From Prediction to Prescription: Large Language Model Agent for Context-Aware Maintenance Decision Support

Haoxuan Deng[1, *], Bernadin Namoano[1], Bohao Zheng[1], Samir Khan[1], and John Ahmet Erkoyuncu[1]

[1]*School of Aerospace, Transportation and Manufacturing, Cranfield University, Bedford, MK43 0AL, UK*

{haoxuan.deng, bernadin.namoano, bohao.zheng, samir.s.khan, j.a.erkoyuncu}@cranfield.ac.uk

ABSTRACT

Predictive analytics with machine learning approaches has widely penetrated and shown great success in system health management over the decade. However, how to convert the prediction to an actionable plan for maintenance is still far from mature. This study investigates how to narrow the gap between predictive outcomes and prescriptive descriptions for system maintenance using an agentic approach based on the large language model (LLM). Additionally, with the retrieval-augmented generation (RAG) technique and tool usage capability, the LLM can be context-aware when making decisions in maintenance strategy proposals considering predictions from machine learning. In this way, the proposed method can push forward the boundary of current machine-learning methods from a predictor to an advisor for decision-making workload offload. For verification, a case study on linear actuator fault diagnosis is conducted with the GPT-4 model. The result demonstrates that the proposed method can perform fault detection without extra training or fine-tuning with comparable performance to baseline methods and deliver more informatic diagnosis analysis and suggestions. This research can shed light on the application of large language models in the construction of versatile and flexible artificial intelligence agents for maintenance tasks.

1. INTRODUCTION

Predictive analytics for product health management has attracted increasing attention from the industry with the rise of machine learning in the last decade. With the advent of advanced data processing and statistics methods, features and patterns of the system's running state can be captured from historical logs and sensor data. By doing this, potential system failure can be forecasted and allow people to outline the plan for maintenance or adjustment in advance of system deterioration. This can not only prolong the lifespan of the system but also lead to lower costs of periodic checking and overhauls in traditional preventive and reactive maintenance (Zonta et al., 2020).

Since the 2010s, deep learning that builds upon artificial neural networks (ANNs) played an essential role in predictive analytics and performed state-of-the-art results in many situations. Without tedious and complex feature engineering, deep learning can be effectively and efficiently applied to different data formats and draw relatively accurate predictions in an end-to-end way compared to other methods.

Even though the exciting breakthrough brought by deep learning for predictive maintenance, most of the research on this topic mainly focuses on boosting prediction metrics of the proposed methods such as precision or recalling rate, which provide limited information to the maintenance plan outlining (Roy et al., 2016). A higher prediction accuracy may indicate a more stable and reliable alarm in practical applications but does not necessarily suggest helpful decision support. It is practical and meaningful to know how to address an issue rather than merely anticipate it, especially in dealing with a complicated system containing numerous variables. Under this circumstance, predictions may only be treated as notifications and consequently ignored by human operators due to restricted proactive guidance. Thus, there is a strong call for extending machine learning beyond predictor to a more engaged advisor for action recommendation and insightful analysis (Matyas et al., 2017).

The mentioned main issue cannot be overcome by pure data-driven approaches based on statistics and algorithms since data collected from sensors only represent low-level signal patterns that are hard to analyze by human beings. Thus, it is difficult to form useful and helpful advice or guidance for decision-making (Sapna et al., 2019). To address this challenge, knowledge of contextual information is required to elaborate the prediction results into high-level representations such as natural language or graph-structured data so that human beings can view them straightforwardly. Hence, the industry calls for a more advanced agent system that can generate human-understandable descriptions for reviewing and validation based on detected faults.

The critical gap lies in the missing link between sources on the low-level data side and the high-level knowledge side.

In the last five years, there has been dramatic progress in the natural language process (NLP) because of the occurrence of large language models (LLMs). By pretraining a very deep neural network with billions of parameters on an extensive textual corpus, the LLMs can be multi-task learners with impressive performances on a wide range of tasks including article summarization, multilingual translation, and text generation. More importantly, recent research indicates the emergent capability of the LLMs for multi-step reasoning to accomplish more complex tasks without much human supervision and hardcore programming (Bommasani et al., 2021).

This exciting phenomenon indicates a solution to the mentioned challenge that the link can be regarded as a step-by-step transformation workflow starting from data to knowledge using LLMs with proper prompts. A basic idea is to allow LLMs to be aware of the fault in the system at first, and then parse relevant search queries related to the predicted fault for knowledge retrieval in the database. The obtained search results can then be combined and summarized as a document for action recommendation. In this way, the LLM is an information fusion unit to elaborate predictions with information from different databases for decision support.

According to this motivation, in this research, GPT-4, a popular large language model released by OpenAI (OpenAI et al., 2023), is applied to implement the above idea. The agent is built upon a fault classification model and external knowledge databases for the retrieval-augmented generation of the system maintenance support documentation. In addition, a use case on linear actuator fault diagnosis will be conducted for proof-of-concept verification. In summary, the contribution of the paper can be summarized as:

Develop an agent for linking prediction results with the knowledge base to generate descriptions for maintenance decision-support based on the large language model.

The remaining of the paper is organized in the following structure. In section two, some related work of this research will be presented for a preliminary introduction to the critical concept used in the proposed method. In section three, the system diagram and the framework will be illustrated in detail including the principles and workflow of the method. Then, there is a use case for linear actuator fault diagnosis will be conducted and experiments will be carried out to first show the effectiveness of methods for fault detection without extra training and fine-tuning. After that, another experiment will show how the LLMs can output a more context-related conclusion for a more satisfactory decision support delivery based on predictions.

2. Related Work

2.1. Random Convolution Kernel Transformation (ROCKET) for Time Series Classification

For system state monitoring, multiple sensors will be installed on an asset to record a series of time-ordered data points during the system running. The collected time series will vary when the equipment works under different conditions, conversely, the time sequence data can represent in what situation the system is working and suggest what potential fault will probably be. To build the relationship between the time sequence with the corresponding system state, a classifier is the most effective way to implement, and this task is called Time Series Classification (TSC). It is one of the basic and essential time series mining that aims to assign unseen samples with labels in the training data by pattern exploitation. With TSC, a real-time collected time series can be categorized into states for a quick diagnosis. Therefore, the TSC is vital and commonly blind tightly to the industrial Internet of Things (IoT) for automatic system fault detection.

However, it is a challenge to apply conventional statistical or machine learning methods for the TSC. The main reason is the continuity property of the sequence of observable data points along time. Unlike textual data, which can be discretized by a set of sub-words (tokens) for processing, it is difficult to figure out the proper segmentation and transformation of the given time series for dimensionality reduction. This will cause an issue called the ‘curse of dimensionality’ and the model will be hard to recognize and capture discriminative features in the data for categorization.

There are fruitful results in TSC (Faouzi, n.d., 2022). Baseline methods such as K-nearest neighbors (KNN) classification with dynamic time warping (DTW), the bag-of-pattern method (BoP), and the remarkable ensemble classifier HIVE-COTE are proposed for this purpose, but they suffer from heavy computation and memory usage. Approaches based on deep learning such as recurrent neural network (RNN), InceptionTime (Fawaz et al., 2019), and relatively new Transformer-based models (Nie et al., 2022) are becoming popular. Although these methods boosted the accuracy and are able to generalize to different datasets compared to traditional machine learning, the model has to be trained to optimize parameters for inference which are either time-consuming or resource-intensive. Even worse, all the deep learning methods require retraining when samples are out of training data leading to a low extendibility in industrial applications.

To address these challenges, random convolution kernel transformation, short for ROCKET, was proposed to transform the time series into a vector representation using a random convolution kernel for classification (Dempster et al., 2019). Unlike conventional convolutional neural networks (CNN), parameters in the ROCKET are generated randomly and require no optimization or fine-tuning during the data transformation. Without an iterative learning process, the

ROCKET is efficient in computation and can adapt to different time sequences. Also, the ROCKET combined with traditional classification models such as the 1-NN classifier, support vector machine (SVM), and ridge classifier can achieve or even exceed state-of-the-art TSC algorithms with lightweight computation in an endurable timespan.

Therefore, considering the computational efficiency, extendibility, and performance, the ROCKET will be applied as the method for time series processing in this project. Using ROCKET as an encoder for the time series classification, the vectorization result will be processed and recognized by the LLMs to outline the prescription.

2.2. Retrieval-Augmented Generation (RAG)

Research on the application of LLMs in various workflow automation is conducted to unleash the power of LLM's human-like logical reasoning and inference capability. However, one of the main challenges comes in the *hallucination issue* of the LLMs. It means that the fake or incorrect information will be generated by LLMs. This can cause failure in task performance and may hurt the trustworthy between humans and the LLMs when they are in cooperation (Huang et al., 2023).

An effective approach for alleviating the hallucinating issue is to enable the LLMs to generate their responses based on some factual evidence from other existing sources such as the internet or knowledge databases. According to the retrieved information, the LLMs can follow the requirements and instructions given in the prompt to compile information that is rooted in ground truth and users' demands. It combines the searching techniques and the generation ability of LLMs to offer reliable and user-friendly information to people. This concept is defined as retrieval-augmented generation (RAG). The RAG has successfully been used in text, image, and multimodality searching and generation, but the application in the time series analysis on industrial sensor networks is yet fully explored. This piece of research is an initial exploration of applying the same idea for time series classification and allowing the LLM can generate the document based on the prediction to mitigate the hallucinations. In this research, the RAG will be the main methodology for retrieving historical time series samples to label the newly arrived data as a prediction outcome. The fault analysis can then be generated based on the retrieved result by the LLM.

2.3. Prompting Engineering, Chaining, and LLMs Agent

To obtain desirable outcomes from the LLMs, it is crucial to craft proper instructions for model commanding, and this concept is referred to as prompt engineering. Depending on the emergent capability, the LLMs can generate responses following descriptions in the prompts, and this is an effective way to alleviate the hallucination issue. Despite the usefulness, it is also cumbersome to tune the proper prompt to get satisfactory outcomes in a trial-and-error way.

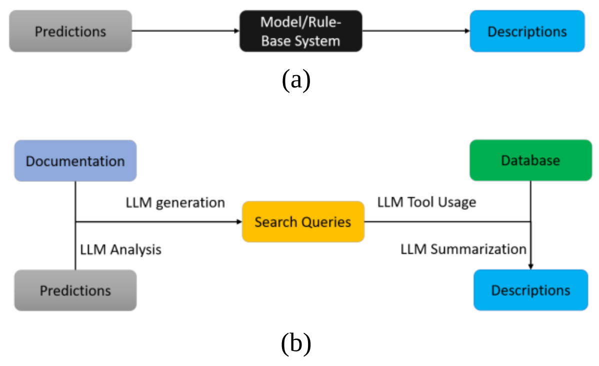

Figure 1. Different methods to convert predictions to descriptions (a). End-to-end generation (b). Multi-step transformation with the LLM

Instead of composing zero-shot prompts fully manually for LLMs to arrive at solutions immediately, (Wei et al., 2022) proposed chain-of-thought prompting that breaks down complex tasks into sequential sub-tasks and encourages the LLMs to figure out answers to each problem. By doing this, the LLMs can enhance their ability to successfully carry out intricate tasks such as math reasoning and arithmetic computations. In addition, (Yao et al., 2022) developed the ReAct prompting to enable LLMs to incorporate external tools usage and observation results obtained after tools leverage into their reasoning activity. The experiment indicated an apparent improvement in performance for LLMs on interactive text-based games and online shopping tasks as compared to traditional imitation learning or reinforcement learning approaches.

Furthermore, multiple prompts for different purposes can be serialized into a chain for workflow automation. Building on this advantage, the concept of the LLM agent, or AI agent, emerges to facilitate more functional applications of LLMs across diverse domains. The fundamental principle is to treat the LLMs as a connector or a controller among toolsets including databases, calculators, and web browsers to produce a series of actions based on their logical reasoning. In each step, LLMs can yield more reliable intermediary results and merge findings from prior stages to aggregate a more solid outcome in the final. Moreover, users can monitor the problem-solving process and understand the rationale provided by the agent, offering an opportunity for human intervention via a natural language interaction. Preliminary successful implementations of the LLMs agent in various fields are illustrated in (Xi et al., 2023).

3. Design of the Context-Awareness Agent

3.1. Initial Analysis

To compile a report of fault diagnosis with fault type, fault description, and potential recovery or maintenance strategies,

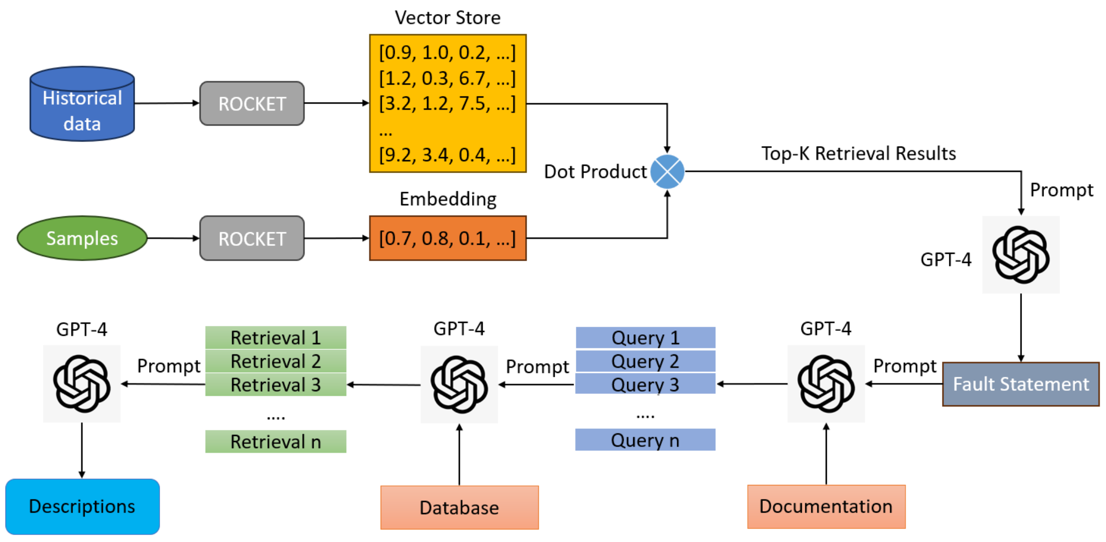

Figure 2. The diagram of the agent system

a direct way, as shown in Figure 1. (a), is to define a rule-based system that allows the prediction to go through and route to a corresponding solution. Or to craft a dataset of prediction-solution pairs to train a model for transformation. However, defining rules and collecting datasets manually are human-labor intensive and time-consuming. Also, the predefined rule-based system or model is hard to update and extend to unseen situations. From this perspective, end-to-end generation may be friendly in the development stage but may be challenging to maintain when the model or the system has been deployed in the production environment in industrial environments due to low adaptability to a dynamically evolving situation.

Another way to consider this problem is to divide the transformation into a series of steps with the LLM, external database, and tools (usually a bunch of calling Application Programming Interface), as shown in Figure 1. (b). After receiving the prediction from the data processing step, LLM will be asked to recognize the fault based on the prediction result and try to reason about what problem should be solved based on providing the contextual background described in the documentation. The generated search queries will then be thrown to a database by LLM's tools calling capabilities to retrieve relevant information on repairing suggestions. Finally, the LLM can be instructed to summarize all information into a report for human beings to review.

In this way, the workflow can be independent of a fixed set of rules and ensure contextual information is involved in the final description generation. In each step, human operators can track and offer comments or feedback to inject their expertise into the agent by providing prompting to get a more comprehensive result. The essential idea of the proposed method is *the multi-step generations based on factual evidence*, which is the core idea of the RAG. In the following section, the details of the agent system will be illustrated.

3.2. Overview of the Agent System

The diagram of the agent system is illustrated in Figure 2. A set of historical data will be transformed into a set of vectors and stored in memory for comparison. The method for sample vectorization is the ROCKET as introduced in the above section. In detail, several randomly generated convolution operators will slide the input time series to conduct dot product computation. In mathematics, according to the paper (Dempster et al., 2019), the outcome from implementing a kernel, *w*, with dilution, *d*, and bias *b*, on a specific set of time series *X* from position *i* in X_i is presented as follows:

$$X_i * w = \left(\sum_{j=0}^{l_{kernel}-1} X_{i+(j\times d)} \times w_j\right) + b \qquad (1)$$

A feature map *M* will be obtained from the kernel computation, and two real values will be extracted as features for each kernel including the *maximum values* and the *proportion of positive values* (*ppv*) in the *M* by the following formula:

$$ppv(M) = \frac{1}{n}\sum_{0}^{n-1}[m_i > 0] \qquad (2)$$

Where m_i is the numerical value in the feature map. Therefore, there will be two features produced per kernel operation, and for an effective time series representation, 10000 kernels are used to transform the data leading to 20000 features to represent each time series. The ROCKET algorithm is applied to all samples in the historical database which will be stored for retrieval. Since there is no parameter optimization and fine-tuning during the data processing, the computational efficiency can be extremely fast compared to deep learning or other statistical methodologies. Therefore, the requirement on hardware configuration is much lower enabling a constant vectorization of new samples as experience accumulation.

When an unlabeled sample comes, it will also be transformed into a vector or said embedding and compute the Euclidean distance among all embedded samples in the vector store. After sorting based on distance, the most similar records will be considered as the target and the label will be assigned to the new data for classification result. For a common RAG implementation, in each retravel, the first five similar, or said the top-5 similar records will be extracted to promise a high hitting rate. In this agent system, top-5 retrieval is applied.

In the next step, the top-5 similar retrieval results will be fed to the LLM with a prompt to warp the prediction with contextual information including background introduction, technical details, and the system configuration to form a fault diagnosis statement in the following format:

Retrieval results: [‘fault type1’, ‘fault type2’, ..., ‘fault type5’]

Diagnosis results: [‘fault type’]

Inference evidence: [fault type1 with <score1>, ...]

Description of the Fault: This state indicates that…

In this compact diagnosis report, retrieval results will be shown, and the classification is presented as ‘fault type’. In addition, inference evidence is the list of scores of the retrieval. In this case, the Euclidean distance is used for an interpretable purpose so that people can understand how the system gets the result. The more similarity between the unlabeled one and the records, the smaller the Euclidean distance will be. The description of the fault is summarized in the given document to explain to human operators clearly what is happening in the system with plain natural language.

In the next step, the brief statement is fed back to the LLM later and the fault type will be recognized for parsing the query for searching the database. For example, if the detected fault is ‘spalling’ on a ball-screw actuator given in the statement, the LLM can generate highly related several searching strings:

- How to recover spalling damage in ball-screw actuators?
- What are replacement options for ball-screw actuators with spalling damage?
- Replacement options for ball-screw actuators with spalling damage.
- Diagnosing spalling in linear actuators for effective maintenance.

These questions are then used for matching contents in a database, for instance, a general knowledge base e.g. Wikipedia, or a specific expert system with the tool usage capability of the LLM. All the obtained information will be summarized to direct maintenance suggestions and action recommendations in the final step.

4. USE CASE ON LINEAR ACTUATOR FAULT DIAGNOSIS

4.1. Experimental Setup

The time series data[1] is collected on a linear actuator system reported in the paper (Ruiz-Carcel & Starr, 2018). The detailed description including the mechanical components, structure, and parameters of configuration are all illustrated clearly in the article. This paper will not fully reintroduce the actuator system. The dataset acquired during the testing is the starting point of the introduction to the agent system application use case.

At first, the rig was operated under typical working conditions without any malfunctions to gather a substantial volume of data that represents the system’s behavior under varying loads and motion patterns. Two distinct motion profiles were examined:

- Trapezoidal profile with a constant speed set point
- Sinusoidal profile with a smooth transition speed

In this paper, only data under the trapezoidal profile is considered for simplicity, the utilization of multiple profiles can be taken into account in future upgrades. The trapezoidal profile is tested for normal and faulty conditions under three distinct load scenarios: 20kgf, 40kgf, and -40kgf. The full motion sequence was repeated 5 times in one working situation under a load as one test. Each test will be conducted 10 times repetitively to generate a dataset with an adequate amount of observation in each case studied. a total of 50 samples, in every scenario analyzed. Furthermore, three distinct mechanical flaws in different degradation levels were intentionally introduced into various portions of the system to simulate modes typically experienced by these types of machines. The faults of the system in this dataset include:

- Spalling from level 1 to level 8 (8 states)
- Lack of lubrication from level 1 to level 2 (2 states)
- Backlash from level 1 to level 2 (2 states)

In short, including the normal and all other flaw states, there are 13 different types and 650 samples in each load circumstance leading to a total of 1950 samples. For evaluation, 20% of all samples will be randomly selected to form a testing dataset, and the remaining samples will be used for constructing the vector store as introduced in section 3.2.

4.2. Implementation

For analysis, samples under all 13 flaws in each load can be visualized in Figure 3 and Figure 4 after using the moving average smoothing with the window size 20 and 15 respectively reported in the paper (Ruiz-Carcel & Starr,

1.The dataset can be found on the link: https://cord.cranfield.ac.uk/articles/dataset/Data_set_for_Data-based_Detection_and_Diagnosis_of_Faults_in_Linear_Actuators_/5097649

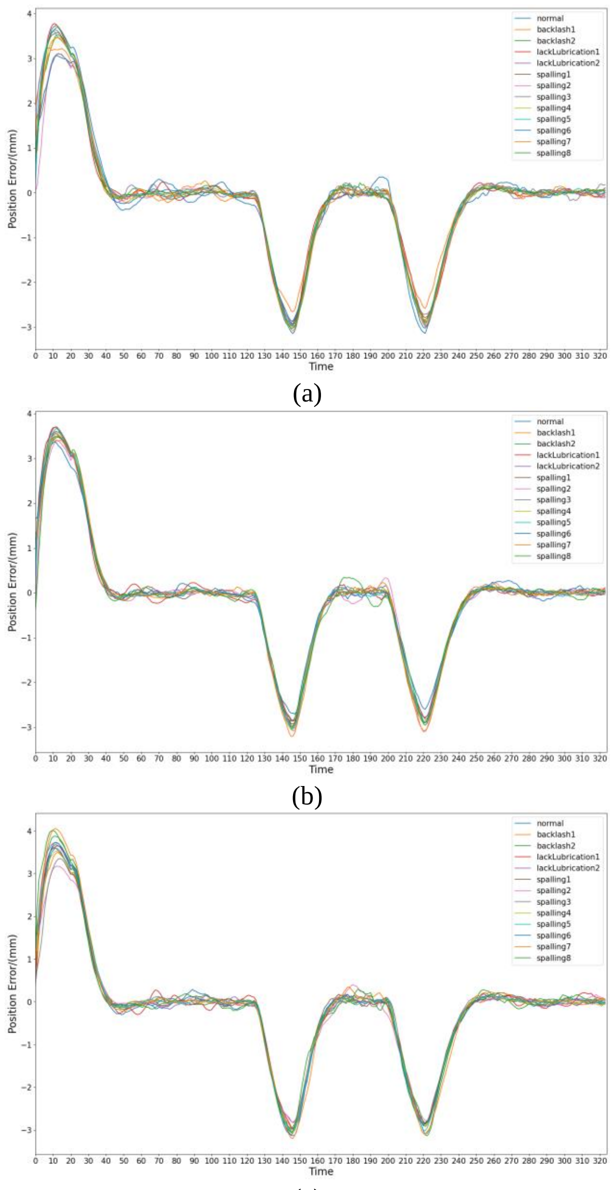

Figure 3. Position error (mm) signals in different fault states under three load conditions after smoothing. (a) 20kgf (b) 40kgf (c) -40kgf

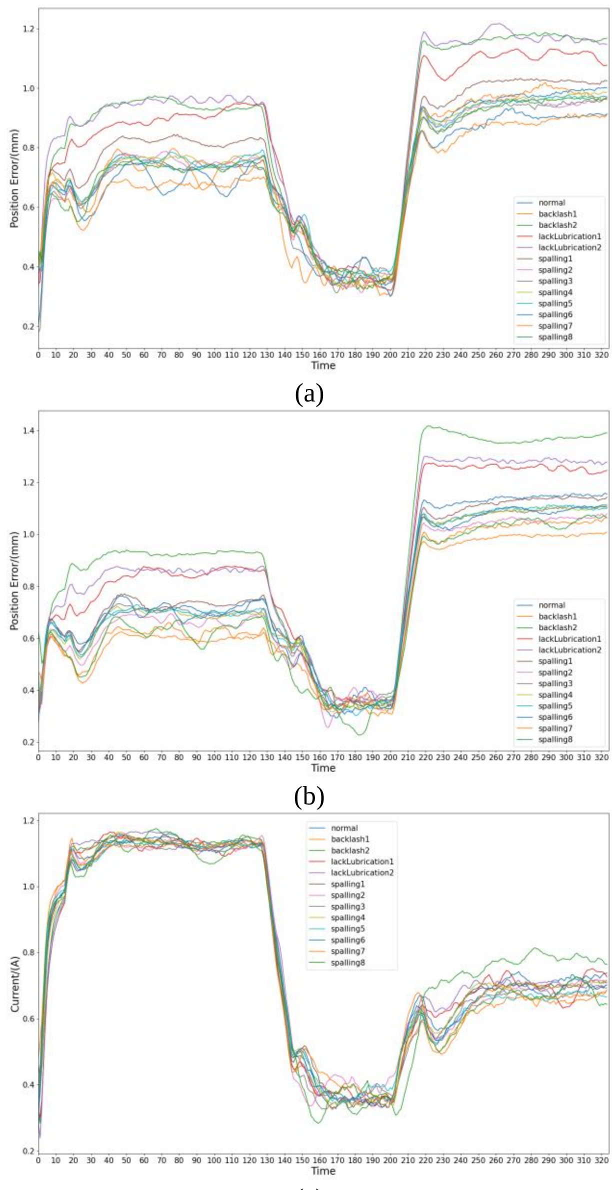

Figure 4. Current (A) signals in different fault states under three load conditions after smoothing. (a) 20kgf (b) 40kgf (c) -40kgf

2018). As shown in Figure 3, no matter in which situation, the position error signals in each fault are distributed too close to be separated from others, while the pattern of current signals is more distinguishable. Therefore, the current signal is the univariant for time series processing in this use case.

Then the univariant time series will be transformed into a vector representation using the ROCKET. To further improve the computational efficiency, a trick from the variant of the ROCKET, namely MiniROCKET (Dempster et al., 2020), is applied. The main difference is that the kernel length usage is fixed to 9, instead of randomly selected from choices {7, 9, 11} in the original ROCKET. By doing this, it can make the result more deterministic. In this use case, the ROCKET is implemented with the Python package called Pyts (Faouzi & Janati, 2020).

Then the vector consisting of features computed from each kernel (20000 dimensions) will be stored together as a vector database for retrieval.

During the validation, prediction accuracy will be tested in three different loading conditions individually and the average value will also be computed. For the diagnosis report

Table 1. Fault classification accuracy under 20kgf, 40kgf, and -40kgf loading situations with different methods.

Method	20kg	40kg	-40kg	Average
InceptionTime (with 100 epochs)	83.8462%	86.0465%	79.2308%	83.0412%
ROCKET + top-1 retriever	63.0769%	75.1938%	46.1538%	61.4748%
ROCKET + ridge classifier	80.7692%	82.1705%	83.0769%	82.0055%
ROCKET + top-5 retriever	83.0769%	91.4729%	81.5385%	85.3628%

generation, given a loading, randomly select a sample from the testing dataset to review the fault statement and the relativity between the generated recommendations and the fault type.

4.3. Results

To summarize the results of the experiment, different methods for the classification based on the ROCKET features with Euclidean distance metric are listed in Table 1. The comparison between the proposed fault detection method is compared with the performance of the strong deep learning baseline method, the InceptionTime. There is an obvious gap between the top-1 precision based on the ROCKET vectorization and the baseline algorithm, only around 60% on average versus the InceptionTime, which is more than 80% over three loading circumstances. However, the computational time can be cut down dramatically by the ROCKET method. For the same historical dataset, for instance, using all 520 samples under 20kg, the result can be obtained with CPU (i7-12700H @3.30Hz) for about 25 seconds, while using the InceptionTime with 100 training epochs, the prediction drawn from scratch requires more than 60 seconds on GPU (Nvidia GeForce RTX 3060 Laptop GPU). Even though the deep learning model can quickly conclude after training, the parameters are fixed once the training is done. When out-of-distribution samples come, the model requires to be updated without forgetting previously obtained knowledge. Retraining or fine-tuning the model in this way is still an ongoing research topic. In contrast, using the ROCKET with the distance-based metric retriever, new samples can be encoded in nearly real-time to query existing vector databases to get the results. Thus, it shows the potential of on-the-fly data processing capability in industrial applications.

One way to further improve the classification accuracy is the incorporation of the ridge or logistic classifiers which can reach a better prediction outcome. As shown in Table 1, the ROCKET feature with the ridge classifier can even surpass InceptionTime in terms of prediction accuracy under the load of 40kg. In addition, an alternative approach is to use a top-5 retriever. By doing this, when one of the retrieved samples indicates the correct label, the prediction can be treated as correct. It can obviously boost the prediction accuracy (up to 85%) compared to any other top-1 classifiers, while with the cost of bringing the noise and uncertainty by considering more historical samples. Some types of faults may have highly similar patterns in the time series data, resulting in their simultaneous extraction as targets in the retrieval process. For instance, the ‘spalling2’ sequence can potentially retrieve ‘spalling1’ or ‘normal’ records from the vector database because their shapelets share great similarities. More importantly, this kind of uncertainty is usually unknown when deploying the system into the real production environment and delivering imprecise information to the users.

This problem can be addressed by refining the fault in a prompt in the brief fault diagnosis statement generation using the LLM. This is meaningful, as in real-life scenarios, system degradation occurs gradually and may not have a clear boundary or change in different malfunctions from a macro perspective. Therefore, two similar types of faults such as ‘spalling1’ and ‘spalling2’ may have nearly identical effects to the system and appear to have the same level of degradation. Thus, they can be regarded as the same fault type when proposing maintenance suggestions in practice. By reassigning these faults with a shared label in the prompt for the LLM, we can shrink the noise introduced by the top-5 retriever and take practical considerations into account while generating maintenance suggestions, which will make them more useful and accurate in real-life situations. This is the first step towards integrating realistic contextual information into the fault diagnosis process, making it more reliable and practical.

After getting the refined fault labels, the LLM (in this case, the GPT-4) can be instructed for multiple query generation, database retrieval, and the final summarization with handcrafting prompts step-by-step as illustrated in Figure 2 (The completed prompt can be found in the project repository listed in the Appendix). This can compile the final diagnostic report for converting the classification result to an actionable plan. Also, the Google Chrome web browser is selected as a general knowledge base used for query searching in this proof-of-concept verification. As a result, Figure 5 shows a part of the generated document indicating that the GPT-4 can give context-aware suggestions to a detected fault.

5. Discussions and Future Work

The use case successfully demonstrates the feasibility of applying the LLM to convert the single prediction to the

Retrieval results: ['spalling7_40_1_5', 'spalling7_40_9_3', 'spalling7_40_10_4', 'spalling7_40_7_3', 'spalling7_40_10_2']
Diagnosis results: Refined fault type1: Obvious fault in spalling
Inference evidence: [spalling7_40_1_5 with 3.1987850822921007, spalling7_40_9_3 with 3.772884271423856, spalling7_40_10_4 with 3.989529120390713, spalling7_40_7_3 with 4.239072661571159, spalling7_40_10_2 with 4.822921558144848]
Description of the Fault: This state indicates that the ball-screw mechanism within the actuator has significant surface damage. This level of spalling affects the actuator's smoothness and efficiency severely. The presence of such a fault suggests that the actuator's performance and longevity are compromised, necessitating immediate attention and likely repair or replacement to restore normal operation.

Searching for:
how to repair spalling damage in ball-screw actuators
best practices for preventing spalling in linear actuators.
replacement options for ball-screw actuators with spalling damage
diagnosing spalling in linear actuators for effective maintenance
cost-effective solutions for spalling damage in actuator mechanisms

Searching results:
Reconditioning the Actuator's Ball Screw: This involves repairing and/or replacing worn or damaged internal components of the actuator.
Replacing Cover Bands: When necessary, the cover bands should be replaced to ensure the proper functioning of the actuator.
Addressing Premature Flaking or Abnormal Damage: This may include looking for the cause of the problem, such as careless handling, excessive misalignment, or insufficient lubrication, and taking measures to prevent machine failure.
Checking for Damage: It is crucial to check for metal fragments that can indicate broken parts and could cause further damage. Also, inspect ball nut threads for damage and replace them if necessary.
Troubleshooting and Countermeasures: Check for specific damage conditions like the raceways of the screw shaft and ball nut and/or the surface of the ball peeling off like scales. Implement countermeasures based on identified causes, such as misalignment or lack of lubrication.
Seeking Professional Help: For severe damage or complex repairs, consulting with or hiring a professional service that specializes in actuator repair might be beneficial. Emergency repair teams can restore damaged ball screws and actuators when time is critical.
Observing Safety and Proper Procedures: Always follow safe handling practices and the manufacturer's instructions to minimize the risk of further damage during the repair process.
…

Repair and Reconditioning Strategies
To address immediate damage and restore functionality, the following steps are vital:
Ball Screw Reconditioning: Essential for repairing or replacing the internal parts of the actuator to improve functionality.
Cover Bands Replacement: Required when these components are damaged to maintain efficiency.
Addressing Underlying Causes: Investigating and rectifying root causes like misalignment or insufficient lubrication to prevent recurrence.
Damage Inspection: Identifying broken parts through inspection is crucial for preventing further damage.
Implementing Countermeasures: Specific damage conditions should be examined and addressed accordingly, emphasizing the correction of misalignment and proper lubrication.
Professional Consultation: Seeking expert advice is recommended for complex or severe damages.
Safe Handling and Procedure Adherence: Following manufacturer guidelines and safe practices is imperative throughout the repair process.
…

In summary, the effective management of spalling in balls crew actuators involves a balanced approach of timely and appropriate repairs, adherence to best practices for prevention, and the consideration of replacement when necessary. The combination of condition-based maintenance and leveraging cost-effective technological solutions plays a crucial role in enhancing the longevity and reliability of these actuators, ultimately ensuring their optimal performance in various applications.

Figure 5. A partial piece of an example document for the randomly selected 'spalling7' fault.

prescription of repairing and reconditioning strategies for maintenance decision support. Without any extra training or fine-tuning, and no requirement on manual feature engineering, dataset construction, and rule-based system definition, the LLM can automatically link different (public or private) knowledge sources to compile a reasonable solution after the fault diagnosis based on humans' intention. Therefore, it improves the functionality of current machine learning as a more proactive and user-friendly production for industrial applications. In addition to the work presented in this paper, there are a few interesting directions that can be explored in the future.

From Univariant to Multivariant: In this study, only univariant time series (the current signal) is considered for the ROCKET feature construction, while the data related to position error has been ignored due to the similar shape patterns in the time domain. This raises the question of how to incorporate multiple time series patterns into the vector store section for fault diagnosis. The ROCKET can be extended to process multivariant time series, and it is a candidate update to the proposed agent system with this capability. In addition, all the introduced methods are in the time domain, how to integrate information from the frequency domain into the proposed framework is another question. By doing this, the time series can be analyzed from different points of view and construct more distinguishable features for retrieval with less amount of data.

From Suggestion to Automation: The report generated after the workflow is expected to offer assistance to people in maintenance planning. A further idea is to explore how to connect the decision with the action to automate the entire maintenance process from fault detection and identification to system recovery and reconfiguration. A quick idea is to extend the sequentialization of prompts till to execution stage by incorporating external application programming interfaces (APIs) to directly link to actuators for the system maintenance. Recent relevant research is conducted for this purpose such as code-as-policy (Liang et al., 2023). In this way, it is exciting to develop an automated agent that can be self-awareness, self-decision, and self-action to the system health management without too much human intervention. However, it is also important to note that the verification and evaluation from the human side are critical to ensure the final action satisfies all practical and aesthetic requirements in production environment. How the agent can learn from human feedback to further align their performance with our expectations and values is a critical consideration for this direction.

From Ad-hoc Prompting to Long-Term Memory: The prompts used in this paper are yet fully automatically generated. Handcrafting is still needed during the prompting process. For every generation, the GPT-4 should reload all information from scratch and provide suggestions merely limited to the information written in the prompt. Hence, the agent cannot view and refer to any of previous diagnostic reports to improve its performance and keep accumulating experience for future analysis. Some studies show that if an agent can learn from the contents generated by itself, after self-learning on these contents, the performance may improve and even exceed the human level. AlphaGo, for instance, can self-play with enormous virtual games by itself and eventually defeat top-ranked human players (Silver et al., 2017). A further question is whether this similar idea can be applied to the agent. If the prompting and previous diagnosis reports can be stored in long-term memory and retrieved for new situations by the agent itself. It is possible to let the agent itself to prompt itself automatically with lower human supervision. This can cut down the requirement of human knowledge and computation time. Also, it can increase the likelihood of producing more optimal solutions that people have yet to conceive.

From the Given Knowledge Base to Self-Exploration: It is noticeable that automation is built upon a human-defined logic written in prompts. Thus, the basis of automation still relies much on human labor and insight. More importantly, the knowledge base for agent retrieval is also created and mainly maintained by human beings, thus, heavily restricting the potential of machine knowledge discovery. It is then followed by a question of how to allow the machine to acquire knowledge with the self-exploring capability to discover new methods to alleviate human bias and errors in maintenance tasks. Furthermore, it is fascinating to investigate how to enable the agent to contribute to the existing knowledge with human beings together for knowledge acquisition in the system health management domain.

6. Conclusion

This paper presents an LLMs agent-based method for elaborating predictions from machine learning to actionable strategy descriptions for maintenance decision support. A use case of linear actuator fault diagnosis is studied with an agent built upon ROCKET time series representation, the concept of RAG, and the prompts chaining technique. By prompting engineering, the LLM agent can recognize the fault and parse highly relevant queries to the database using a search tool, (in this case, the Google Chrome web browser), and summarize the retrieval results to report to human operators. The study demonstrates the possibility of constructing autonomous agents for proactive decision assistance without much human supervision and training and shows how current LLMs can be integrated into the industrial workflow.

Acknowledgement

This research was supported by the Center for Digital Engineering and Manufacturing at Cranfield University (UK)

Appendix

The code for this project can be found on the link: https://github.com/BlueAsuka/Rocket-RAG

References

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2021). *On the Opportunities and Risks of Foundation Models*. https://arxiv.org/abs/2108.07258v3

Dempster, A., Petitjean, F., & Webb, G. I. (2019). ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, *34*(5), 1454–1495. https://doi.org/10.1007/s10618-020-00701-z

Dempster, A., Schmidt, D. F., & Webb, G. I. (2020). MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 248–257. https://doi.org/10.1145/3447548.3467231

Faouzi, J. (n.d.). *Time Series Classification: A review of Algorithms and Implementations*. Retrieved March 23, 2024, from https://inria.hal.science/hal-03558165

Faouzi, J., & Janati, H. (2020). pyts: A Python Package for Time Series Classification. *Journal of Machine Learning Research*, *21*(46), 1–6. http://jmlr.org/papers/v21/19-763.html

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., & Petitjean, F. (2019). InceptionTime: Finding AlexNet for Time Series Classification. *Data Mining and Knowledge Discovery*, *34*(6), 1936–1962. https://doi.org/10.1007/s10618-020-00710-y

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. https://arxiv.org/abs/2311.05232v1

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., & Zeng, A. (2023). Code as Policies: Language Model Programs for Embodied Control. *Proceedings - IEEE International Conference on Robotics and Automation*, *2023-May*, 9493–9500. https://doi.org/10.1109/ICRA48891.2023.10160591

Matyas, K., Nemeth, T., Kovacs, K., & Glawar, R. (2017). A procedural approach for realizing prescriptive maintenance planning in manufacturing industries. *CIRP Annals*, *66*(1), 461–464. https://doi.org/10.1016/J.CIRP.2017.04.007

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. https://arxiv.org/abs/2211.14730v2

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2023). *GPT-4 Technical Report*. https://arxiv.org/abs/2303.08774v6

Roy, R., Stark, R., Tracht, K., Takata, S., & Mori, M. (2016). Continuous maintenance and the future – Foundations and technological challenges. *CIRP Annals*, *65*(2), 667–688. https://doi.org/10.1016/J.CIRP.2016.06.006

Ruiz-Carcel, C., & Starr, A. (2018). Data-Based Detection and Diagnosis of Faults in Linear Actuators. *IEEE Transactions on Instrumentation and Measurement*, *67*(9), 2035–2047. https://doi.org/10.1109/TIM.2018.2814067

Sapna, R., Monikarani, H. G., & Mishra, S. (2019). Linked data through the lens of machine learning: An Enterprise view. *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*. https://doi.org/10.1109/ICECCT.2019.8869283

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature 2017 550:7676*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, *35*. https://arxiv.org/abs/2201.11903v6

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., … Gui, T. (2023). *The Rise and Potential of Large Language Model Based Agents: A Survey*. https://github.com/WooooDyy/LLM-Agent-Paper-List.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). *ReAct: Synergizing Reasoning and Acting in Language Models*. https://arxiv.org/abs/2210.03629v3

Zonta, T., da Costa, C. A., da Rosa Righi, R., de Lima, M. J., da Trindade, E. S., & Li, G. P. (2020). Predictive maintenance in the Industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, *150*, 106889. https://doi.org/10.1016/J.CIE.2020.106889