain hardly any synonym variations. BM25's actual performance in real search scenarios might be even lower [14].

Table 6 shows the results of selected good configurations on a bigger Demo Model evaluation set. The results are consistent with the smaller evaluation set, with the best-performing configuration achieving an nDCG@10 of 0.49 and an MRR of 0.78. However, the recall values are generally lower, which may be due to the increased model size and complexity. This suggests that while embedding-based retrieval is effective for finding relevant model elements, it may struggle to retrieve all relevant model elements in larger or more complex models.

Answer to RQ1: Embedding-based retrieval is a feasible approach for model element retrieval in RAG systems, significantly outperforming classical keyword-based methods like BM25. However, embedding models can't find all relevant model elements.

5.2 RQ2: Embedding Model Comparison

In Table 3, we first compare the retrieval performance of all embedding models. For each model, the remaining pipeline components (serialization and preprocessing) are selected to use the best-performing configuration identified. This approach reflects real-world application scenarios where models are typically not deployed in suboptimal combinations. Thus, we compare the achievable performance upper bounds of the models. However, this approach may introduce a certain hyperparameter bias and favor data-specific outliers.

KaLM-Embedding-Gemma3-12B, currently leading in the MTEB Retrieval benchmark, demonstrates strong performance in nDCG@10 but less frequently places relevant model elements at the top rank position. Qwen3-Embedding-4B achieves the best overall retrieval performance, delivering consistently high values across all metrics. It is not significantly worse than the respective best model in any considered metric. The larger sibling model Qwen3-Embedding-8B achieves slightly weaker results despite having double the parameter count, suggesting that higher model capacity does not necessarily lead to better retrieval performance.

The smallest embedding models perform overall worse. A possible cause is their limited context length, which may truncate relevant information located at the end of model elements.

The recall progression over various cut-off values k shown in Figure 3 also suggests that keyword-based methods like BM25 find relevant documents early but provide no additional results beyond the first five returned model elements. In contrast, dense embedding models continue to improve recall even at higher cut-off values. The superiority of powerful embedding models becomes particularly evident in the range of $k \approx 10$–200 returned model elements. This range is especially relevant for RAG systems, as it approximately corresponds to the number of model elements that fit into the context window of downstream LLMs. For higher cut-off values $k \to 1000$, the recall of embedding models converges again, and differences between models diminish.

Overall, modern dense embeddings clearly outperform classical keyword-based methods. In particular, Qwen3 and KaLM models achieve consistently high performance. This confirms that the semantic representation capability of models is crucial for system success.

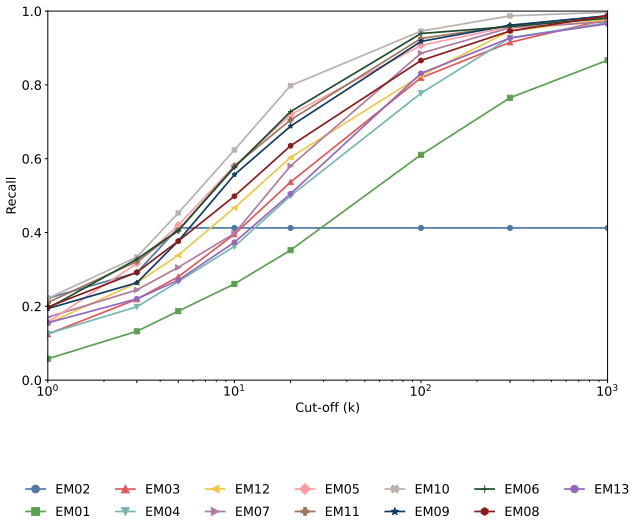


Figure 3: Recall progression over cut-off values k.

MMTEB transfer. We compare the observed model rankings with their performance in the MMTEB Retrieval benchmark [6]. The MMTEB retrieval score has a high correlation with the nDCG@10 values observed in our benchmark, indicating that models performing well on MMTEB also tend to perform well in our model retrieval context ($R^2 = 0,8566$). Also, the Kendall rank correlation coefficient between the model rankings in MMTEB and our benchmark is 0.64, which is statistically significant ($p = 0.01$), indicating a strong agreement in the relative performance of the models across both benchmarks.

Answer to RQ2: Embedding models performing well in normal text retrieval benchmarks like MTEB also perform the best in our model retrieval benchmark, with a high correlation between the rankings.

5.3 RQ3: Serialization Format

Table 2 compares the retrieval performance of different serialization formats. On average, SF7 (Markdown KV) achieves the best performance, closely followed by SF5 (TOON), indicating good stability over different embedding models. However, the other formats also appear frequently among the best configurations (see Table 5).

XML (SF2) has the lowest average score across all metrics (see Table 2), yet still appears twice among the top 15 overall combinations (Table 5).

XML followed by JSON has the highest average token count per model element (see Table 3). TOON and Yaml have the lowest token count on average, which is consistent with related work.

Answer to RQ3: Markdown KV and TOON serialization formats achieve the best retrieval performance on average, but other formats are also frequently among the top configurations, indicating embedding model-specific effects.

Table 7: ANOVA results for nDCG@10.

	df	SS	MS	F	p	η_p^2
DV	1.0	0.63	0.63	633.68	< 0.001	0.90
SF	6.0	0.07	0.01	11.74	< 0.001	0.49
EM	12.0	1.96	0.16	164.86	< 0.001	0.97
DV × SF	6.0	0.01	0.001	1.19	0.32	0.09
DV × EM	12.0	0.26	0.02	21.82	< 0.001	0.78
SF × EM	72.0	0.14	0.002	2.03	0.002	0.67
Error	72.0	0.07	0.001	–	–	–

5.4 RQ4: Preprocessing

Table 1 compares the retrieval performance of different data preprocessing strategies for Qwen3-Embedding-4B. The *mopilot* strategy, which combines multiple preprocessing steps, achieves the best overall performance. However, the single step of dereferencing links (DV6) also achieves a significant improvement over the plain baseline, suggesting that this step is particularly effective in enhancing retrieval performance. The other individual preprocessings don't show much improvement over the plain baseline.

DV3 reduces the token count to almost 50% of the original while preserving the same retrieval performance. This prerpocessing might be the single best optoin for reducing inference cost of embedding models, while preserving overall performance. DV7 does not increase the token count much due to the small size of the linked files.

The only tested embedding model that does not benefit from link dereferencing is BM25 (see Figure 4). This comes naturally, as this preproessing step distributes important keywords further into non-relevant chunks.

Answer to RQ4: The choice of data preprocessing strategy has a significant impact on retrieval performance. Including model structure by dereferencing links is particularly effective in improving retrieval performance. The amount of attributes does not have a big effect on retrieval performance.

5.5 Factor Importance

To quantify the influence of different factors on retrieval performance, we conducted a multifactorial ANOVA. Table 7 presents the results of the analysis for nDCG@10. The analysis reveals that the embedding model has the largest effect on retrieval performance ($\eta_p^2 = 0.97$), followed by data preprocessing ($\eta_p^2 = 0.90$) and serialization format ($\eta_p^2 = 0.49$). All main effects are statistically significant ($p < 0.001$).

Figure 4 shows a heatmap of nDCG@10 values for all combinations of embedding models and serialization strategies for plain and mopilot preprocessings. The heatmap shows that while the choice of embedding model has a dominant influence on performance, the impact of serialization format is more nuanced and varies with the embedding model used.

6 Discussion

In this section, we discuss the implications of our findings for practitioners and researchers and address potential threats to the validity of our study.

6.1 Implications

For practitioners, our results confirm that embedding-based retrieval over MDE model elements is viable today using general-purpose text embedding models without any domain-specific fine-tuning. In more than half of our test cases, embedding models found a relevant model element first (Median MMR = 1, see Figure 5). Also, modern embedding models, like Qwen3-Embedding-4B are superior for model element retrieval in comparison to keyword-based mechanisms, like BM25. This also holds if there are no chunks that hold the whole searched information, as embedding models can also reliably find partial searched information.

MTEB retrieval scores serve as a reliable proxy for model selection ($R^2 = 0.86$, Kendall $\tau = 0.64$), so practitioners can consult publicly available MTEB leaderboards to guide their choice. Among the evaluated models, Qwen3-Embedding-4B offers the best cost-performance trade-off; its larger sibling Qwen3-Embedding-8B is marginally weaker despite doubling the parameter count, highlighting that model size alone should not drive selection.

Preprocessing is the single most impactful tuning lever available without touching the embedding model itself. The most effective individual step is link dereferencing (DV6), which raises nDCG@10 from 0.39 to 0.52 by inlining the name, type, and identifier of referenced model elements—making the local relational context explicit without increasing chunk count. Practitioners are advised to enable this step first before experimenting with other strategies.

For serialization, Markdown key-value (SF7) and TOON (SF5) offer the most robust performance across models and produce substantially fewer tokens per model element than XML or JSON, reducing both embedding latency and context consumption. SF7 can therefore serve as a safe default, with model-specific tuning reserved for scenarios where the heatmap in Figure 4 indicates a different format is clearly better.

Regarding retrieval depth, a cut-off of $k \approx 20$ recovers at least one relevant model element reliably, while $k \approx 100$ is still not enough to cover the full set of relevant model elements for multi-hop queries – especially for larger MDE models, like the Demo Model. RAG system designers should account for this by combining embedding retrieval as an anchor point finder with explicit graph traversal for multi-hop scenarios or large-scale MDE models.

For researchers, the strong MTEB transferability result offers a practical starting point: standard text retrieval benchmarks remain informative predictors even for structured model elements, so extending MTEB rather than building entirely new benchmarks may be a fruitful direction. Also, exploring other established information retrieval techniques, such as hybrid retrieval, might transfer well.

A key open challenge revealed by this work is the *anchor-point limitation*: even the best models find the directly queried model elements reliably but struggle to retrieve the full set of implicitly required model elements along the relational chain. This suggests that standard dense retrieval is insufficient for multi-hop model queries, motivating future work on graph-aware retrieval strategies, such as iterative retrieval, re-ranking based on structural proximity, or hybrid approaches that combine dense retrieval with explicit relation traversal.

The ANOVA interaction between serialization format and embedding model ($\eta_p^2 = 0.67$, $p = 0.002$) shows that different models

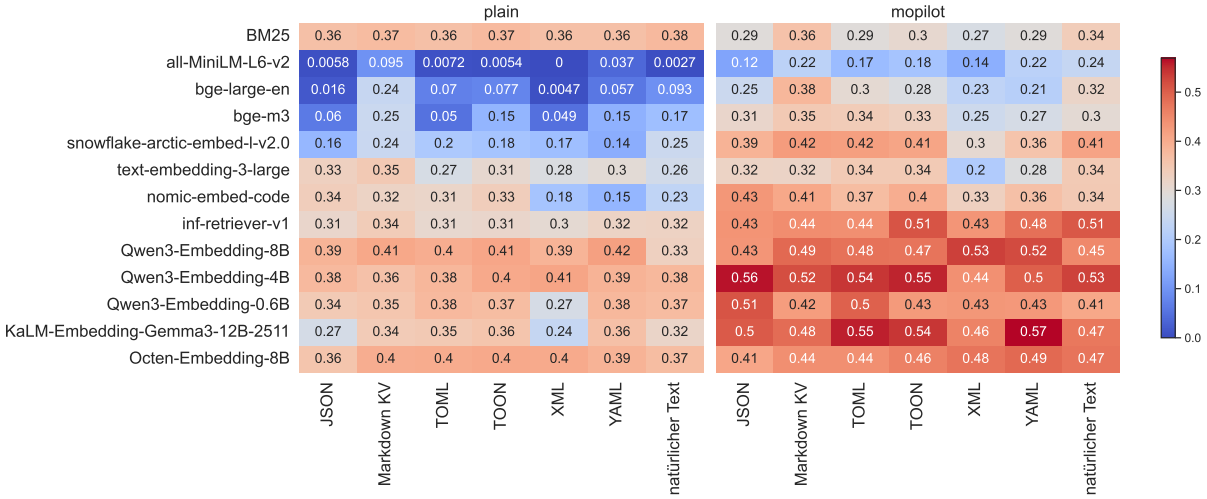


Figure 4: Heatmap of nDCG@10 values for all combinations of preprocessing and serialization strategies.

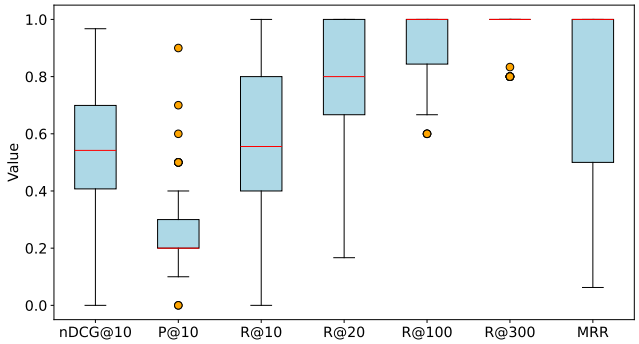


Figure 5: Boxplot of different metrics per question for the combination of Qwen3-Embedding-4B, mopilot, and JSON.

respond differently to format syntax: some models are robust across all formats, while others are sensitive to specific representations. This interaction warrants embedding-model-specific format selection in future work and cautions against evaluating serialization strategies with a fixed embedding model.

The dominance of link dereferencing over all other preprocessing steps (including format-level choices) further suggests that *structural context enrichment* (making the model graph locally explicit) is more important than choosing the right serialization syntax. Investigating richer forms of neighborhood expansion, such as multi-hop link dereferencing, graph-aware embeddings, or type hierarchy inclusion, is a promising avenue.

6.2 Threats to Validity

Following the guidelines of Runeson and Höst [20], we outline and address potential threats to the validity of our research.

Construct validity. The benchmark questions were generated automatically from human-implemented metamodel-level transformation patterns. While this ensures reproducibility and avoidance of model-specific bias, the resulting questions variation is mainly driven by the different question templates and specific model elements, which artificially inflates BM25 MRR and may not reflect the diversity of natural-language queries engineers formulate in practice. The boxplot in Figure 5 illustrates that performance varies across questions. Furthermore, the retrieval metrics adopted from MTEB and BEIR (nDCG@10, Recall@20/100, MRR) measure ranking quality; they do not directly capture whether the retrieved context is sufficient for a downstream LLM to generate a correct answer, leaving end-to-end RAG quality for future evaluation.

Internal validity. The comparison of embedding models at their individually optimized configurations (best serialization and preprocessing combination) may introduce hyperparameter bias, as models that are sensitive to a particular combination could appear artificially stronger. Long serializations are truncated at each model's context length, meaning that large model elements may be evaluated under different conditions depending on the model's token budget. The DV×EM interaction ($\eta_p^2 = 0.78$) confirms that preprocessing gains are not uniform across models, so aggregate rankings may mask individual model behavior.

External validity. The study was conducted exclusively on PREEvision EEA models (Demo Model and vCANdrive). Both are realistic but synthetic demonstration models rather than real production-use models. While PREEvision captures many different domains within its metamodel, both evaluation MDE models originate from the same tool and metamodel. Transferability to other MDE tools (such as Palladio and SysML) and other modeling domains (e.g., process models, software architecture) has not been verified.

Reliability. All retrieval experiments use exact nearest-neighbor search, ensuring fully deterministic and reproducible results. The evaluation pipeline is automated end-to-end: question generation, embedding, retrieval, and metric computation are parameterized by configuration files and can be re-executed on any PREEvision-conformant model using the same transformation rules.

7 Conclusion

We present an empirical study on embedding-based retrieval of MDE model elements for use in RAG pipelines, comparing 13 embedding models, 7 serialization formats, and 7 data preprocessing strategies on PREEvision EEA models. Our results demonstrate that general-purpose text embedding models can be applied to structured model elements without domain-specific finetuning, and that MTEB leaderboard rankings transfer reliably to this new domain. Aside of embedding models, preprocessing—especially link dereferencing—has the largest impact on retrieval quality, while compact serialization formats offer a robust and token-efficient default.

At the same time, our study reveals a limitation of dense retrieval for MDE: embedding models reliably identify directly queried anchor elements but often struggle to recover the full set of implicitly required model elements along relational chains. Closing this gap through graph-aware retrieval strategies—such as iterative multi-hop retrieval, re-ranking by structural proximity, or hybrid dense–graph approaches—is the primary open challenge and the most promising direction for future work.

Acknowledgments

This work was funded by the German Research Foundation (DFG) - SFB 1608 - 501798263 and Core Informatics at KIT (KiKIT) of the Helmholtz Assoc. (HGF).

References

[1] Nico Adler, Jörg Schäuffele, and Tomasz Witkowski. 2025. *Seamless, agile, and holistic automotive systems engineering with PREEvision: How to future-proof systems engineering practices for E/E systems.* White Paper. Vector Informatik GmbH. https://www.vector.com/int/en/download/preevision-white-paper-systems-engineering/

[2] Nastaran Babanejad, Ameeta Agrawal, Aijun An, and Manos Papagelis. 2020. A Comprehensive Analysis of Preprocessing for Word Representation Learning in Affective Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 5799–5810. doi:10.18653/v1/2020.acl-main.514

[3] Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: A comparison of retrieval performances. *Lecture notes on software engineering* 2, 3 (2014), 262.

[4] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The Faiss library. arXiv:2401.08281 [cs.LG]

[5] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL]

[6] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. 2025. MMTEB: Massive Multilingual Text Embedding Benchmark. arXiv:2502.13595 [cs.CL]

[7] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. 2020. A Survey on Knowledge Graph Embeddings with Literals: Which model links better Literal-ly? arXiv:1910.12507 [cs.AI]

[8] Daniel Gomm and Madelon Hulsebos. 2025. Metadata Matters in Dense Table Retrieval. In *ELLIS workshop on Representation Learning and Generative Models for Structured Data.* https://openreview.net/forum?id=rELWIvq2Qy

[9] Martin Grohe. 2020. word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. arXiv:2003.12590 [cs.LG]

[10] José Antonio Hernández López, Carlos Durá, and Jesús Sánchez Cuadrado. 2023. Word Embeddings for Model-Driven Engineering. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS).* 151–161. doi:10.1109/MODELS58315.2023.00036

[11] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open Domain Question Answering over Tables via Dense Retrieval. arXiv:2103.12011 [cs.CL]

[12] MUHİTTİN IŞIK and Hasan Dağ. 2020. The impact of text preprocessing on the prediction of review ratings. *Turkish Journal of Electrical Engineering and Computer Sciences* 28, 3 (2020), 1405–1421.

[13] Angelika Kaplan, Jan Keim, Marco Schneider, and Ralf Reussner. 2025. HubLink: A Novel Question Answering Retrieval Approach over Knowledge Graphs. In *RAGE-KG 2025: The Second International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs (RAGE-KG 2025) co-located with 24th International Semantic Web Conference (ISWC 2025). Ed.: D. Dobrij (CEUR Workshop Proceedings, Vol. 4079).* CEUR-WS, 1–18. 46.23.01; LK 01.

[14] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906 [cs.CL]

[15] Florian Kunneman, Thiago Castro Ferreira, Emiel Krahmer, and Antal van den Bosch. 2019. Question Similarity in Community Question Answering: A Systematic Exploration of Preprocessing Methods and Models. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, Ruslan Mitkov and Galia Angelova (Eds.). INCOMA Ltd., Varna, Bulgaria, 593–601. doi:10.26615/978-954-452-056-4_070

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL]

[17] Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Hao Zhang, Xinyi Dai, Yasheng Wang, and Ruiming Tang. 2025. CoIR: A Comprehensive Benchmark for Code Information Retrieval Models. arXiv:2407.02883 [cs.IR]

[18] Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yu Gu, Zhiyuan Liu, and Ge Yu. 2023. Structure-Aware Language Model Pretraining Improves Dense Retrieval on Structured Data. arXiv:2305.19912 [cs.IR]

[19] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, Andreas Vlachos and Isabelle Augenstein (Eds.). Association for Computational Linguistics, Dubrovnik, Croatia, 2014–2037. doi:10.18653/v1/2023.eacl-main.148

[20] Per Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2008), 131. doi:10.1007/s10664-008-9102-8

[21] Gerald Schaefer. 2008. Does compression affect image retrieval performance? *International Journal of Imaging Systems and Technology* 18 (08 2008), 101 – 112. doi:10.1002/ima.20152

[22] Jörg Schäuffele. 2016. E/E architectural design and optimization using PREEvision. In *SAE 2016 World Congress and Exhibition 217486.*

[23] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs.IR]

[24] Julián Urbano, Harlley Lima, and Alan Hanjalic. 2019. Statistical Significance Testing in Information Retrieval: An Empirical Analysis of Type I, Type II and Type III Errors. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19).* Association for Computing Machinery, New York, NY, USA, 505–514. doi:10.1145/3331184.3331259

[25] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. 2026. Retrieval-Augmented Generation for Natural Language Processing: A Survey. arXiv:2407.13193 [cs.CL]

[26] Xin Ye, Hui Shen, Xiao Ma, Razvan Bunescu, and Chang Liu. 2016. From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16).* Association for Computing Machinery, New York, NY, USA, 404–415. doi:10.1145/2884781.2884862

[27] Haoteng Yin, Jinha Kim, Prashant Mathur, Krishanu Sarker, and Vidit Bansal. 2025. How to Talk to Language Models: Serialization Strategies for Structured Entity Matching. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 7851–7865. doi:10.18653/v1/2025.findings-naacl.437