Generalized Embedding Models for Industry 4.0 Applications

Christodoulos Constantinides[1], Shuxin Lin[2], Dhaval Patel[2]
[1]IBM, [2]IBM Research
{christodoulos.constantinides@, shuxin.lin@, pateldha@us.}ibm.com

Abstract

In this work, we present the first embedding model specifically designed for Industry 4.0 applications, targeting the semantics of industrial asset operations. Given natural language tasks related to specific assets, our model retrieves relevant items and generalizes to queries involving similar assets, such as identifying sensors relevant to an asset's failure mode. We systematically construct nine asset-specific datasets using an expert-validated knowledge base reflecting real operational scenarios. To ensure contextually rich embeddings, we augment queries with Large Language Models, generating concise entity descriptions that capture domain-specific nuances. Across five embedding models ranging from BERT (110M) to gte-Qwen (7B), we observe substantial in-domain gains: **HIT@1 +54.2%, MAP@100 +50.1%, NDCG@10 +54.7%** on average. Ablation studies reveal that (a) LLM-based query augmentation significantly improves embedding quality; (b) contrastive objectives without in-batch negatives are more effective for tasks with many relevant items; and (c) balancing positives and negatives in batches is essential. We evaluate on a new task and finally present a case study wrapping them as tools and providing them to a planning agent. The code can be found here.

1 Introduction

As Large Language Models (LLMs) advance, automating a wide range of tasks spanning from general activities like reading emails to specialized, domain-specific queries, has emerged as the next frontier of innovation. For example, in industrial settings, a plant operator may request optimal setpoints for critical control variables, or an industrial data scientist might seek assistance in identifying key sensor variables for predictive maintenance. Increasingly, LLM agents adopting specialized personas aim to handle such tasks by combining LLM reasoning with domain tools accessed via APIs.

Recent examples include MDAGENT (Kim et al., 2024), a multi-agent system developed for medical decision support. While effective at guiding users through predefined options, MDAGENT relies solely on LLMs' internal knowledge, limiting its ability to address complex, real-world scenarios requiring dynamic, context-sensitive reasoning. This highlights a critical gap: the need for systems capable of integrating domain knowledge flexibly to support multifaceted decision-making.

Recommender systems offer a compelling solution, excelling at assisting users in navigating complex decision spaces by leveraging historical data, item properties, and user interactions. Embedding models integrated into LLM-augmented recommenders have recently demonstrated promise, especially in domains such as entertainment (Gao et al., 2023), by enhancing interactivity and explainability. A key advancement driving this progress is the emergence of instruction-driven, domain-specific embeddings (Anderson et al., 2024; Xu et al., 2024; Weller et al., 2024; Li et al., 2024), which tailor embeddings using task-specific prompts to capture nuanced semantics. These methods outperform general-purpose embeddings by ensuring contextually relevant retrieval. Moreover, leveraging LLMs as teacher models to generate synthetic data has further improved domain-specific embedding training (Wang et al., 2024).

Collectively, these trends underscore the need for domain-specialized embedding models that combine LLM capabilities with targeted contextual understanding, enabling flexible, accurate decision support in complex industrial environments.

1.1 Challenges in Industry 4.0

Despite the potential of domain-specific embedding techniques as powerful tools within a LLM-driven framework, their application in Industry 4.0 remains under-explored. Implementing embedders in industrial settings presents unique challenges.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 2234–2251
November 4-9, 2025

One key challenge is the accessibility of specialized knowledge. Actionable, fine-grained information is often embedded in technical documents, such as International Standards Organization (ISO) manuals, which are not structured for immediate use and are difficult to access in practice.

A second challenge lies in task-specific instruction. Instruction-tuned embedders depend on well-defined domain-relevant guidance, yet much of the necessary instructional knowledge in industrial contexts is informal or remains partially formalized.

A third concern involves asset-related knowledge. Industrial environments comprise a large number of heterogeneous assets, but the amount of information available per asset is often sparse. Developing techniques to augment and cross-leverage knowledge across assets is still an open problem.

To address these limitations, we propose a multi-task, asset-specific fine-tuning strategy that aims to bridge domain gaps and enable more context-aware, robust intelligence in industrial decision making.

Our main contributions are as follows:

- We formalize tasks for the Industrial domain and introduce the **first embedding models specialized for the industrial domain**, designed to assist SMEs with maintenance tasks. We identify nine tasks from ISO documents and train a multi-task embedder (Figure 2) to retrieve the final answer rather than generating it. We demonstrate an average increase of ACC@1 by 54% for our use case. To address the data scarcity issue, we provide the prepared dataset for future research work.

- We demonstrate through ablation studies that popular representation learning methods that use in-batch negatives are prone to false negatives in certain data settings. Additionally, we highlight the importance of maintaining a balance between positive and negative samples within the batch.

- We investigate the use of our domain-specific embedder on a new unseen task. Additionally, as a case study (see Section 6), we show how this multi-task embedder can serve as a suite of tools (Figure 7) invoked by a ReAct agent (Yao et al., 2022) for planning and reasoning on industrial queries.

2 Industrial Multi-Task Embedder

2.1 Foundational Concepts in Industry 4.0

This section introduces key terminology and foundational concepts relevant to tasks and frameworks in Industry 4.0 applications, particularly in the areas of predictive maintenance, asset management, and sensor-based monitoring. We define six core concepts which are referred here as **items**, that are central to understanding and implementing industrial AI systems.

Asset: A physical resource or piece of equipment used in industrial operations or production processes. Examples include electric generators, transformers, and wind turbines.

Equipment Class: A grouping of assets based on shared functional or operational characteristics. For example, "combustion engines" form an equipment class that includes subtypes like "diesel engines" and "gas turbines".

Equipment Type: A specific category within an equipment class that characterizes the function or configuration of an asset. A diesel power generator, for instance, is an equipment type under the "diesel engine" class.

Failure Mode: A particular way in which an asset can fail, including forms of degradation or malfunction. An example is "insulation deterioration" in electric motors.

Sensor: A device used to measure or monitor a physical parameter such as temperature, vibration, pressure, or rotational speed, often for the purpose of detecting abnormal conditions.

Subunit: A defined functional component within a larger system, typically responsible for a specific task. An example is the "fuel feed system" in a boiler, which manages the delivery of fuel to the combustion chamber.

2.2 Data Collection

We collected documents from the International Organization for Standardization (ISO) containing information on maintaining industrial assets (ISO, 2018, 2016). The data were originally in diverse tabular formats. To illustrate the relationships between assets (e.g., motors, machines) and their associated sensors (e.g., temperature, environmental), we constructed a sample bipartite graph. In Appendix Figure 9, assets are shown as purple nodes and sensors as yellow nodes, with directed edges representing their relationships. For a given asset,

connected nodes denote **positive documents**, while missing edges correspond to **negative documents**.

2.3 Tasks Definition

We identify **nine distinct tasks** that are integral to our system, each targeting a specific aspect of industrial asset understanding. The underlying data used for generating tasks is extract from structured tabular formats as discussed in Section 2.2.

Asset to Sensors (**A2S**): Identify relevant sensors that can monitor the condition of a given asset.

Component to Failure Mode (**C2FM**): Possible failures related with a given asset component.

Equipment Class Type to Category (**E2CAT**): Determine the category corresponding to a given equipment class type.

Equipment Category to Class Type (**E2CLT**): Identify class types that fall under a given equipment category.

Equipment Unit to Subunit (**EU2SU**): Identify subunits belonging to a given equipment unit.

Failure Mode to Class (**FM2CLS**): Classify a failure mode under its appropriate failure class.

Failure Mode to Components (**FM2CMP**): Components related to a given failure and asset.

Failure Mode to Sensor (**FM2S**): Sensors capable of detecting given failure in a specific asset.

Sensor to Failure Mode (**S2FM**): List failure modes that can be detected by a given sensor for a specific asset.

For instance, the task Failure Mode to Sensor (**FM2S**) focuses on identifying appropriate sensors for detecting a given failure mode in a specific asset. We have provided an example prompt and expected response for the FM2S task in Figure 1, and representative examples for each task are included in Appendix Table 4.

Instruct: What sensors can be applied to detect a fault in an asset and its category?

Query: Asset: electric motor, Category: electric, Fault: stator windings fault

Asset description: Converts electrical energy into mechanical energy to power various industrial machinery.

Fault description: A Stator windings fault is a type of industrial failure mode where the electrical windings in the stator of an electric motor or generator become damaged or degraded, often due to overheating, insulation breakdown, or physical stress, leading to reduced performance, efficiency, or complete failure of the equipment.

Sensors: Current, Vibration, Temperature, Axial Flux, Cooling Gas

Input , LLM augmented , Ground truth

Figure 1: Example query with LLM augmentation and answer for the FM2S task.

3 Problem Formulation and Solution

In this section, we present our multi-task embedder for the industrial domain, with for modeling the the relationships between various entities such as assets, components, sensors, failure modes, and others. Our approach, as outlined in Figure 2, leverages tabular information, which are subsequently enriched using LLMs for improved query relevance and answer generation, to train an embedding model.

3.1 Problem Formulation

Given a set of tasks T, where each task $T_i \in T$ consists of a set of queries Q, and each query $q \in Q$ is associated with a set of relevant items I, the objective is to model the relationships between queries and their corresponding items. Each task involves queries grounded in various assets A, and semantically similar items may share similar relationships across tasks.

For example, consider a task T_i focused on identifying relevant sensors. Each query includes an asset from A (e.g., an *electric generator*) and a failure mode (e.g., *misalignment*). The relevant items which are drawn from the set I, are sensors (e.g., *vibration sensor*) capable of detecting that failure mode for the given asset.

The type of relevant items varies by task: depending on the context, items may represent *sensors*, *failure modes*, *components*, or *sub-components*.

During training, the embedding model observes queries from each task involving a subset of assets and their associated items. The goal is to generalize these relationships, enabling the system to infer relevant items for unseen queries involving different assets.

To this end, we learn a function f that maps both queries and items into a shared latent space $\mathbb{R}^d$, such that embeddings of relevant query-item pairs are positioned closer together. In inference, the system retrieves the top-k most relevant items for a given query based on proximity in this latent space.

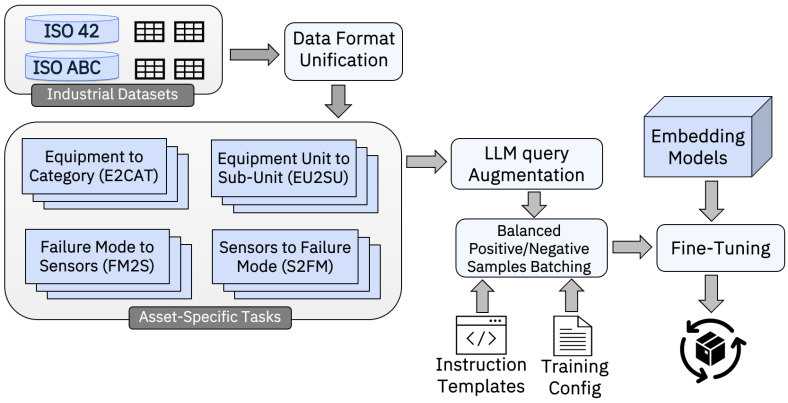

Figure 2: Overall data preparation and training flow for embedding models

3.2 Instruction Template

Given a relevant query-item pair (q^+, i^+) sampled from Table 1 in Apendix, we construct training examples using the following instruction-tuning template:

$$q^+_{\text{inst}} = \texttt{Instruct:}\ \{\textit{instruction}\}\ \texttt{.}\ \texttt{Query:}\ q^+,$$
$$o^+_{\text{inst}} = \{\textit{output-tag}\}\ :\ i^+.$$

{*instruction*} is a natural-language sentence describing the task, and q^+ is the query. For each task, we generate multiple paraphrased versions of the instruction using LLMs to promote generalization.

Figure 1 illustrates a complete prompt, showing the `Instruct` and `Query` fields for a task involving electric motors. Table 4 in Appendix provides one representative instruction and query pair per task.

3.3 LLM Query Augmentation

From the tabular data in the ISO documents, the information is not very descriptive to build a good model. For instance, for the sensor to failure mode task the query contains only the asset name, and its sensor. This makes it difficult to learn semantics for this asset and failure mode generalize on new assets. For this reason, we augment the query using an LLM with a one sentence description of its entities. We augment with a probability p, which acts a type of dropout to avoid overfitting. Figure 1 provides an example augmentation (light blue color). We provide further impacts of this design decision on the ablation studies in Section 5.1.

We prompt the LLM to augment the query with instructions shown in Appendix Table 5. The concise generated descriptions help encode semantic context and enable the embedding model to better capture task-relevant relationships.

3.4 Data Splitting

To make the experiment unbiased and more realistic, we split the train/validation/test set queries by assets for the tasks that is possible, otherwise we split randomly.

3.5 Query and Item Embeddings

Depending on the model used, we employ either the *mean pooling* or *last token pooling* strategy which are two common techniques for generating fixed-size embeddings from variable-length token sequences. A brief description is given in Appendix Section G.1.

3.6 Loss Function and Batching

To learn the embedding function f, we use a contrastive learning framework (Hadsell et al., 2006). Given a labeled triplet $\langle q, i, l \rangle$ where $l = 1$ indicates a relevant query-item pair and $l = 0$ indicates a negative pair, we minimize the margin loss:

$$\mathbb{L} = l \cdot d(q,i)^2 + (1-l) \cdot \max(\epsilon - d(q,i), 0)^2$$

Here, $d(q,i)$ denotes the Euclidean distance between the query and item embeddings, and ϵ is a margin parameter.

During training, for each task, we use all available positive (q^+, i^+) and negative (q^+, i^-) pairs. Since the number of negative pairs significantly exceeds the number of positive ones, we balance each training batch to include an equal number of positives and negatives. Additional analysis on this design choice is presented in the ablation study (Section 5.3).

We avoid loss functions that rely on in-batch negatives due to the high risk of false negatives caused by the relatively small item set and the fact that each query can be associated with multiple valid items (see Appendix Figure 10).

4 Experimental Setup

In this section we present the comparison before and after fine-tuning several embedding models on the 9 industrial tasks we prepared.

4.1 Embedding Models

We compare different models in retrieving the correct answer from a set of candidate items. The models used are: BERT (Kenton and Toutanova, 2019), MPNet (Song et al., 2020), BGE-large-v1.5 (Xiao et al., 2023), gte-Qwen2-7B-instruct (Li et al., 2023), and E5-Mistral-7B-Instruct (Wang et al., 2023) which have varying sizes. We also compare against BM25 retriever (Robertson et al., 2009), which is a bag-of-words retrieval function.

4.2 Training Settings

For all the models we use 4 A100 80GB GPUs with a training batch size of 32 per device. We do a full fine-tuning for BERT, MPNet, and BGE-large-v1.5, while for gte-Qwen2-7B-instruct and E5-Mistral-7B-Instruct we use LoRA (Hu et al., 2021) to fit them into the memory. We use mean pooling for all the models to generate the embeddings, apart from Qwen-7B for which we use last token pooling since it is how it was trained and has better results. We train a single model on all the different tasks for 3 epochs. We apply query augmentation as described in section 3.3 with 50% probability using Llama-3.1-70B-Instruct.

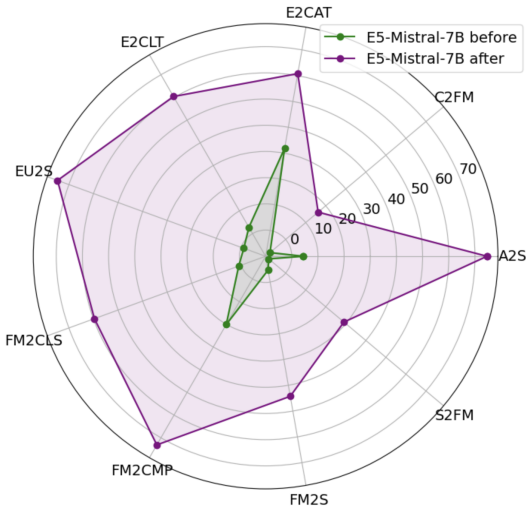

Figure 3: MAP@100 before and after finetuning E5 Mistral 7B for each task.

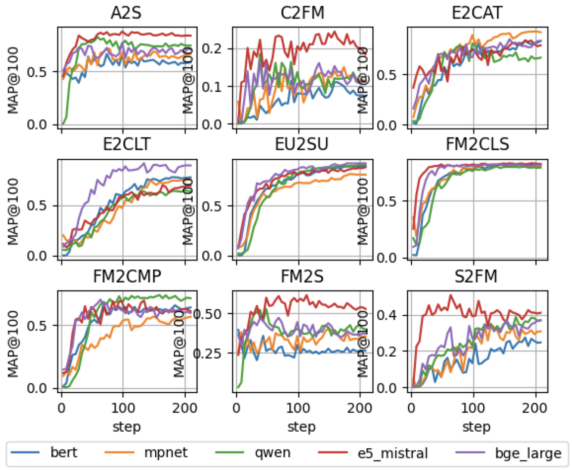

Figure 4: Validation performance during training for every task (MAP@100).

4.3 Fine-Tuning Results

Figure 3 provides a result using fine-tuned model for all the task. We observe consistent performance improvements across all tasks after fine-tuning. As shown in Figure 4, MAP@100 increases steadily on the validation set during training, demonstrating the effectiveness of our fine-tuning approach. The best-performing model varies by task, highlighting the diverse nature of the task set. Even prior to fine-tuning, the E5 Mistral model demonstrates strong zero-shot capabilities on several tasks, suggesting that it already captures some aspects of instruction semantics. A detailed breakdown of task-level performance before and after fine-tuning is provided in Appendix Table 6. More discussion on the generalizability in Appendix Section D.

5 Ablation study

5.1 Effects of LLM-generated description

We investigate what are the effects of adding LLM-generated description on the model's performance. We vary the probability of adding one sentence of LLM-generated description from 0 to 1 with 0.2 strides and see the effects on the test performance for each task (Figure 5). For this experiment we fixed the model to Mistral E5. For some tasks we can see performance boost (A2S, E2CLT, S2FM). It is notable that augmenting the query with a single sentence boosts performance when the probability of augmenting is not 0 or 1. This can be thought as some type of dropout. We believe that the rest of the tasks could also be benefited if the dataset was larger with more queries.

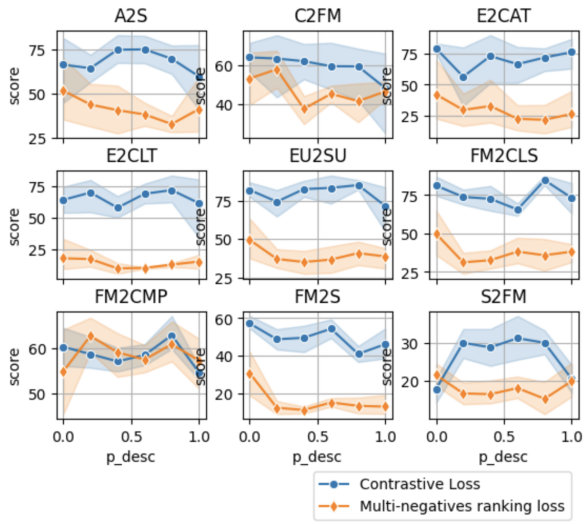

Figure 5: Loss comparison and effects of varying probability of augmenting the query using an LLM on the model's performance (MAP@100) per task.

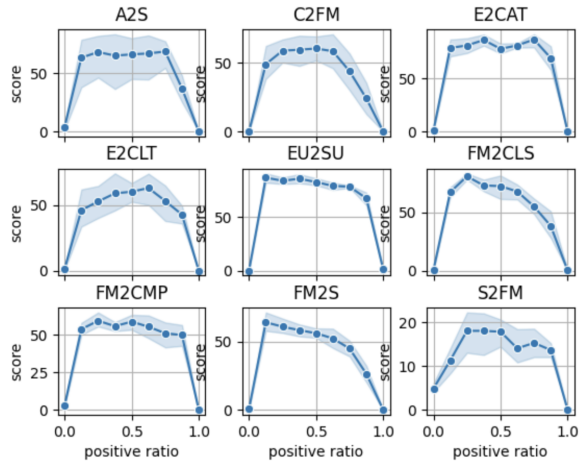

Figure 6: Effects of varying in-batch positives to negatives ratio per task (MAP@100).

5.2 Effects of loss function

We compare contrastive loss that we adopted in our system (Section 3.6) against multi-negatives ranking loss (Henderson et al., 2017). One big difference between the two losses, is that multi-negatives ranking loss uses in-batch negatives. The loss function is defined as:

$$\mathbb{L} = -\log \frac{\phi(q^+_{\text{inst}}, i^+)}{\phi(q^+_{\text{inst}}, i^+) + \sum_{n_i \in \mathbb{N}} \phi(q^+_{\text{inst}}, n_i)}$$

where $\mathbb{N}$ denotes the set of all in-batch negatives, and $\phi(\cdot)$ is the cosine similarity. For this loss function we only provide the query and positive examples, and for the k_{th} query q_k in the batch, all the positives from the rest of the in-batch samples are used as in-batch negatives i_l where $k \neq l$. Figure 6 performance of contrastive against multi-negatives ranking loss. The results indicate that the in-batch negatives hurt the performance due to the high chance of false negatives.

5.3 Effects of in-batch pos. to neg. samples ratio

Using Contrastive Loss, we vary the ratio of in-batch positives to negatives and study its impact. The batch size is fixed to 32, and the ratio is varied from 0 to 1 with 0.125 strides. The experiment is repeated 5 times with a different random seed. We chose to not apply any augmentation on the queries to avoid interference with this experiment. Figure 6 shows MAP@100 broken down by task for varying positives to negatives ratio. When the batch consists of only positive or only negative samples, the performance is very poor. Overall, a batch with balanced positives and negatives is performing the best.

5.4 Generalizability on a New Task

We evaluate the adaptability of our model fine-tuned on nine Industry 4.0 tasks by testing on a new task, namely Multiple-Choice Question Answering (MCQA). Using a RAG pipeline we assess performance on FAILURESENSORIQ (Constantinides et al., 2025), an MCQA dataset on asset failure modes and sensors. For each question, we retrieve the closest semantically option using the base model and our fine-tuned embedding models.

Answering accuracy on 2667 questions before/after finetuning is: BERT 57.48%/58.38%, MPNet-base-v2 58.46%/57.77%, and BGE-Large-v1.5 58.71%/58.31%.

Overall, there is no clear improvement on this task, and more work towards this direction is needed. This is a particularly challenging and highly ambitious experiment because we train the embedding models on highly structured prompts following a specific distribution (Example in Figure 1), and test it on a completely different task and format (MCQA).

6 Case Study

We present a real case study on how the agent interacts with our tools for a given user query. We do a qualitative analysis on the planning of the agent to solve industrial tasks given our industrial tools.

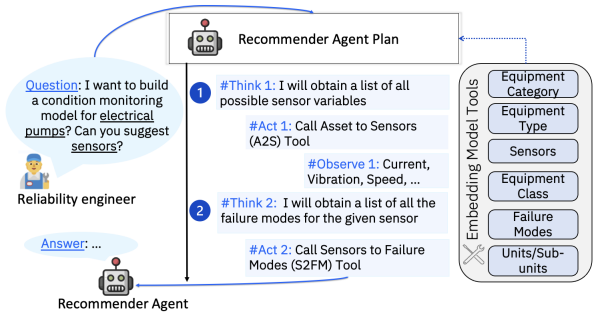

Figure 7: Integration of industrial tools with a reasoning agent.

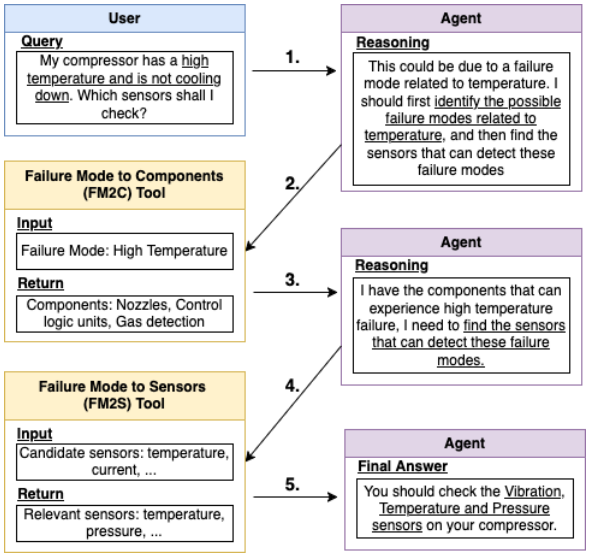

Figure 8: Interaction between user and agent.

6.1 Setup

We wrap our multi-task, domain-adapted, embedding models as langchain[1] tools, which accept different arguments. We provide a description for each tool, its arguments, what they return, the data types, and state which arguments are optional. In some of the tools we make calls to an LLM to get a one sentence description of the input entities (e.g. sensor, failure mode) which can help for more meaningful embeddings. We also provide an optional argument for candidate items in case the agent has some candidate items which it wants to pass to the tool to keep the most relevant. We provide as context 3 examples to the agent on how to call the tools to solve a simple question. We purposely use examples that are unrelated with each other, so that the agent can brainstorm on new ideas and not follow the same trajectory as the examples in context.

[1]https://www.langchain.com/

6.2 Interaction

Figure 8 presents the interaction between user, agent, and our designed industrial tools described in section 2. The reliability engineer is querying the system that he is **observing high temperature in their compressor**. Then the agent constructs a plan and invokes several of our tools to finally return the relevant sensors, along with generated information around the affected components.

1. The reliability engineer is observing high temperature in his compressor.

2. The agent invokes the Failure Mode to Component (FM2C) tool with the failure mode as argument to discover which components can experience this failure, and goes on and on.

6.3 Discussion

The agent seems to follow an interesting pattern. It first finds the components that can experience the given fault with tool calling. Then, given all the components, it provides some candidate sensors that it thinks are used to monitor these components (probably from prior knowledge) and asks the tool to narrow them down to only the relevant ones, getting 2 out of 3 correct in the final response. One thing that is notable is that none of these tools were given as in-context examples to the model, so these decisions from the model seem rather unbiased.

7 Conclusion

We show that fine-tuned embeddings improve performance on the nine defined tasks and generalize well to questions about related industrial assets, achieving comparable results on new unseen assets. However, the generalizability on the unseen FailureSensorIQ MCQA task still needs improvement and we leave this as a future work. We finally present a case study where we deploy the model as a tool within an agentic system.

8 Limitations

We currently fix k when retrieving top-k relevant items, which may not match the actual number of relevant results. In the future, we plan to adopt dynamic thresholds (e.g., distance-based or cross-encoder filtering) to improve precision and recall.

Another limitation is that the LLM-generated entity descriptions (Section 3.3) can hallucinate details. We have made some analysis in section I. Future work will automate this process. We plan

to explore attention maps (Rateike et al., 2023; Sriramanan et al., 2024), activation analysis (Yehuda et al., 2024), and perplexity-based methods (Sriramanan et al., 2024) to better understand and reduce hallucinations and their impact on embedding quality.

References

Peter Anderson, Mano Vikash Janardhanan, Jason He, Wei Cheng, and Charlie Flanagan. 2024. Greenback bears and fiscal hawks: Finance is a jungle and text embeddings must adapt. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 362–370, Miami, Florida, US. Association for Computational Linguistics.

Christodoulos Constantinides, Dhaval Patel, Shuxin Lin, Claudio Guerrero, Sunil Dagajirao Patil, and Jayant Kalagnanam. 2025. Failuresensoriq: A multi-choice qa dataset for understanding sensor relationships and failure modes. *arXiv preprint arXiv:2506.03278*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *Preprint*, arXiv:2303.14524.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

ISO. 2016. Iso 14224:2016 - petroleum, petrochemical and natural gas industries — collection and exchange of reliability and maintenance data for equipment. Last reviewed and confirmed in 2022; remains current.

ISO. 2018. Condition monitoring and diagnostics of machines — general guidelines. Geneva, Switzerland. International Organization for Standardization (ISO). This publication was last reviewed and confirmed in 2023. Therefore, this version remains current.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1. Minneapolis, Minnesota.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. 2024. Mdagents: An adaptive collaboration of llms for medical decision-making. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

LangChain. 2024a. Langchain arxiv tool integration. `https://python.langchain.com/docs/integrations/tools/arxiv/`. Accessed: 2025-06-07.

LangChain. 2024b. Langchain wikipedia tool integration. `https://python.langchain.com/docs/integrations/tools/wikipedia/`. Accessed: 2025-06-07.

Yuxuan Lei, Jianxun Lian, Jing Yao, Mingqi Wu, Defu Lian, and Xing Xie. 2024. Aligning language models for versatile text-based item retrieval. In *Companion Proceedings of the ACM Web Conference 2024*, pages 935–938.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024. Making text embedders few-shot learners. *Preprint*, arXiv:2409.15700.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Miriam Rateike, Celia Cintas, John Wamburu, Tanya Akumu, and Skyler Speakman. 2023. Weakly supervised detection of hallucinations in llm activations. *arXiv preprint arXiv:2312.02798*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. Llm-check: Investigating detection of hallucinations in large language models. *Advances in Neural Information Processing Systems*, 37:34188–34216.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. *Preprint*, arXiv:2401.00368.

Orion Weller, Benjamin Van Durme, Dawn Lawrie, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. 2024. Promptriever: Instruction-trained retrievers can be prompted like language models. *Preprint*, arXiv:2409.11136.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Ran Xu, Wenqi Shi, Yue Yu, Yuchen Zhuang, Yanqiao Zhu, May Dongmei Wang, Joyce C. Ho, Chao Zhang, and Carl Yang. 2024. BMRetriever: Tuning large language models as better biomedical text retrievers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22234–22254, Miami, Florida, USA. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yakir Yehuda, Itzik Malkiel, Oren Barkan, Jonathan Weill, Royi Ronen, and Noam Koenigstein. 2024. Interrogatellm: Zero-resource hallucination detection in llm-generated answers. *arXiv preprint arXiv:2403.02889*.

A Industrial relational graph example

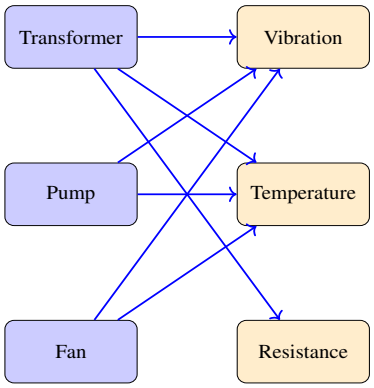

Figure 9: Bipartite graph showing relationship "is monitored" between assets and sensors.

B In-batch negatives is prone to false negatives

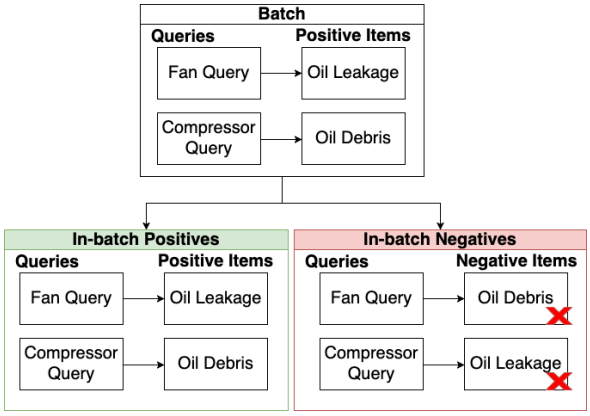

Figure 10: An example of how in-batch negatives is prone to false negatives. Both Fan and Compressor are related to both Oil Leakage and Oil Debris. In the selected batch, the positive items will be used as negatives and this will result in in-batch false negatives. In our dataset there is a high chance of this happening, since for each query there is a high number of related items when compared to the total unique items.

C Task Distribution

From a quantitative perspective, Table 1 presents a distribution across various tasks, focusing on the number of queries, the number of items, and the average number of related items per query. There is significant variability in both the number of items and the average related items per query across tasks, indicating diverse levels of complexity in task structure.

Task	Query Count	Item Count	Avg. Items per Query
A2S	10	53	12.6
C2FM	44	6	1.0
E2CAT	10	107	10.7
E2CLT	42	156	4.5
EU2SU	43	1191	33.1
FM2CLS	140	62	1.0
FM2CMP	254	44	2.7
FM2S	111	53	4.5
S2FM	485	55	1.0

Table 1: Distribution of raw data points across tasks.

D Out-Of-Distribution Asset Generalizability

As mentioned in the Data Splitting Section 3.4, for the tasks applicable we split the dataset into train/validation/test sets stratified by assets, which represent a diverse set of industries. The assets are: Electric Motor, Steam Turbine, Aero Gas Turbine, Compressor, Power Transformer, Fan, Reciprocating Internal Combustion Engine, Industrial Gas Turbine, Electric Generator. For instance, power transformers have use in energy transmission and utilities, whereas Compressors are used in Oil & Gas. This can be thought as an Out-Of-Distribution (OOD) experiment. Even though these assets come from different domains, they still share some similar characteristics (components/failure modes/sensors) which can explain why there is improvement when training/testing between different assets. Other works have explored the generalizability of LLMs on OOD tasks (Lei et al., 2024).

E LLM Augmentation Costs

With today's service-oriented LLMs, costs are usually calculated in terms of the number of tokens generated. In our case, for each prompt, we generate a concise one-sentence description for each entity, which would yield around 20-30 tokens per prompt. Example generated descriptions can be found in Appendix 4.

F Error Case Analysis

Entity	Name	Frequency
Sensor	Vibration	36
Failure	Bearing Wear	34
Component	Electrical Submersible Pump	14
Failure Class	Abnormal Instrument Reading (AIR)	4
Equipment Class	Snubbing, surface well control equipment	2
Equipment Type	Centrifugal	2

Table 2: Most frequent **False Negative** retrieved items per task/entity.

Entity	Name	Frequency
Equipment Type	Rigid	30
Component	Wiring	24
Sensor	Compressor Temperature	16
Equipment Class	Snubbing, Surface Equipment	6
Failure	Damaged Labyrinth	7
Failure Class	Spurious Operation (SPO)	4

Table 3: Most frequent **False Positive** retrieved items per task/entity.

G Experiments Details and Results

In this section, we present details for each task including example queries, LLM augmentation, and answer. Table 4 presents representative examples for each task type covered in our framework. For each task, we show the input query, the augmentation applied by the LLM (if any), and the corresponding answer. Tasks span a range of reliability and maintenance reasoning types—from sensor selection (A2S, FM2S) to failure mode identification (C2FM, FM2CMP, FM2CLS) and equipment/component mapping (E2CAT, E2CLT, EU2SU). Where applicable, LLMs are prompted to enhance the query context with domain-specific descriptions to improve relevance and answer quality. In some cases (marked with an asterisk), no augmentation is applied as the original query contained sufficient detail.

Next, we provide $MAP@100$ retrieval performance before and after fine-tuning each model (Figure 11). Table 6 compares retrieval performance across several industrial domain tasks before and after fine-tuning for a range of embedding models, including BM25, BERT, MPNet, BGE, Qwen2, and E5-Mistral. Metrics reported include HIT@1, MAP@100, and NDCG@10. Across all tasks, fine-tuning significantly boosts performance, especially for HIT@1, where models like E5-Mistral-7B and MPNet-base-v2 often outperform others. Notably, E5-Mistral-7B consistently achieves the highest post-tuning scores across most tasks, indicating strong alignment between embedding quality and domain-specific retrieval needs.

Traditional lexical methods like BM25 perform poorly, especially on HIT@1, highlighting the limitations of non-semantic approaches in complex technical domains. Fine-tuned dense retrievers (e.g., MPNet and BGE) show marked improvements. Even though smaller models like BERT (100M) still lag behind, they still have a decent performance, making the system suitable to be deployed on the edge. These results emphasize the importance of model scale, architecture, and task-aware fine-tuning in enhancing semantic retrieval for industrial applications.

G.1 Pooling methods

Last Token Pooling. Given a relevant query q^+ and item i^+, we concatenate them with an `[EOS]` token. The resulting sequence is passed through the transformer model f, and we extract the embedding corresponding to the final `[EOS]` token from the last hidden layer.

Mean Pooling. Let $T_1, \ldots, T_n$ be the tokens of the instruction-formatted query q^+_{inst}. These tokens are input to the model f, and we compute the average of the last-layer activations to obtain the final embedding. The same procedure is applied to the instruction-formatted item i^+_{inst}.

Task	Example question	Example augmentation	Example answer
A2S	Instruct: What sensors are relevant for the given asset and its category? Query: Asset: aero gas turbine, Category: rotating	Asset description: A rotary machine that extracts energy from steam and converts it into mechanical work	Sensor: amps
C2FM	Instruct: Given an asset component, what are the possible failure modes it could experience? Query: Component: Turbo-expanders	Component Description: Turbo-expanders are industrial components that convert the pressure energy of a high-pressure gas into mechanical energy, often used in power generation, refrigeration, and other applications where gas expansion can be harnessed to drive turbines or other machinery.	Failure mode: Failure to set/retrieve (SET)
E2CAT	Instruct: In the context of a given equipment category, what equipment is the most relevant? Query: Equipment category: Electrical	Equipment Category Description: The Electrical equipment category includes a wide range of devices and systems that generate, transmit, distribute, and utilize electrical energy, such as generators, transformers, circuit breakers, and lighting fixtures	Equipment: Coiled tubing, work strings
E2CLT	Instruct: For a given equipment class, which types of equipment are most essential? Query: Equipment Class: Swivels	Equipment Class Description: Swivels are industrial components that allow for rotational movement, enabling hoses, pipes, or other equipment to pivot freely while maintaining a secure connection.	Equipment Type: Toxic gases
EU2SU	Instruct: What components and their groups are part of a specific equipment unit? Query: Equipment unit: Subsea pipelines	(No augmentation was done*)	Component group: Mounting assembly, Component name: Mounting connector
FM2CMP	Instruct: Given the asset name and failure mode class, what components are involved? Query: Asset name: well completion, failure mode class: Low output	(No augmentation was done*)	Component: Top drives
FM2CLS	Instruct: For the given failure description, which failure mode class applies? Query: Failure Description: Failed set/retrieve operations	(No augmentation was done*)	Failure class: Power/signal transmission failure
FM2S	Instruct: What sensors can be applied to detect a fault in an asset and its category? Query: Asset: electric motor, Category: electric, Fault: stator windings fault	Asset description: Converts electrical energy into mechanical energy to power various industrial machinery. Fault description: A Stator windings fault is a type of industrial failure mode where the electrical windings in the stator of an electric motor or generator become damaged or degraded, often due to overheating, insulation breakdown, or physical stress, leading to reduced performance, efficiency, or complete failure of the equipment.	Sensor: output power
S2FM	Instruct: What failure modes can be detected by reading a sensor in an asset and its category? Query: Asset: electric motor, Category: electric, Sensor: current	Asset description: Converts electrical energy into mechanical energy to power various industrial machinery. Sensor Description: Sensor that measures electrical current in various systems to detect anomalies and prevent overloads or system failures	Failure mode: Rotor Windings Fault

Table 4: Example queries, LLM augmentation, and answers for each task. No augmentation was done for queries that we determined that has already enough information to be answered and the performance was already good.

`Provide a one sentence description for the equipment category.`
`Provide a one sentence description for the equipment type.`
`Provide a one sentence description for the industrial component.`
`Provide a one sentence description for the industrial failure mode.`

Table 5: Examples of instruction prompts for LLM query augmentation

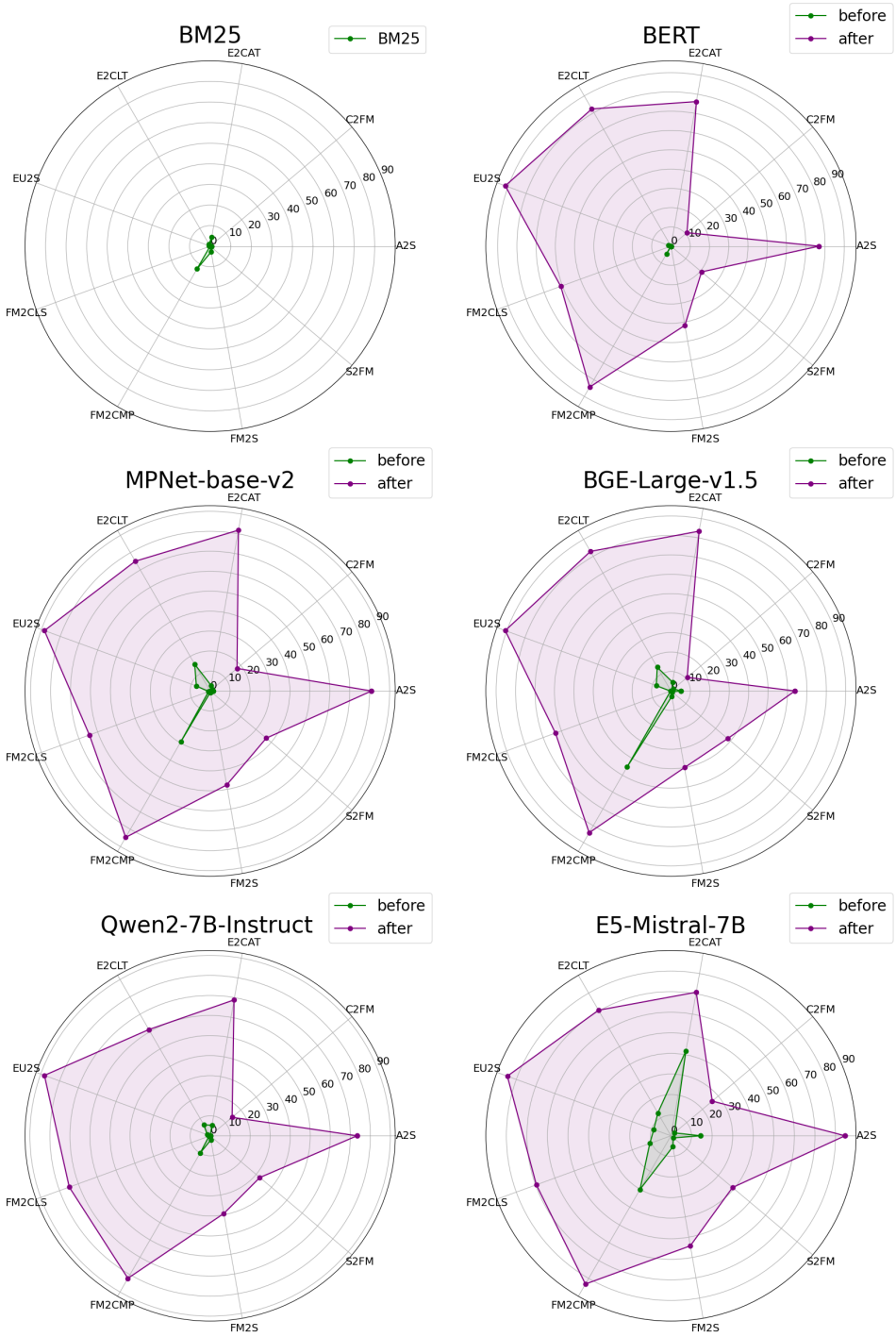

Figure 11: MAP@100 retrieval performance before and after fine-tuning each model and task.

Task	Model	Size	HIT@1		MAP@100		NDCG@10	
			before	after	before	after	before	after
Asset to Sensor (A2S)	BM25	-	0.00		0.76		2.42	
	BERT	110M	0.00	80.7	0.00	76.87	0.00	80.66
	MPNet-base-v2	110M	5.26	**91.23**	1.69	80.94	2.47	85.19
	BGE-Large-v1.5	335M	0.00	84.21	5.20	63.81	2.77	69.05
	Qwen2-7B-Instruct	7B	0.00	84.21	0.16	73.40	0.00	78.27
	E5-Mistral-7B	7B	**15.79**	**91.23**	**14.43**	**84.62**	**18.46**	**87.03**
Component to Failure Mode (C2FM)	BM25	-	0.00		0.30		0.73	
	BERT	110M	0.00	4.22	0.00	10.89	0.00	13.46
	MPNet-base-v2	110M	0.00	9.64	1.48	17.6	**1.87**	21.54
	BGE-Large-v1.5	335M	0.00	3.61	1.71	10.91	0.95	13.96
	Qwen2-7B-Instruct	7B	0.00	8.43	0.02	14.33	0.00	18.79
	E5-Mistral-7B	7B	0.00	**9.64**	**2.18**	**26.20**	1.83	**32.50**
Equipment to Category (E2CAT)	BM25	-	14.29		4.61		12.53	
	BERT	110M	0.00	77.78	0.00	76.17	0.00	86.78
	MPNet-base-v2	110M	0.00	**100.0**	2.82	81.83	6.99	88.95
	BGE-Large-v1.5	335M	22.22	88.89	4.59	**83.46**	9.80	**90.90**
	Qwen2-7B-Instruct	7B	0.00	88.89	5.37	68.82	3.56	74.38
	E5-Mistral-7B	7B	**44.44**	77.78	**41.87**	70.80	**54.70**	83.61
Equipment to Class Type (E2CLT)	BM25	-	0.00		1.08		00.00	
	BERT	110M	0.00	**85.71**	0.00	82.22	0.00	83.30
	MPNet-base-v2	110M	2.38	73.81	**15.48**	75.07	**18.83**	78.39
	BGE-Large-v1.5	335M	**7.14**	**85.71**	14.12	**82.89**	16.2	**83.99**
	Qwen2-7B-Instruct	7B	0.00	66.67	6.33	61.19	9.25	64.54
	E5-Mistral-7B	7B	0.00	73.81	12.76	70.41	16.43	69.61
Equipment Unit to Subunit (EU2SU)	BM25	-	0.00		0.53		0.55	
	BERT	110M	0.00	**100.0**	1.35	**91.56**	1.53	**95.91**
	MPNet-base-v2	110M	**14.63**	**100.0**	7.34	88.43	14.95	92.84
	BGE-Large-v1.5	335M	7.32	92.68	8.03	90.79	**16.24**	91.43
	Qwen2-7B-Instruct	7B	0.00	95.12	1.52	88.06	4.15	93.94
	E5-Mistral-7B	7B	0.00	92.68	**8.88**	84.63	13.85	88.74
Failure Mode to Components (FM2CMP)	BM25	-	0.00		0.00		0.00	
	BERT	110M	0.00	48.42	0.00	60.89	0.00	69.04
	MPNet-base-v2	110M	0.00	61.99	0.86	64.43	0.93	70.40
	BGE-Large-v1.5	335M	0.00	54.75	0.23	63.28	0.00	69.82
	Qwen2-7B-Instruct	7B	0.00	**70.59**	0.57	**74.86**	0.94	**78.51**
	E5-Mistral-7B	7B	**4.07**	59.73	**10.94**	69.62	**13.69**	74.26
Failure Mode to Class (FM2CLS)	BM25	-	8.14		12.78		14.55	
	BERT	110M	3.23	82.26	4.62	84.41	4.57	84.97
	MPNet-base-v2	110M	20.43	**82.8**	29.31	**84.68**	31.94	**85.17**
	BGE-Large-v1.5	335M	**34.41**	81.72	**45.07**	84.20	**50.04**	84.85
	Qwen2-7B-Instruct	7B	2.15	77.96	9.96	82.13	12.69	83.18
	E5-Mistral-7B	7B	5.91	**82.8**	30.08	83.14	39.3	83.98
Failure Mode to Sensor (FM2S)	BM25	-	0.00		2.84		4.45	
	BERT	110M	0.00	38.38	0.00	41.71	0.00	50.70
	MPNet-base-v2	110M	0.00	**49.49**	0.70	47.81	0.63	59.48
	BGE-Large-v1.5	335M	0.00	48.48	2.77	40.03	2.77	48.45
	Qwen2-7B-Instruct	7B	0.00	42.42	2.10	39.39	4.10	46.87
	E5-Mistral-7B	7B	**3.03**	48.48	**5.38**	**54.27**	**7.73**	**61.69**
Sensor to Failure Modes (S2FM)	BM25	-	00.00		0.46		0.00	
	BERT	110M	0.00	11.48	0.00	20.76	0.00	25.04
	MPNet-base-v2	110M	0.00	19.34	0.03	36.76	0.00	**44.36**
	BGE-Large-v1.5	335M	0.00	27.54	0.66	38.12	0.85	42.52
	Qwen2-7B-Instruct	7B	0.00	20.33	0.36	32.41	0.27	37.32
	E5-Mistral-7B	7B	0.00	**33.77**	**1.57**	**39.07**	**1.55**	43.81

Table 6: Retrieval performance before and after fine-tuning. For tasks with a high average number of related items per query, the HIT score is naturally higher.

H Towards a more Robust Baseline Domain Embedder from Web Data

The pre-trained embedders didn't show a satisfactory performance (Table 6). In our effort to build a more robust baseline, we use an agent-driven domain web data collection approach and fine-tune using Masked Language Modeling (MLM).

H.1 Dataset Preparation

We set up a ReAct agent (Yao et al., 2022) equipped with two tools that connect to external knowledge sources: ArXiv (LangChain, 2024a) and Wikipedia (LangChain, 2024b). We then use a series of multiple-choice industrial questions from the `FailureSensorIQ` dataset (Constantinides et al., 2025) and employ the ReAct agent to generate answers by leveraging information retrieved from these external knowledge bases. During execution, all interactions between the ReAct agent and the external tools are stored as key-value pairs. Table 7 provides an example of the key-value store for Wikipedia, focusing on a particular key named "Rotor windings fault in electric motors". We treat the values as unique passages for fine-tuning an embedding model tailored to our domain. Overall, we collected 10552 passages from Wikipedia and 11515 passages from ArXiv.

H.2 Model Training and Experimental Results

We train different models using Masked Language Modeling (MLM) on the collected passages and report the results in Table 8. We train for 100 epochs on a machine with 1 Nvidia A100 (80GB), using a batch size of 32 and learning rate of $2 * 10^{-5}$, weight decay of 0.01, and token masking probability of 15%. Overall, the performance is still unsatisfactory. The key takeaways are: (a) carefully curated datasets with domain-specific instructions based on day-to-day operations are crucial for learning good embeddings, and (b) publicly available web documents lack sufficient details on the relationships necessary to model interactions between different industrial entities (e.g., sensors, failure modes, components).

I Hallucination Analysis on the LLM Augmentation

As discussed in Section 3.3, our approach leverages an LLM to augment queries, making it crucial to assess the quality of the generated text by LLM.

Motor Type (Wikipedia)	**Rotor Windings Fault Relevance Summary**
Reluctance Motor (Reluctance motor)	Rotor does **not have windings**; torque is generated via magnetic reluctance. **Not relevant** to rotor winding faults.
Brushed DC Motor (Brushed DC electric motor)	Rotor includes windings and uses brushes for commutation. Wear and tear may impact windings. **Relevant** to rotor winding fault scenarios.
Doubly Fed Induction Generator (Doubly fed electric machine)	Rotor has **field windings** connected to external circuits; faults in rotor windings can affect performance. **Highly relevant** to rotor winding fault analysis.

Table 7: Summary of rotor windings fault relevance for different electric motor types from Wikipedia pages

We extract all the LLM generated text to conduct deeper study. All together, the dataset includes 255 unique entities spanning assets, sensors, failure modes, equipment components, categories, class types, subunits, and units and we have 225 summary in a form of single sentence is extracted. We analyze the distribution of token lengths and perplexity scores for entity descriptions generated using LLaMA-3.3-70B-Instruct (Figures 12 and 13). The token counts follow an approximately bimodal normal distribution, while perplexity exhibits a heavy right-tailed distribution.

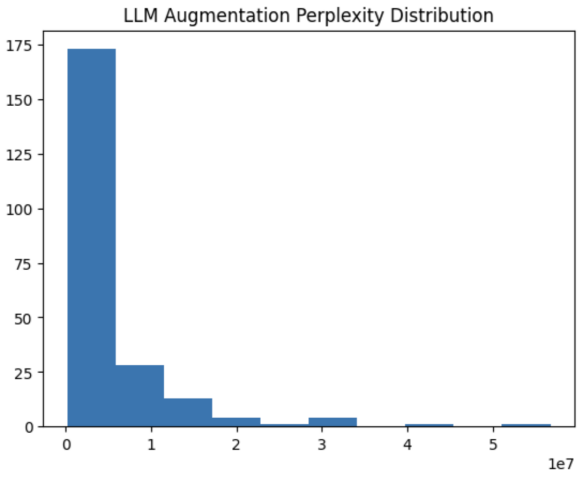

Figure 12: Perplexity Distribution of the augmented entity descriptions using Llama-3.3-70B-Instruct.

Task	Model	ACC@1		MAP@100		NDCG@10	
		Before	After	Before	After	Before	After
Asset to Sensor (A2S)	BERT	0.00	0.00	0.00	0.00	0.00	0.00
	MPNET	0.00	0.00	**1.51**	0.00	**0.71**	0.00
	BGE	0.00	0.00	**6.36**	0.00	0.00	0.00
Component to Failure Mode (C2FM)	BERT	0.00	0.00	0.00	**0.16**	0.00	**1.49**
	MPNET	0.00	0.00	0.00	0.00	0.00	0.00
	BGE	0.00	0.00	**0.31**	0.00	0.00	0.00
Equipment to Category (E2CAT)	BERT	0.00	**16.67**	1.38	**3.18**	3.52	**9.32**
	MPNET	0.00	0.00	0.41	**2.13**	5.77	**6.22**
	BGE	**16.7**	0.00	**6.80**	0.75	**12.2**	0.00
Equipment to Class Type (E2CLT)	BERT	**4.88**	0.00	1.74	**1.99**	**3.67**	1.14
	MPNET	0.00	0.00	7.13	**12.5**	**11.80**	4.58
	BGE	2.44	**19.5**	4.88	**6.86**	4.99	**9.43**
Equipment Unit to Subunit (EU2SU)	BERT	2.63	**10.53**	1.97	**4.96**	5.67	**13.57**
	MPNET	**15.79**	0.00	**8.92**	4.58	**22.68**	12.47
	BGE	18.4	**31.58**	10.96	**11.94**	24.11	**31.18**
Failure Mode to Components (FM2CMP)	BERT	0.00	0.00	**0.28**	0.00	**0.31**	0.00
	MPNET	**0.45**	0.00	**2.70**	0.27	**3.34**	0.53
	BGE	0.00	0.00	**0.92**	0.00	**10.57**	0.00
Failure Mode to Class (FM2CLS)	BERT	0.00	**5.08**	1.15	**12.7**	0.60	**16.5**
	MPNET	**20.34**	20.34	30.80	**33.37**	34.68	**37.52**
	BGE	**29.66**	28.81	**37.85**	35.85	**41.31**	38.22
Failure Mode to Sensor (FM2S)	BERT	0.00	0.00	0.00	**2.1**	0.00	**0.8**
	MPNET	0.00	0.00	0.23	**0.83**	0.00	**0.13**
	BGE	0.00	0.00	**4.25**	0.00	**2.27**	0.00
Sensor to Failure Modes (S2FM)	BERT	0.00	**2.13**	0.82	**3.46**	0.78	**3.31**
	MPNET	**2.13**	0.00	**0.36**	0.00	**0.66**	0.00
	BGE	**0.71**	0.00	**3.65**	0.00	**4.05**	0.00

Table 8: Performance before and after Masked Language Modeling (MLM) on industrial web documents. Overall, performance remains poor and improvements are inconsistent. This underscores the importance of carefully curating instruction-based datasets tailored to specific industrial tasks, as demonstrated in our methodology.

To further evaluate quality, we manually inspect generated descriptions in the top 5% percentile of perplexity scores for factual accuracy (Example descriptions in Table 4 as an example). Although these prompts are found to be factually correct, token count and perplexity may still serve as useful signals for identifying potential hallucinations and warrant further investigation.

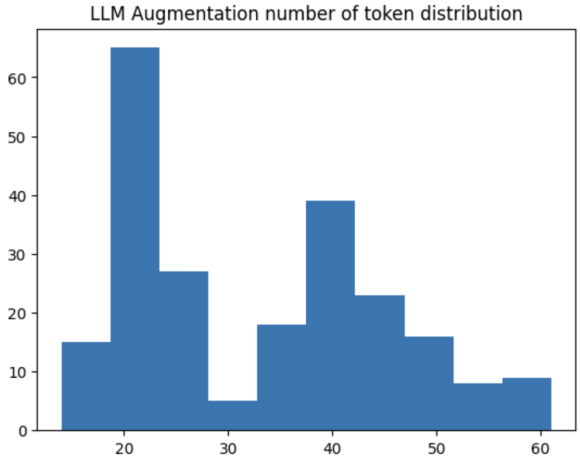

Figure 13: Number of Tokens Distribution with the augmented entity descriptions using Llama-3.3-70B-Instruct.

Metric	Value
Total Evaluations	222
Mean Score	0.874
Median Score	0.800
Standard Deviation	0.101
Minimum Score	0.5
Maximum Score	1.0
Distribution (Score Range → Count)	
0.0 – 0.2	0
0.2 – 0.4	0
0.4 – 0.6	2
0.6 – 0.8	5
0.8 – 1.0	215

Table 9: Summary Statistics and Distribution of Groundedness Scores

I.1 FactChecker Agent

To validate the factual consistency of textual summary, we develop a ReAct-based fact-checking agent that performs retrieval-augmented verification. For each summary, the agent queries authoritative sources such as Wikipedia and arXiv to gather evidence that either supports or contradicts the claim. As specified in Table 14, the protocol enables fine-grained evaluation by aggregating evidence across steps, assigning confidence scores, and detecting contradictions. This dual-source validation strategy culminates in a groundedness score for the overall rationale, offering a robust assessment of factuality in generated explanations.

We evaluate the factual grounding of 222 agent-generated summary using a groundedness scoring metric ranging from 0.0 (no support) to 1.0 (strong support). Table 9 show the outcome of our experiment. The analysis reveals a high overall factual accuracy, with a mean score of 0.874 and a median of 0.800, indicating that most rationales are well-supported by reliable technical sources. The standard deviation of 0.101 reflects moderate variability, while the minimum and maximum scores were 0.5 and 1.0, respectively. Distribution analysis shows that the vast majority (215 out of 222) of scores fall within the 0.8 to 1.0 range, confirming strong evidence backing the agents' conclusions. Only a small fraction exhibits weaker grounding, highlighting areas for potential improvement.

Task: Assess whether the following step-by-step rationale about an industrial asset and its associated sensor-based failure mode identification is **factually grounded** in reliable technical or scientific sources.
Your role is that of a Reliability Engineering Expert specializing in sensor-based condition monitoring and failure diagnostics across a wide range of industrial assets (e.g., turbines, pumps, motors, HVAC systems, rotating machinery, etc.).
Context:
- Asset Type: "{asset_type}"
- Sensor Type: "{sensor_type}"
- Rationale: "{rationale_text}"
Instructions:
1. Search reliable sources such as **Wikipedia**, **arXiv**, and other authoritative engineering or maintenance references to find passages that either **support** or **contradict** the claims made in each step of the rationale.
2. For each reasoning step, assess whether the technical claim is:
- correct: factually grounded and logically sound.
- partially_correct: partially grounded but includes gaps or weak assumptions.
- incorrect: not supported or contradicted by reliable sources.
3. For each step:
- Provide a **confidence score** (0.0–1.0) reflecting your certainty.
- Provide a **brief comment** justifying your assessment.
- Include **any relevant supporting or contradicting passages** you find from external sources with **citations**.
... 5. Assign a **groundedness score** to the entire rationale, from 0.0 (no support or contradicted) to 1.0 (strongly supported).
6. You must conclude with a Finish action that returns a fully filled, valid, and parseable **JSON object** matching the exact structure below.
- Do **not** use placeholders.
- The process is **not complete** unless a proper JSON is returned.
Output Format (JSON):

```
{
"asset_type": "{asset_type}",
"sensor_type": "{sensor_type}",
"rationale": "{rationale_text}",
"evaluation": [
{
"step": 1,
"status": "correct" | "partially_correct" | "incorrect",
"comment": "...",
],
"contradicting_passages": [
.....suppressed....       "justification": "...",
"groundedness_score": 0.0
}
}
```

Figure 14: Sensor-Groundedness Evaluation Prompt