Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P1487-cd

Large Language Models for Extracting Failed Components from Consumer Electronics Feedback

Jean Meunier-Pion
Laboratoire de Génie Industriel, CentraleSupélec, Université Paris-Saclay, France.
E-mail: jean.meunier-pion@centralesupelec.fr

Zhiguo Zeng
Laboratoire de Génie Industriel, CentraleSupélec, Université Paris-Saclay, France.
E-mail: zhiguo.zeng@centralesupelec.fr

Anne Barros
Laboratoire de Génie Industriel, CentraleSupélec, Université Paris-Saclay, France.
E-mail: anne.barros@centralesupelec.fr

Large Language Models (LLMs) have demonstrated over the past few years a strong capability in natural language understanding, opening new opportunities in reliability analysis based on text data. In the meantime, customer review data offer valuable insights into system failures, but the unstructured nature of natural language makes failure information extraction challenging. In this study, we address the problem of failed component extraction from customer reviews of tablet computers, aiming to detect failures at a component level to assess both system and component reliability. We propose a novel approach using LLMs for this task and frame it as a multi-label classification problem. Our method combines the design of a prompting strategy with the use of pre-trained lightweight LLMs to automatically extract the desired information. We conduct a comparative evaluation of state-of-the-art non-proprietary LLMs on this task. To support this work, we introduce a newly annotated dataset of 1,215 customer reviews, of which 356 mention at least one failure, annotated specifically for component failure detection. This fine-grained failure detection framework aims at enabling more accurate reliability assessments by pinpointing individual component failures within the broader system context. Our preliminary results show the potential of LLMs to leverage unstructured textual data for component-level reliability analysis. Code and data available here: `https://github.com/jmpion/FaCET-ESREL2025`

Keywords: Large Language Models, Failed Components, System Reliability, Customer Feedback, Natural Language Processing, Multilabel Classification, Consumer Electronics.

1. Introduction

Reliability is defined as the probability that a product performs its intended function without failure, under specified conditions, for a specified period of time in Yang (2007). To assess system reliability, failure information is thus necessary. In general, such failure information is available in the form of historical records or lifetime testing data.

However, with the quick rising of advanced **N**atural **L**anguage **P**rocessing (**NLP**) methods, the possibility of extracting failure information from text data offers a unique opportunity to leverage more failure-related data for reliability analysis. Previous studies have explored the usage of NLP for reliability-related tasks, such as Zhang et al. (2020) which aimed at studying maintenance work order records to classify among several failure codes and handle multi-class classification, under class imbalance. In Arif-Uz-Zaman et al. (2017), maintenance work orders were used along downtime data to classify unplanned events, in order to provide more accurate failure dates. In general, several such studies have been conducted on industrial proprietary text data and using traditional frequentist approaches to derive failure information from maintenance textual data. Other works (Meunier-Pion et al. (2021)) explored the possibility to classify reviews mentioning product failures. Nevertheless, the limitation of such

studies is that the information mined from text is limited, and does not include crucial knowledge such as time-to-failure, failure modes, failure causes, nor the failed components of the malfunctioning system. However, in the domain of consumer electronics, such as tablet computers, where systems are made of complex interactions of many subsystems and components, understanding system failures at the component level is essential for correctly assessing the system reliability. This can then guide design improvements, enhancing customer satisfaction and limiting costs due to maintenance and warranty claims. Thus, while prior research has shown the feasibility of detecting failures through customer feedback, a crucial gap remains: identifying specific components associated with these failures. The basic idea can be summarized with the following question: what component fails when there is a failure? E.g., if a tablet computer fails, is it due to the screen, to the battery, to the motherboard, etc.? Addressing this gap not only enables better reliability modeling but also provides actionable insights for targeted interventions.

To address this research question, this study introduces a novel task, **Fa**iled **C**omponents **E**xtraction from **T**ext (**FaCET**), whose goal is the extraction of failed components from unstructured text data. The task leverages advances in NLP to move beyond failure detection and towards a deeper understanding of how failures manifest in complex systems. A new dataset, **CuReFaCET** (**Cu**stomer **Re**views for **Fa**iled **C**omponent **E**xtraction from **T**ext), was developed to specifically support this task, by providing 1,215 customer reviews labeled across 60 components for failure detection at the component level. The targeted contribution of this dataset is both to address the FaCET task, while bringing a public text dataset for reliability engineering, in a field which lacks large annotated public text datasets.

Given the scarcity of public labeled data in this domain, traditional supervised learning methods are limited in their applicability. To address this challenge, we explored the potential of **L**arge **L**anguage **M**odel**s** (**LLMs**), which have demonstrated state-of-the-art performance across various NLP tasks (OpenAI (2024)). LLMs are particularly well-suited for this task due to their ability to perform very well in zero-shot scenarios.

2. Framework

2.1. *Task description: FaCET*

The FaCET task aims at extracting failed components from text data. Formally, we can formulate the FaCET task as follows (see Figure 1 as well). We are given an input text t. In this text, we are interested in product $\mathcal{P}$. Let $\mathcal{C}_\mathcal{P}$ be the set of components of product $\mathcal{P}$. For instance, in our case study we work on tablet computers, and a product can be $\mathcal{P} =$ "Tablet A". Then, the set of components can be $\mathcal{C}_\mathcal{P} = \{$"screen", "battery", "motherboard"$\}$. Of course, this serves as an example, and there will be more components if we want to be exhaustive (see Section 2.2). To the given text t, knowledge is attached in the form of labels indicating whether a reader would infer that components are not failed, failed, or maybe failed. In that respect, t comes with a partition of $\mathcal{C}_\mathcal{P}$ with three disjoint subsets N_t, F_t, and $M_t \subset \mathcal{C}_\mathcal{P}$ both corresponding respectively to the **N**ot-failed components, **F**ailed components, and **M**aybe-failed components of $\mathcal{P}$ from text t. For instance, for a review $t =$ "My battery stopped working.", we would have $N_t = \{$"screen", "motherboard"$\}$, $F_t = \{$"battery"$\}$, and $M_t = \emptyset$.

The goal of the task is to come with a model enabling an accurate mapping from a text t and a product $\mathcal{P}$ to the three sets N_t, F_t, and M_t. Therefore, we call it failed components extraction, as the main goal is to extract the failed components set F_t, while also correctly classifying not-failed and maybe-failed components.

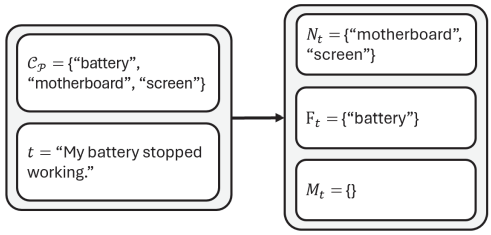

Fig. 1. FaCET idea and toy example.

Here, this formalization is limited in the sense that we only consider a multi-label classification task where the set of components is predefined. We formalize the FaCET task as a closed Information Extraction (IE) task. Another, potentially more challenging, formalization of the FaCET task would be the open IE formalization where $\mathcal{C}_{\mathcal{P}}$ is not given a priori. However, the current assumption of having predefined component labels is reasonable in the sense that such information should be fixed and known at least by manufacturers.

2.2. *Dataset description: CuReFaCET*

In CuReFaCET, there are 1,215 customer reviews, from Amazon, all dealing with products of the same type: tablet computers. Among those 1,215 reviews, 356 report at least one failure, while the remaining ones mention no failure. All reviews are labeled across 60 different component labels with either one of those values: 0 (not-failed), 1 (failed), 2 (maybe-failed). Additionally, a summary of the failure is written by a human annotator to briefly describe the failure. An uncertainty data flag is added to customer reviews whose annotations are uncertain. In particular, reviews reporting dislike, or issues tending to be quality issues rather than components losing their functionality, got flagged for uncertainty. 43 reviews have an uncertainty data flag, meaning that the dataset comprises 313 failure reviews which are labeled with certainty and no ambiguity. Besides, we work under the assumption that customers correctly identify failed components, though customers are not experts and could misidentify failed components, introducing noise. Sometimes, customers give enough indication to estimate a time-to-failure of their purchased units. In that respect, an additional attribute for time-to-failures is provided in this dataset. However, the current study does not use this piece of information.

The existence of "maybe-failed" labels is justified by the ambiguity of natural language, which in several ways can let the question "which component failed?" unanswered, while answering "Yes" to the question "did the unit fail?". Two instances of such a phenomenon are given in Table 1. For the first one, a customer describes that a hinge does not function correctly, but they do not specify which one it is. Thus, in the dataset, "Hinge" is labeled as failed, but we incorporate "Left hinge" and "Right hinge" for more granularity, both of which are maybe failed. Other reviews actually specify which hinge is failed, making this level of detail relevant. For the second example, the customer says that there are charging problems, but no additional information is given as to which component is failed. It could be due to the "Battery", to the "Charger", or even the "Charging port". For this reason, some subsystems of the product are created in the labels, like "Charging system" to account for core functions of the product, which depend on several parts.

Table 1. Examples of maybe failed components.

Excerpt	Failed	Maybe
"Just to wake it **the hinge** must be right or else I need to tilt the whole device."	A hinge	Left hinge? Right hinge?
"**Power charging problems** twice now."	Charging system	Battery? Charger? Charging port?

3. Methodology

In this work, the idea is to have a language model which, as shown in Figure 1 extracts failed components based on a given review. A major challenge however, is to enforce the LLM to structure its output so that it matches a desired format and remains consistent from one inference to another. The proposed overall framework is presented in Figure 2. There are seven configuration variables which were identified in this framework and that are listed below.

1. Language Model - The language model used for generating outputs is the main ingredient that can be changed for extracting failed components. (E.g., Gemma 2 9B, Meta-LlaMA 3 8B, etc.)

2. Prompting strategy - The prompt passed to the model to ask it to generate an answer can

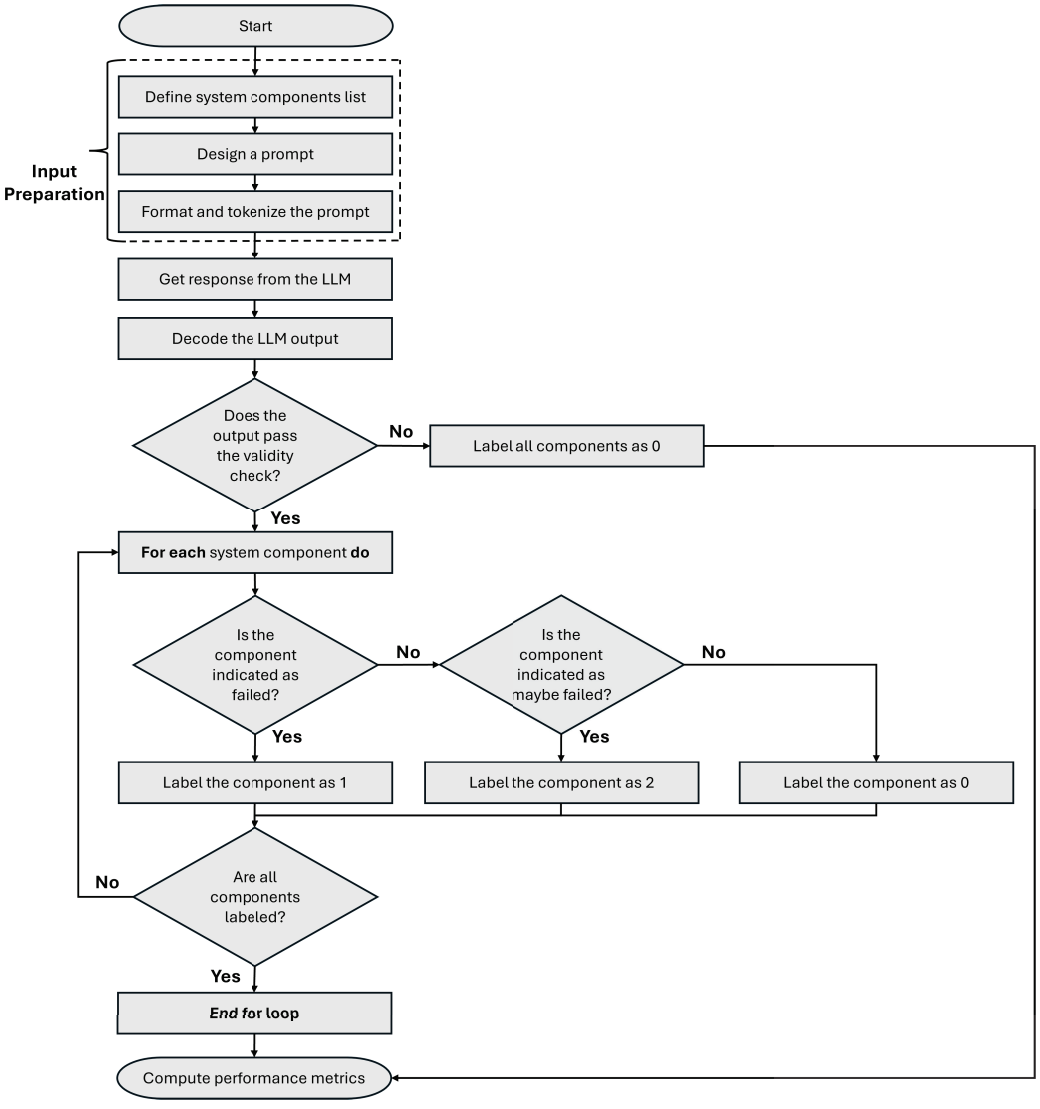

Fig. 2. Flowchart of the overall framework.

vary (E.g., do we ask for bullet points answers, or for a JSON format? Do we include maybe-failed components, or not?, etc.)

3. Performance metric - The metric used to compare predicted label vectors to reference label vectors. (E.g., Hamming loss? Exact match ratio? False positives, false negatives? How are maybe-failed labels considered?)

4. Log parsing - The method used to convert LLM outputs (logs) to label vectors that can be compared to reference label vectors (E.g., JSON extraction? Use of several assistants? What pattern for pattern matching?)

5. Data subset - The subset of the data on which performance is computed. (E.g., all data? Only data with an uncertainty data flag? Do we include examples with maybe-failed components?)

6. Component subset - The subset of components on which the performance is computed. (Do we include groups like ”Charging system” and

"Audio system", or not? This can be varied both in the prompting strategy and in the performance evaluation.)

7. Error handling - How errors are handled. (E.g., how are empty responses from an LLM treated? How are pattern matching errors handled?)

In this study, the default configuration is the following. By default, the language model used is the open-source Meta-Llama-3-8B AI@Meta (2024). The prompting strategy consists of asking for a JSON with failed and maybe-failed components, along with details. The template is provided in Figure 3. The performance metric is the average of the macro-averaged F1-scores for each component, as will be detailed in Section 4. The log parsing approach consists of pattern matching on '{}' to find JSON formats in natural language LLM answers. The data subset is the whole CuReFaCET dataset. The component subset is the whole set of components. Finally, the error handling strategy consists of matching empty responses and errors in general to not-failed label vectors.

To clearly visualize what an output of an LLM can be, given the default prompt template, we provide an example in Figure 4.

```
f'''Given the list of components of a tablet
computer: {COMPONENT_NAMES}, Give me the list of
failed components mentioned in the following review:
{REVIEW_CONTENT}
Provide the information in JSON format: {{ "Failed
components": "...", "Maybe failed components":
"...", "Details": "..." }}'''
```

Fig. 3. The default prompt template.

```
```json
{
"Failed components": ["Battery"],
"Maybe failed components": [],
"Details": "The customer states that the battery has stopped
working, but the computer still functions when plugged in."
}
```
```

Fig. 4. Example of an LLM answer.

4. Results

4.1. *Performance metrics*

The task at hand is a multi-label multi-classification task. For each instance, several labels need be assigned, and for each label, more than two classes exist. To assess models on such a task, several methods exist, and traditional approaches include using the Hamming loss, which is defined as in Eq. (1), or the exact match ratio, defined as in Eq. (2).

Let p denote the number of components, or "**p**arts" the system under study has. Here, p corresponds to the number of labels each instance has, since each label represents the state of a component of the system. Besides, p is fixed ($p = 60$ in CuReFaCET). For each instance indexed by i, we denote by y_i its true label vector, with $y_i \in \mathcal{C}^p$, where $\mathcal{C} = \{0, 1, 2\}$ is the set of classes for each label (0 for not-failed, 1 for failed, and 2 for maybe-failed). Let $\hat{y}_i$ be the predicted label vector for instance i.

Now, for each component indexed by $1 \leq j \leq p$, let y_i^j (resp. $\hat{y}_i^j$) be the label of component j according to y_i (resp. $\hat{y}_i$). Let us denote by y and $\hat{y}$ the tensors containing all y_i instances and all $\hat{y}_i$ instances. Let n be the number of instances over which we compute the Hamming loss. Then the Hamming loss is calculated as given in Eq. (1).

$$HL(y, \hat{y}) = \frac{1}{np} \sum_{i=1}^{n} \sum_{j=1}^{p} \mathbb{1}_{y_i^j \neq \hat{y}_i^j} \tag{1}$$

The metric that we want to maximize is in fact the Hamming score, which is $HS = 1 - HL$. It corresponds to a sort of naive accuracy where we compute how many labels are correctly predicted across all instances and all components. However, one limitation of this metric is that it can be very high, especially in highly imbalanced scenarios.

Another metric used in this work is the exact match ratio, where we compute the share of label vectors that are exactly predicted. This metric is by definition lesser than the Hamming score, and is less affected by data imbalance, though for datasets containing a large proportion of instances having the same label vectors, data imbalance remains an issue when using the exact match ratio.

$$EM(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i = \hat{y}_i} \tag{2}$$

However, the dataset of this study is highly imbalanced. There are significantly more not-failed labels across the dataset, since units usually have just a few failed components, when they fail. Besides, there are significantly more label vectors set to $0_{\mathcal{C}^p}$, since there are more customer reviews with no failure than with failure.

Therefore, while the Hamming score and the exact match ratio will be given as general indicators of the performance, we define the key performance indicator in this work as the average macro-averaged F1-score over all components (see Eq. 3). From now on, we will refer to this metric as F1-score, for conciseness.

$$F1(y, \hat{y}) = \frac{1}{p} \sum_{j=1}^{p} F1_{macro}(y^j, \hat{y}^j) \qquad (3)$$

$y^j \in \mathcal{C}^n$ (resp. $\hat{y}^j$) denotes the j-th true (resp. estimated) labels of all n instances.

4.2. *Language model benchmark*

4.2.1. *Model selection*

For all experiments, the default configuration is as stated at the end of Section 3. Starting from this configuration, we first conducted a language model benchmark, by evaluating different LLMs on the task. The models that were used in the benchmark were the open-source and below-50B-parameters models of the top 50 of the LM sys leaderboard Chiang et al. (2024), at time 2:00 PM on August 27, 2024. This leaderboard is constantly updated, thus new language models may fit the above features in the future and become relevant to try. The selected models were Gemma 2 27B Instruct, Gemma 2 9B Instruct (Google (2024)), Llama 3 8B Instruct, Llama 3.1 8B Instruct (AI@Meta (2024)), and C4AI Command-R (35B).

4.2.2. *Benchmark results*

The language model benchmark results are presented in Table 2. We include a dummy baseline, for which predicted labels are all set to 0.

The results show that the dummy baseline has the better Hamming score and Exact Match ratio, which is explained by the high data imbalances, as there are 859/1,215 reviews labeled with vectors equal to $0_{\mathcal{C}^p}$, and a total of 639 failed labels, 2,512 maybe-failed labels, and 69,749 not-failed labels, in the whole dataset. This high data imbalance also explains why the Hamming Score and Exact Match ratio suggest high performance, while the F1-score indicates otherwise, as anticipated in Section 4.1. On the other hand, all benchmarked language models outperform the dummy baseline on the F1-score, which accounts for their capability to detect failed components.The best model is Gemma-2-27B, which outperforms the dummy baseline by +18%, and the worst-performing model by +12%. Thus, the selected language model can significantly affect the performance. Besides, model size is not necessarily correlated with performance as the C4AI Command-R model stands behind Llama 3 8B and Gemma 2 9B, while having four times their sizes. Finally, when automatically parsing LLMs logs for extracting JSON formats, Gemma 2 and C4AI Command-R models each produced at most five format errors over the 1,215 customer reviews of the dataset, while the Llama 3 models produced more than one hundred JSON format errors. When such format errors were detected, the predicted labels were considered as 0s.

Table 2. Benchmark results.

Model	HS	EM	F1
Llama-3.1-8B-It	93.0%	57.5%	38.5%
C4AI Command-R	91.3%	29.8%	42.2%
Llama-3-8B-It	94.2%	61.8%	43.1%
Gemma-2-9B-It	**95.6%**	**63.5%**	**47.6%**
Gemma-2-27B-It	95.4%	61.6%	**50.9%**
Dummy baseline 0	**95.7%**	**71.1%**	32.6%

4.2.3. *Error analysis*

To improve the performance over the initial benchmark, we conducted an error analysis on the predictions from the benchmark. To analyze the errors, for each model, the five components with the worst F1-score were identified and their confusion matrices, normalized row-wise, were displayed. Then, the most common error types were detected as the non-diagonal cells of such confu-

sion matrices, with the highest values. Analyzing the confusion matrices, it appears that the most frequent type of errors occurs when maybe-failed components are predicted as not-failed components. The components with the most errors are specific parts of tablet computers, for which very few true failures are annotated. For instance, as CuReFaCET comprises 2-in-1 tablets equipped with a keyboard, some components are specific keys, which sometimes failed. However, when the keyboard failed, customers would rarely specify which key failed. Thus, models face difficulties in correctly deciding between 0 labels and 2 labels.

4.3. *Handling maybe-failed components*

4.3.1. *Prompt engineering*

To mitigate the issue of having too many maybe-failed components predicted as not-failed, the prompt template used for FaCET was changed. We incorporated additional guidelines to improve the detection of maybe-failed components, while making explicit the labeling rules relative to maybe-failed annotations. The following lines were added at the end of the default prompt template: "\n In addition, here are guidelines to follow:\n- If the review specifies that at least one component failed, all components that are not mentioned as failed should be considered as not failed.\n- However, if no component is specified as failed, but we know based on the review that the unit failed, then all components should be considered as maybe failed."

4.3.2. *New results*

The results obtained with the new prompt template are provided in Table 3. Except for Gemma 2 models, the language models obtain worse scores for all metrics, compared to the initial benchmark. This is explained by a significant increase in errors where a not-failed component is predicted as maybe-failed, while the models still struggle to correctly distinguish labels 0 and 2. The Gemma 2 models however significantly increased their performance, reaching 60.2% F1-score, +27% above the baseline and the other model families. Also, Gemma 2 9B outperforms its larger version in this scenario, while matching the dummy baseline on the Hamming score and exact match ratio, and pushing further the F1-score.

Table 3. Benchmark results.

Model	HS	EM	F1
Llama-3.1-8B-It	77.0%	54.4%	35.6%
C4AI Command-R	78.5%	7.8%	41.1%
Llama-3-8B-It	70.8%	46.0%	38.0%
Gemma-2-9B-It	**95.7%**	**69.7%**	**60.2%**
Gemma-2-27B-It	92.3%	60.6%	**55.4%**
Dummy baseline 0	**95.7%**	**71.1%**	32.6%

5. Discussion and Conclusion

This study introduced the FaCET task and a novel text dataset for failure information extraction. The inherent linguistic ambiguity was addressed by accounting for maybe-failed components, when one knows that a system failed without knowing exactly which component is at fault. The results showed the possibility to use LLMs to extract structured failure information from natural language data. In particular, JSON formats were used to structure knowledge and a special "Details" attribute (see Figure 3) was used to tackle LLMs verbosity and trick them into writing long explanations in a specific slot. In the end, the best models reached a F1-score $>60\%$, significantly improving over the baseline and weaker models. These results highlight the potential for extracting failure data from non-expert text. A targeted output of FaCET is to obtain visual representations of systems showing which parts are failed (see Figure 5 for instance). The extracted information can be used in real-world reliability assessments by connecting them with time-to-failure information obtained either from the text corpus under study, metadata, or related databases containing temporal information.

The main challenges included the high data imbalance of the dataset, which led to the careful selection of a performance metric, and the resource-intensive use of LLMs, necessitating multi-GPU setups with 1-2 hours per dataset pass, per model. Future directions include benchmarking state-of-the-art proprietary models, continuing the error analysis to refine the p rompt template which has

Fig. 5. Example of system health state visualization.

not been over-engineered yet, annotating degraded state labels, and extracting more failure-related information, like time-to-failure.

Acknowledgement

This work was performed using computational resources from the "Mésocentre" computing center of Université Paris-Saclay, CentraleSupélec and École Normale Supérieure Paris-Saclay supported by CNRS and Région Île-de-France.

References

AI@Meta (2024). Llama 3 model card.

Arif-Uz-Zaman, K., M. E. Cholette, L. Ma, and A. Karim (2017, August). Extracting failure time data from industrial maintenance records using text mining. *Advanced Engineering Informatics 33*, 388–396.

Chiang, W.-L., L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica (2024). Chatbot arena: An open platform for evaluating llms by human preference.

Google (2024). Gemma.

Meunier-Pion, J., Z. Zeng, and J. Liu (2021). Big Data Analytics for Reputational Reliability Assessment Using Customer Review Data. In *Proceedings of the 31st European Safety and Reliability Conference (ESREL 2021)*, pp. 2336–2343. Research Publishing Services.

OpenAI (2024, March). GPT-4 Technical Report. arXiv:2303.08774.

Yang, G. (2007). *Life Cycle Reliability Engineering* (1st edition ed.). John Wiley & Sons, Inc.

Zhang, T., A. Bhatia, D. Pandya, N. V. Sahinidis, Y. Cao, and J. Flores-Cerrillo (2020). Industrial text analytics for reliability with derivative-free optimization. *Computers & Chemical Engineering 135*.