Generation of Semantic Knowledge Graphs from Maintenance Work Orders Data

Renita Tahsin [b], Yunqing Li [a], Mohammad Sadeq Abolhasani [c] and Farhad Ameri [a, *]

[a] *School of Manufacturing Systems and Networks, Arizona State University, Tempe, AZ, U.S.A,*
E-mails:farhad.ameri@asu.edu, yunqing.connie.li@asu.edu
[b] *Texas Children's Hospital, Houston, TX, U.S.A*
E-mail: renita.tahsin@gmail.com
[c] *School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, U.S.A,*
E-mail: mabolhas@asu.edu

Abstract. Industrial maintenance activity data is typically stored in unstructured form within the databases of maintenance management systems. For this reason, effectively exploring the data and uncovering valuable patterns concealed within it is often highly challenging. Consequently, historical maintenance data is seldom analyzed or reused for purposes such as failure prevention, maintenance history reconstruction, or maintenance diagnostics. If the knowledge embedded in maintenance data is liberated and formalized, it can significantly improve the intelligence of maintenance management systems by enabling knowledge reuse. This research aims to help advance the progression from data to information and knowledge through data-driven creation of knowledge graphs built from the unstructured data available in maintenance work orders. A Simple Knowledge Organization System (SKOS) thesaurus is used to support automated entity extraction from text. The thesaurus is extended with the aid of a fine-tuned Large Language Model (LLM). A formal ontology provides the semantics of the knowledge graph. A software tool is developed to streamline the semi-automated text-to-graph translation process. The proposed framework was validated based on 100 work orders extracted from the computerized maintenance management system of a construction equipment manufacturer. The experimental validation proved that graph-based representation of work order data could effectively enhance information retrieval, analysis, and pattern extraction particularly if it is supported by formal ontology and rule-based reasoning methods.

Keywords: knowledge graph, maintenance work order, ontology, thesaurus, large language model

1. Introduction

Maintenance is defined as the actions intended to retain an asset or restore it to a state in which it can perform a required function [1]. In order to minimize machine downtime and maximize the availability of critical assets and equipment uptime, manufacturers use a variety of maintenance management tools and systems with varying levels of automation and data processing capabilities. In particular, Computerized Maintenance Management Systems (CMMS) are widely used in most industries as a centralized platform to manage, plan, and organize preventive and planned maintenance activities [2]. Despite the widespread adoption of maintenance automation solutions in the

*Corresponding author. E-mail: farhad.ameri@asu.edu

industry, maintenance management is still a highly human-centric process since efficient acquisition, formalization, and reuse of maintenance knowledge is a major challenge [3].

One of the key features of CMMS is Work Order Management which entails creating, assigning, and tracking work orders for maintenance tasks. A Maintenance Work Order (MWO) is a detailed document that contains information about a specific maintenance task or job. Key elements typically included in a maintenance work order are asset information, problem description (observation), priority level, assigned personnel, and diagnosis and completion notes. Below is an example problem description in a typical MWO:

> *" .. the conveyor belt on equipment ID EQ-987654321 is misaligned, causing excessive friction and abnormal noise. The misalignment is near the drive pulley. There are visible signs of wear and tear on the belt surface"*

This example describes a failure (misalignment) for a conveyor belt, its associated effects (excessive friction, abnormal noise), and the potential cause (belt wear). MWO records are often stored in the semi-structured database of CMMS packages for archiving, reporting, or analysis purposes [4]. Enormous collections of historical maintenance logs, representing a wealth of diagnostic knowledge, can be found in most industries. The knowledge embedded in MWOs can be used to inform the maintenance diagnosis process. However, the data provided by MWOs are often under-used since it is not presented in a computable form [1]. Additionally, the data is often incomplete and flooded with misspellings, specialized vocabulary, abbreviations, and contradictory statements since they are prepared by human technicians and reliability engineers with varying levels of experience and background knowledge. There is a need to develop systematic methods and models for converting work order data into more formal representations to improve the reusability and findability of the knowledge embedded in MWOs. As the number of MWO records increases, manual search and analysis of those records becomes more cumbersome and less efficient. Without proper tools and techniques for analyzing, mining, and contextualizing the data, the usefulness of maintenance logs is severely limited.

Advanced techniques supported by Natural Language Processing (NLP) and Machine Learning (ML) can be applied to extract useful patterns and rules from the raw text that are otherwise hidden in the historical maintenance work order data. These techniques are particularly useful for cleaning maintenance logs, extracting important terms, classifying, and clustering similar work orders, and identifying associativity relationships between the extracted terms. One difficulty in the effective analysis of MWO data using NLP is that the size of MWO data is often smaller than what is needed by most NLP tools [5].

Current methods of representing historical maintenance data primarily involve traditional relational databases and spreadsheet-based systems. The traditional methods often fall short of capturing the complex relationships and interdependencies between different pieces of maintenance data and properly contextualizing the data. Converting the MWO data into a knowledge graph provides a formal and semantically rich representation of maintenance information. Knowledge graphs are quickly becoming popular models for data representation since they promise to bridge the gap between the data and its meaning, and they serve as the natural data model for data integration. Knowledge graphs help maintain data quality and consistency. Additionally, by structuring work order data into a knowledge graph, the contextual information associated with each maintenance task can be represented, thus providing a more comprehensive view of the maintenance landscape. If the knowledge graph is aligned with a formal ontology, then additional benefits such as automated reasoning and inference can be obtained.

There have been some major efforts for automated generation of knowledge graphs from MWO data supported by deep learning and the NLP pipeline [6]. Such works are significant achievements in this field, but they still need to be further developed to address some shortcomings, including a lack of alignment with formal ontologies.

The underlying research challenge that motivates this work is to generate a semantic knowledge graph, aligned with a formal ontology, from MWO text. This work proposes a hybrid approach that incorporates top-down methods for generation and curation of knowledge models by domain experts as well as bottom-up methods for data-driven extension and validation of those models. The main contributions of this paper include:

1. Introducing KnoWo, an open-source, Java-based tool designed for the human-assisted generation of RDF knowledge graphs from maintenance work order text.

2. Development of formal and reusable thesaurus-based on Simple Knowledge Organization System (SKOS) for capturing the key concepts in maintenance work orders, supported by fine-tuning of Large Language Model (LLM).
3. Developing a software tool for semi-automated generation of RDF knowledge graph from MWO text.
4. Introducing Work Order Ontology (WOO), an OWL-based framework designed for semantically structuring maintenance work order data, enabling inference and search capabilities through alignment with the Basic Formal Ontology (BFO).

The remainder of this paper is structured as follows. Section 2 provides a review of the related work. The proposed framework for text-to-KG conversion is introduced broadly in section 3. The thesaurus and ontology are introduced in sections 4 and 5 respectively. Section 6 introduces KnoWo, a tool for the generation of knowledge graphs from MWO text. The paper ends with concluding remarks.

2. Related Works

This section provides an overview of the existing works related to creating knowledge graphs from maintenance work order text. Also, some of the ontological efforts toward maintenance knowledge representation are reviewed.

In the study by Wang et al. [7], a collaborative learning cascade binary tagging framework is introduced, aiming to extract essential insights from the unstructured maintenance records supplied by China Eastern Airlines. The authors successfully built a knowledge graph for diagnosing faults in the civil aircraft environmental control system, incorporating the acquired knowledge into the graph structure.

Ding et al. [8] propose a semi-supervised approach to failure analysis knowledge graph construction. Their method utilizes a semantic module to extract contextual details and identify failure modes from maintenance records. Trained with unlabeled records, it incorporates hard pseudo-label acquisition and a novel self-training algorithm. A taxonomy induction module then extracts failure elements and interrelationships, aiding in decision-support knowledge graph creation.

Moreover, Peng et al. [9] proposed an automatic knowledge graph framework for production line fault maintenance. They construct a pattern layer of the production line fault maintenance knowledge graph, utilizing the BERT-BILSTM-CRF algorithm to extract entities and a template-based method for relationships. The inclusion of an entity recognition dataset further strengthens their study.

In a comprehensive study, Stewart et al., [10] introduced an innovative tool for extracting and visualizing technical information from maintenance work orders (MWOs). They highlighted the challenges of interpreting unstructured, jargon-rich short texts within MWOs and presented a solution through a technical language processing-based approach. The proposed system, consisting of Echidna, an intuitive query interface, and MWO2KG, a deep learning tool, automates the construction of knowledge graphs from both unstructured texts and structured fields of MWOs simultaneously. This advancement enables engineers to efficiently access and query historical asset data by functional location or asset class, identify failure modes, validate maintenance strategies, and improve processes. Moreover, the open-source availability of these tools under the Apache 2.0 License facilitates their adoption and adaptation in various industry contexts. The study also demonstrated the tools' effectiveness through industry-provided data, offering a leap toward leveraging unstructured technical texts for engineering knowledge and decision support.

In recent decades, a variety of information science techniques have been developed to enhance knowledge management and dissemination across different disciplines. Ontologies have proven effective in tackling ambiguity and consistency issues specific to knowledge sharing and reuse in various domains. Essentially, ontologies enable efficient dissemination and information exchange by explicitly defining concepts, attributes, and relationships. Their provision of consistent structures and semantics ensures the validity of the communicated information [11].

Ebrahimipour and Yacout [12] present a methodology for knowledge representation using ontology concepts, overcoming heterogeneity and inconsistency in maintenance records. Their approach combines Bayesian Graphical Model (BGM), Web Ontology Language (OWL), RDF, and ISO standards, constructing transparent cause-effect knowledge for maximum shareability and accessibility.

Ringsquandl et al. [13], however, tailored Ontology-Based Data Access (OBDA) to create an RDF knowledge graph for Siemens smart factories, focusing on optimizing maintenance operations. They illustrated the digital twin's

role as an interface to the physical system, enabling optimization and self-organization without direct interaction. Their emphasis on improving system performance involved identifying missing information between instances of a class. The knowledge graph encompassed master, operational, and transactional data, augmenting the vector space of log files generated by manufacturing equipment. Following OBDA development, a machine learning approach was employed to detect missing entities in RDF triples.

Xu et al. [14] proposed an ontology-based fault diagnosis method for loaders, addressing complex fault diagnosis knowledge. The technique incorporates ontology, Case-Based Reasoning (CBR), and Rule-Based Reasoning (RBR) to achieve effective and accurate fault diagnoses, validated through a case study.

Hossayni et al. [15] pioneered the development of the SemKoRe knowledge graph to collect and disseminate failure data among interconnected users. The SemKoRe maintenance process involves diagnostics to identify the causes and impacts of failures, facilitate accurate repairs, and enhance machine maintenance for future occurrences. Two critical limitations in traditional systems like CMMS and ERP, specifically issues related to sharing maintenance data across different locations and the lack of semantics in user interactions, served as the primary motivation for their research. Through the creation of a flat ontology using two types of machine parts, the authors established a system where machines generate instances of the Failure Occurrence Class containing comprehensive information about each failure, contributing to the development of a robust knowledge graph.

In coal mine equipment maintenance, Zhang et al. [16] introduced a knowledge graph system, utilizing the coal mine equipment maintenance ontology (CMEMO) to establish unified representation, integration, and sharing of knowledge. They propose a novel BERT-BiLSTM-CRF model for enhanced named entity recognition, seamlessly integrating it with the ontology through the Django application framework to build an efficient knowledge graph system.

Pedro et al. [11], on the other hand, introduced a novel information-sharing system employing linked data, ontologies, and knowledge graph technologies. Their approach utilizes RDF and SPARQL (SPARQL Protocol and RDF Query Language) for effective processing and conversion of accident case data, showcasing improved information access, retrieval, and reusability.

In a more recent work, Papadakis et al. [17] proposed a knowledge acquisition pipeline utilizing XML transformations, pattern matching, and elastic search to populate a domain-specific ontology due to the importance of automating rolling stock maintenance to enhance train reliability. They address the challenge of unstructured information in maintenance manuals, hindering computerized analysis. This results in a knowledge graph facilitating a comprehensive description of maintenance tasks, promoting operational efficiency in rail transport.

With recent advances in the field of generative AI, and LLM in particular, researchers have begun adopting LLM-based techniques for MWO data processing and classification. Stewart et al. [18] conducted the first investigation into the effectiveness of Large Language Models for failure mode classification. They concluded that fine-tuning ChatGPT 3.5 with annotated datasets significantly improves the performance of the model (F1=0.8) compared to off-the-shelf GPT 3.5 (F1=0.6) and other text classification models such as Flair (F1=0.6). The annotated dataset in their work maps MWO observations to failure modes based on ISO 14224 classes, which is related to the collection and exchange of reliability and maintenance data for equipment in the petroleum, petrochemical, and natural gas industries.

Based on the review of the current body of work, it is evident that significant progress has been made in leveraging various techniques and methodologies for constructing knowledge graphs from unstructured maintenance records. However, notable gaps remain, including the need for further exploration of semantic interoperability, scalability, and integration with emerging technologies such as generative AI as well as axiomatic ontologies. Future research should aim to bridge these gaps.

3. Proposed Framework

The proposed framework for the generation of a knowledge graph uses two semantic models, namely, a thesaurus and an ontology. The thesaurus provides lexical semantics, whereas the ontology provides logical and structural semantics. The thesaurus is initially developed semi-automatically with the support of an NLP tool and then extended automatically through a fine-tuned LLM. The thesaurus concepts are mapped to the ontology classes to enable semi-automated text translation into knowledge graphs. The generated knowledge graph, which uses

Resource Description Framework (RDF) syntax and semantics, is more available computationally compared to natural language text and can be queried or reasoned over to detect latent or recurring patterns in the data. The proposed approach adopts a human-in-the-loop (HITL) strategy since some contextual and common-sense knowledge is needed for work order text disambiguation and decomposition into a set of interrelated concepts with well-defined semantics. The level of involvement of human experts gradually diminishes as the thesaurus and the ontology become more mature and stable.

4. Maintenance Diagnostics Thesaurus

The Maintenance Diagnostics Thesaurus (MDT) is a controlled vocabulary for maintenance terms that uses the Simple Knowledge Organization System (SKOS) [19] for its syntax and semantics. The thesaurus is used to facilitate automatic detection and extraction of key maintenance concepts from the work order text. SKOS is a standard published by World Wide Web Consortium (W3C) that provides a structured framework for building controlled vocabularies such as thesauri, concept schemes, and taxonomies to be used and understood by both human and machine agents. SKOS models are considered to be lightweight ontologies as they do not have the expressivity of heavyweight, axiomatic ontologies such as OWL models. For this reason, their development cost is relatively low, and they can be readily extended in a decentralized fashion by various user communities.

The main class in a SKOS thesaurus is *skos:Concept*. In SKOS standard, a concept can be viewed as an idea, notion, or unit of thought. What constitutes a unit of thought can be subjective. However, this flexible, rather than restrictive definition of concept in SKOS standard is quite useful for building thesauri and classification schemes that are not necessarily supported by precise, logic-based semantic definitions. For example, in the domain of industrial maintenance, notions such as *failure*, *defect*, *observation*, *maintenance action*, *failed state*, and *asset* fit in the definition of skos:Concept. They may or may not point to real entities, but they represent some notions that are of significance in the maintenance domain.

Each concept (entity with unique semantics within a specific domain) in SKOS has exactly one *preferred label* (skos:prefLabel) and may have several *alternative labels* (skos:altLabel) which are the synonym terms frequently used pointing to the same concept. skos:hiddenLabel is used for capturing the frequent typos and misspellings. Narrower labels (skos:narrower) indicate a more specific form of their broader labels (skos:broader) having a hierarchical link, and the associative relationship is defined through related labels (skos:related). For example, *Fatigue Breakage* is a more specific (narrower) type of *Breakage*. *Defect,* for example, is the broader concept for *Dent* or *Crack*. *Engine Stalling* can be made *skos:related* to *Engine* since the former is a failure mode for the latter. The lexical labels (preferred, alternative, and hidden) of a concept are the primary tokens used for detecting the occurrence of a concept in a text.

The MDT captures a set of vocabulary often used by technicians when documenting their observations during the diagnosis process. Although the work orders usually contain information about the corrective actions as well, the current scope of the MDT is limited to the problem (failure) descriptions only. MDT concepts are categorized under seven concept groups or schemes, namely, Action, Artifact, Condition, Event, Function, Material Substance, and Property. The lower-level concepts under each scheme are collected through tagging relevant terms in an experimental dataset. The experimental dataset is collected from the CMMS of a construction equipment manufacturer. The MDT can be exported in the RDF-JSON data interchange standard and shared across multiple platforms that are compatible with SKOS standards [8].

Fig 1 shows examples of the concepts under the *Condition* concept scheme. These concepts are often used by maintenance technicians to describe the observed condition of an asset (machine or equipment). *Insufficient Torque*, for example, is a narrower concept for *Degraded Functioning* that itself is a narrower concept for *Nonconforming Condition*. Another utility of the MDT is capturing the alternative terms that are often used to refer to the same concept. For example, as shown in Figure 2, *hydraulic leak*, *lube leak*, *hyd leak,* and *oil leak* are synonym terms that are used for labeling the same concept. Using SKOS properties *preferedLabel* and *altLabel*, it is possible to select one preferred label for a given concept and assign multiple alternative labels as needed.

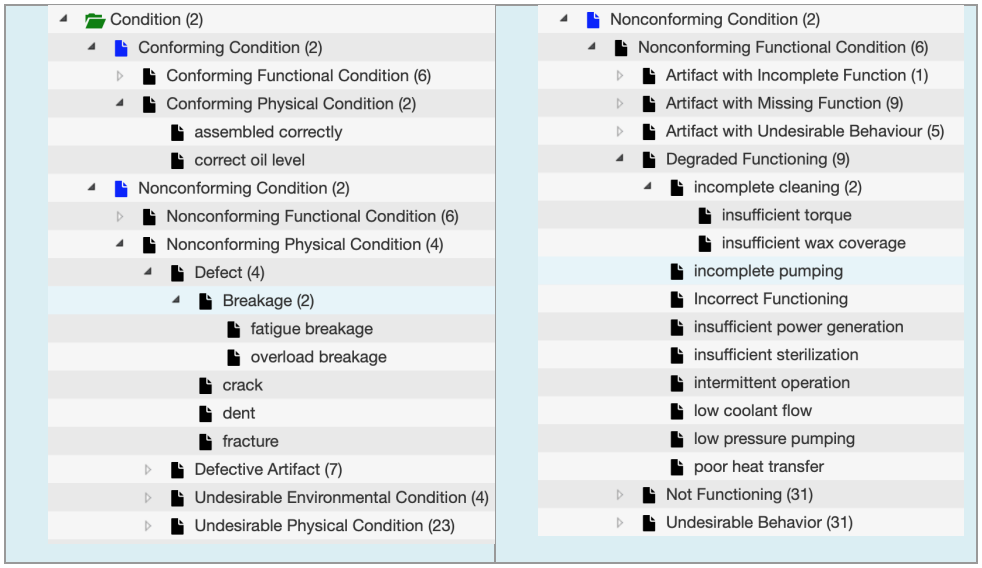

Fig 1. Examples of concepts under the "Condition" concept scheme.

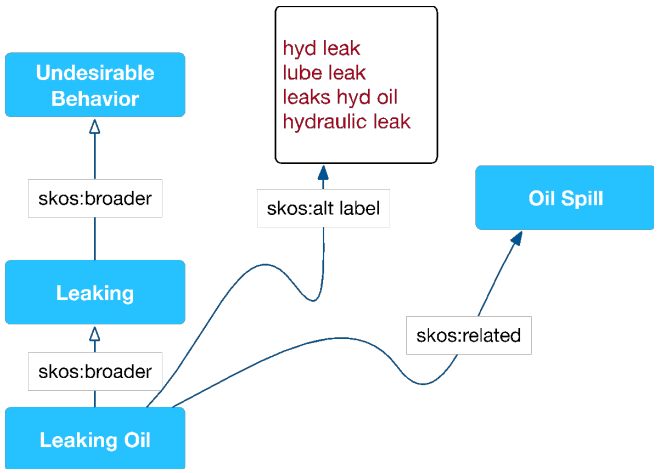

Fig 2. Broader and Related concepts for `Leaking Oil` along with its alternative labels

The development and curation of the thesaurus is facilitated by two tools, namely, Nestor [20] and SKOS Tools [19]. Nestor is a tool that supports structured data extraction from Maintenance Work Orders (MWO). Nestor uses various NLP and text analytics methods in the background. Nestor toolkit is used to automatically extract and manually classify the tokens within the text under a few broad categories such as *item (component, equipment), problem*, and *solution*. SKOS Tool is a web-based toolkit developed internally for creating and managing SKOS thesauri. One of the features of SKOS Tool is that it enables the user to upload the text (cut & paste or file upload), and then select the terms and phrases of interest and classify them under the appropriate broader concepts in an interactive fashion. In this way, the thesaurus is extended incrementally.

4.1. Nestor for preliminary feature extraction from MWOs

Nestor toolkit is a part of the Knowledge Extraction and Application for Smart Manufacturing (KEA) project, within the Systems Integration Division at the National Institute of Standards and Technology(NIST). Nestor was developed with the objective of contextualizing and adapting NLP models to technical text that often contains specialized vocabulary with overloaded meanings and jargon. Off-the-shelf NLP systems with limited context sensitivity often fail to efficiently parse through technical texts such as maintenance work orders [21].

Analyzing large amounts of MWOs manually can be a daunting task since the short texts provided in MWOs are usually full of technical terms and they are noisy and ambiguous. In this work, the Nestor toolkit is used for preliminary analysis of work orders and extraction of the key features that can be later imported into the SKOS thesaurus. The main steps in the workflow of Nestor include importing data (in CSV format), cleaning data, tagging and classifying the extracted words using Nestor UI, and exporting an annotated CSV file that includes the key terms, their class code, their related tags. The Nestor UI is depicted in Fig 3. This interface shows single word (1–Gram) analysis but Nestor is capable of multi-word analysis as well.

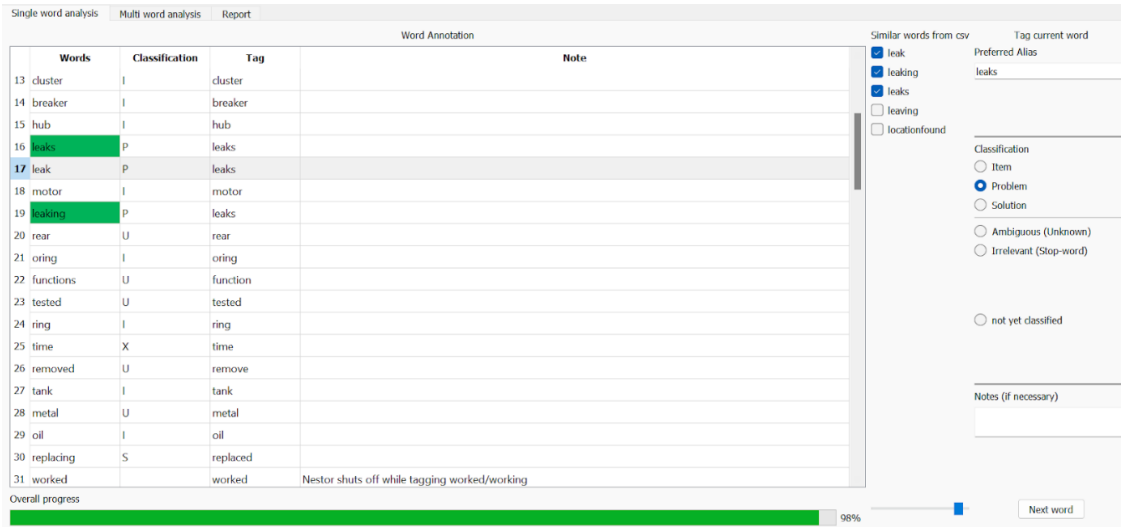

Fig 3. Partial View of Nestor Tool used for Data Tagging.

Nestor receives the MWO raw text in CSV format as the input. The user has the option of selecting the columns that need to be analyzed. Nestor then automatically identifies the terms or phrases that might be of significance for further analysis within the selected columns. In the single-word analysis, Nestor uses three primary categories (Item: I, Problem: P, and Solution: S) and two auxiliary categories (Ambiguous: U and Irrelevant: X). In the multi-word analysis, the tool uses Problem Item (PI) and Solution Item (SI) as the available categories. An example of these classifications is given below in Table 1. Once the words have been determined, the word is manually classified by the user under the appropriate category by selecting the respective code (I, P, S, U, and X) . Each word is also tagged by a preferred alias. For example, the word ‘removed’ is tagged by ‘remove’ as its preferred alias that reduces it to its root form. Nestor has the ability to find similar words that can be tagged and classified together. For example, the words *leaks*, *leak*, *leaking* are synonyms that are classified as a Problem (P) and tagged by ‘leaks’1 as the *preferred alias*. Once tagging is concluded, Nestor generates a CSV file with annotations that is used as the input for the SKOS Tool. The Entity Extractor module of the SKOS Tool uses the annotated file as the input and further classifies the words under Item (I), Problem (P), and Solution (S) categories under more refined and detailed classifications provided by MCT taxonomies.

Table 1. Test Maintenance Raw Text Input and Subsequent Outputs Identified by Nestor

Raw Text	HYDRAULIC GENERATOR IS LEAKING OUT SHAFT SEAL. SHAFT SEAL FAILURE. REQUESTED SERVICE. P/N 89065GT. REPLACED SEAL					
Item (I)	**Problem (P)**	**Solution (S)**	**Problem Item (PI)**	**Solution Item (SI)**	**Ambiguous**	**Irrelevant**
generator, seal, shaft	leaking, failure	replaced	generator leaking, seal failure	replaced seal	P/N 89065GT	Requested, service

4.2. SKOS Tool for concept extraction and classification

The output of Nestor is an annotated CSV file that contains terms that are classified under the broad categories of item, problem, solution, problem item, and solution item. However, the MDT contains more detailed taxonomies. For example, an *item* can be an *Asset*, a *Component*, or a *Functional unit*. A term that is classified as a *problem* by Nestor can be further classified under *Degraded Functioning*, *Not Functioning, Undesirable Behavior,* and so on. SKOS Tool can be used for a more granular classification of the concepts identified by Nestor. The *Term Selector* function of SKOS Tool enables the user to upload the output of Nestor and then place the detected concept under appropriate broader concepts in MDT. Alternatively, the user can also directly insert the MWO short text (problem description) into the provided text box and then select the concepts and add them to the thesaurus. Figure 4 shows the Term Selector user interface.

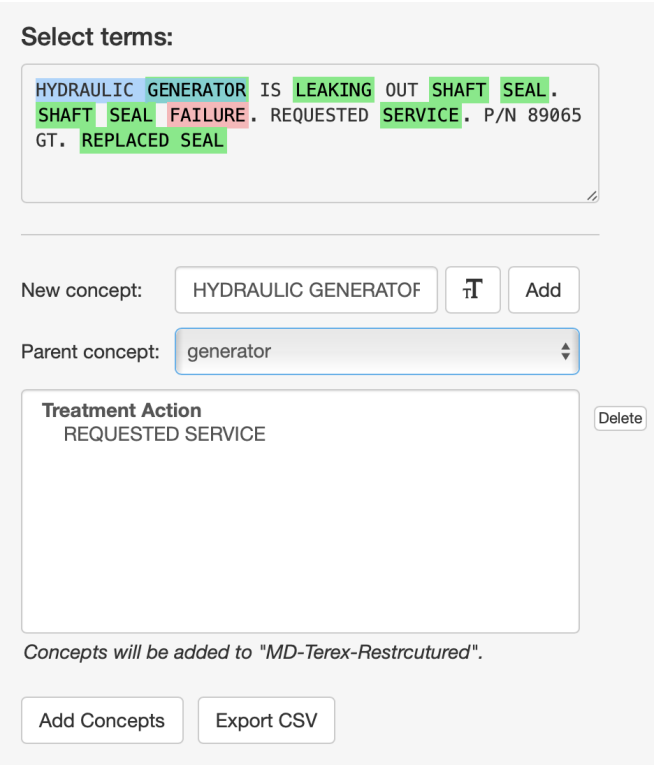

Fig 4. SKOS Tool Term Selector. The user can select and classify concepts under appropriate top concepts.

After the MWO text is inserted, then the tool automatically detects the existing concepts in the thesaurus through either their preferred label (highlighted in green) or their alternative label (highlighted in pink). In the example text in Figure 4, the concept *generator* already exists in thesaurus. However, *hydraulic generator* is not an existing concept. The user can highlight *hydraulic generator* , as a new candidate concept, identify the appropriate broader concept from the parent concept drop-down menu (*generator* in this case) manually, and add the new concept to the thesaurus. The top-level structure of the thesaurus is fairly stable, but extension and branching happen at the lower levels. Different companies can have their own MDT populated with concepts pertaining to their own assets and failure types and modes. The top-level concepts in MDT, however, are generic enough that can be applied to a wide range of assets and operations, but the lower-level concepts can be specialized to meet specific needs of different industries or companies.

4.3. Thesaurus Extension Supported by LLM

The extension of the thesaurus using Nestor and the SKOS Tool requires human supervision, particularly with respect to concept classification. As the size of the maintenance dataset grows, manual classification becomes tedious and time-consuming. To automate the concept classification process, a procedure supported by LLM is introduced in this section. We employ the controlled vocabulary provided by MDT to fine-tune the GPT-3.5 Turbo models [22]. The process begins with extracting concepts and their parent concepts from the thesaurus to construct the MDT dataset. The alternative labels of the concepts are also included in the dataset. This dataset is then split following an 8:1:1 ratio into training, validation, and test sets to ensure a comprehensive evaluation framework. The GPT-3.5 Turbo models are fine-tuned using the training and validation data, focusing on adapting the models to the specific nuances and requirements of the MDT dataset. Evaluation of all models, including the base and fine-tuned versions, is conducted on the test data, employing Precision, Recall, F1 Score, and Accuracy as key metrics. Given the imbalanced nature of the classes within the dataset, both Macro and Weighted Averages are calculated for the first three metrics to provide a more nuanced understanding of model performance across diverse class distributions.

In Table 2, the performance of fine-tuned models is compared with their base counterparts, as well as with BART-large-mnli [23], a zero-shot text classification model. The comparison between the base and fine-tuned models of GPT-3.5 Turbo, alongside BART-large-mnli, on the MDT dataset reveals that fine-tuning significantly enhances model performance in text classification tasks. The fine-tuned GPT-3.5-turbo-1106 model, in particular, demonstrates superior Precision, Recall, F1 Score, and Accuracy, highlighting the effectiveness of model customization to MWO data. The fine-tuned model can suggest the appropriate broader concept from the parent concept list. The fine-tuned model enhances the capability to accurately suggest the most relevant broader concept from the predefined parent concept list, thereby contributing to the automatic expansion of the MDT.

With the classification support provided by LLM, the output of Nestor can be directly used as the input to LLM where the detected terms can be classified under specific broader concept withing the thesaurus.

Table 2. Comparison of Different Models for Text Classification on MWO Data

Model	**Precision**		**Recall**		**F1**		**Accuracy**
	Macro Avg	Weighted Avg	Macro Avg	Weighted Avg	Macro Avg	Weighted Avg	
GPT-3.5-turbo-0613	0.3571	0.2353	0.3571	0.2353	0.3571	0.2353	0.2353
GPT-3.5-turbo-1106	0.1905	0.3137	0.2000	0.1765	0.1531	0.1975	0.1765
BART-large-mnli	0.2917	0.2353	0.2917	0.2353	0.2917	0.2353	0.2353

Fine-tuned GPT-3.5-turbo-0613	0.7200	0.6235	0.8000	0.7647	0.7500	0.6765	0.7647
Fine-tuned GPT-3.5-turbo-1106	0.8333	0.7647	0.8333	0.7647	0.8333	0.7647	0.7647

5. Work Order Ontology (WOO)

Work Order Ontology (WOO) is an OWL ontology that can be used for the formal representation of entities and their relationships in the maintenance domain. The main objective of WOO is to provide the main constructs (classes and properties) needed for generating RDF knowledge graphs from collections of maintenance work orders. Providing a set of primitive classes and relationships is the minimum requirement for construction of an RDF knowledge graph. However, the true value of a semantic knowledge graph can be realized when it is aligned with a formal, axiomatic ontology. When an ontology is applied atop a graph, reasoners serve to validate the accuracy and comprehensiveness of the data, while also facilitating the inference of new facts from the explicit ones. Moreover, this structure enables more efficient traversal and search operations within the graph.

The first step in developing ontologies is to identify a set of competency questions (CQ) [24]. Competency questions are often determined at the early stages of the ontology development process to serve as a set of requirements for the ontology and also determine the scope of the ontology. Some of the competency questions that motivated the development of WOO ontology are listed below:

- What are the causes of different undesirable behaviors of this compressor?
- What are the observed maintenance states of this machine during the past four days?
- What is the most frequent type of defect that has caused failure events on this asset?
- Which artifacts (machine or equipment) have demonstrated degraded functioning caused by overheating?
- What are the functional units that have been in a defunct state at some time within the past 24 hours?

WOO uses Basic Formal Ontology (BFO) [25] as its top-level, or foundational, ontology. However, WOO only uses a subset of BFO terms and, therefore, it does not import BFO entirely. Using a top-level ontology facilitates ontology reuse and also provides a logical framework for ontology development that is consistent with established philosophical theories. WOO uses the ontology development method and procedure recommended by the Industrial Ontologies Foundry (IOF) [26]. Accordingly, each class in WOO has a natural language and formal definition (in First-Order Logic or FOL) to enable unambiguous human-to-human and machine-to-machine communication. The semi-formal definition provides a bridge between natural language and formal definitions. Those definitions are provided below for *Defective Artifact* and *Unit With Leak Failure* classes as examples:

Defective Artifact:
Natural Language Definition: an artifact that is bearer of one or more defects

Semi-formal Definition: every instance of 'defective artifact' is defined as exactly an instance of 'artifact' that is 'bearer of' some defect

First-Order Logic Definition: DefectiveArtifact(x) ↔ Artifact(x) ∧ ∃d(Defect(d) ∧ bearerOf (x,d))

Unit with Leak Failure:
Natural Language Definition: an artifact that is bearer of one or more defects

Semi-formal Natural Language Definition: every instance of 'unit with leak failure' is defined as exactly an instance of 'machine' or 'functional unit' that 'participates in' some 'leaking' or 'has part' some 'component' or 'functional unit' that 'participate in' some 'leaking'

First-Order Logic Definition: `UnitWithLeakFailure(x) ↔ (Machine(x) ∨ FunctionalUnit(x)) ∧ (∃p1(Leaking(p1) ∧ participatesIn(x,p1)) ∨ ∃y(Component(y) ∨ FunctionalUnit(y)) ∧ hasPart(x,y) ∧ ∃p2(Leaking(p2) ∧ participatesIn(y,p2)))`

Both *Defective Artifact* and Unit with *Leak Failure* are examples of *defined* classes. A defined class is a class that can be fully restricted through a set of *necessary and sufficient* conditions (or axioms). If necessary and sufficient conditions cannot be provided for a given class, then the class is treated as a *primitive* class. A primitive class may have one or more necessary or sufficient axioms or can have none. Although it is preferred to fully define all classes in an ontology, in many cases it is not possible. One reason for keeping some terms as primitive terms in the ontology is that those terms (such as process or event) are so basic that it is not possible to fully define them in a non-cyclical fashion. In general, the desired level of formality of the ontology depends on its application. For some applications, over-restricting the semantics of terms will have adverse impacts on the useability and flexibility of the ontology and also it will make reasoning and inference processes more resource-intensive specially for larger knowledge graphs with millions of nodes. An example of a primitive class in WOO is *Functional Unit*:

Functional Unit:
Natural Language Definition: an artifact that has one or more specific functions and is composed of multiple components and is intended to become, or already is, part of a larger machine or equipment

Semi-formal Natural Language Axiom: if x is a 'functional unit' then x is an 'artifact' that has some 'components' and ‘has function’ some ‘function.

First-Order Logic Axiom: `FunctionalUnit(x) → Artifact(x) ∧ (∃y(Component(y) ∧ hasPart(x,y)) ∧ (∃f(Function(f) ∧ hasFunction(x,f))`

The axiom provided for *Functional Unit* makes it *necessary* for every functional unit to have at least one component. Examples of functional units include, a motor, a pump, a cooling system of a CNC machine or a tool changer.

Table 3 Shows natural language definitions for some of the key classes in WOO. It should be noted that the WOO is not intended to serve as a reference ontology for the entire maintenance domain. Rather, it is an application ontology designed for a specific task that is to provide a semantic layer on top of MWO data.

Fig 5 shows some of the main WOO classes and their relationships.

Table 3. Natural language definition for example WOO classes

Class	NL Definition
Maintenance State	A state that holds during a temporal interval when the realizable functions and capabilities of the participating artifact, or the grade of realization of those functions and capabilities, remain unchanged.
Undesirable Behavior	An Artifact Unintended Process that causes some undesirable consequences.
Defect	An attribute, characteristics, or feature inhered in some Artifact that does not conform to the Design Specifications of the Artifact.
Artifact With Degraded Functionality	An Artifact that realizes its primary functions at a degraded level.

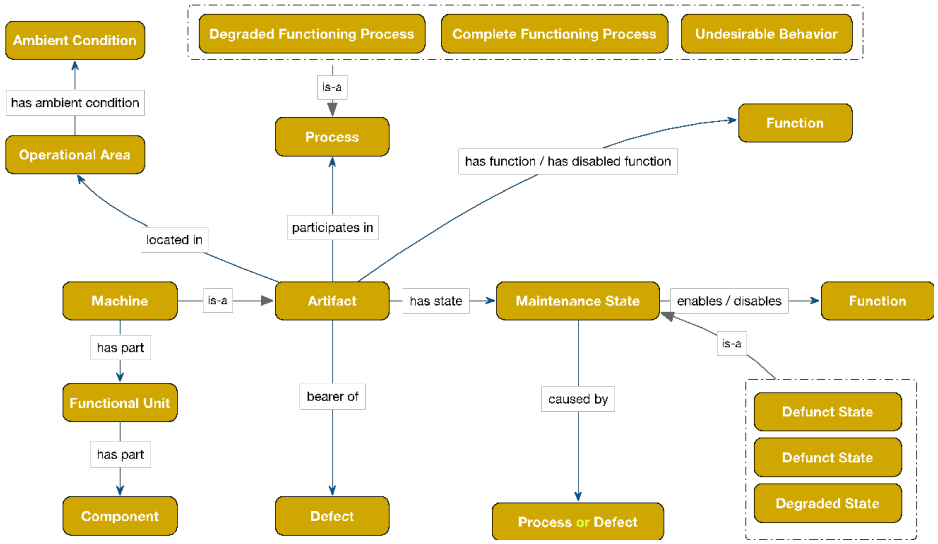

Fig 5. Class diagram showing some of the core classes of WOO and their relations

WOO is developed on Protégé and uses OWL as the ontology language. WOO currently contains 65 classes and 42 object properties. Figure 6 shows some of the top-level classes and properties of WOO. With respect to annotating the ontology and its constructs (classes and properties), Industrial Ontologies Foundry (IOF) annotation rules are adopted.

Figure 6. Top-level classes and properties of WOO

5.1. Reasoning

Reasoning is one of the key services enabled by OWL ontologies that can be conducted both at TBox (terminology component) and ABox (assertion component) levels. Reasoning at TBox typically involves checking subsumption relationships between classes and generating inferred classification based on an explicit classification and

axiomatization. TBox reasoning is also often conducted to check the logical consistency of the ontology. Reasoning in ABox (RDF Graph in this case) is the ability to calculate the set of triples that logically follow from an RDF graph and a set of rules. Although reasoning can be conducted on multiple platforms that use different reasoners and inference methods, in this work, RDFox[1] is used as the platform for reasoning and querying over the RDF triples. One advantage of RDFox is that it supports rule-based reasoning. Also, the focus of this work is primarily on ABox reasoning. The objectives of ABox reasoning are to infer new triples and verifying the consistency of instance data with respect to the ontology.

Two mechanisms are used for reasoning in this work: 1) rule-based reasoning and 2) axiomatic reasoning. RDFox uses *Datalog* language for formulating logical rules [27]. Datalog is a declarative rule language for logical inference, which means it states the desired results without prescribing the steps needed in order to achieve the results. The logical consequences of executing rules written in Datalog are materialized in RDFox as new triples in the graph. Datalog rules are essentially IF-THEN statements that add new data to the graph when certain conditions are met. The IF part of the rule is called the body (or *antecedent*) and the THEN part of the rule is called the head (or *consequent*) that is on the left side of the ":-" operator.

For example, the following rule states that if an entity (x) has a `function` (f) and x is in `defunct state` (s), then the function (f) is disabled:

```
@prefix woo: <http://infoneer.txstate.edu/ontology/MWOO/>.
[?s, woo:disables, ?f] , [?x, woo:hasDisabledFunction, ?f]:-
    [?x, woo:hasFunction, ?f] ,
    [?x, woo:hasState, ?s] ,
    [?s, rdf:type, woo:DefunctState] .
```

A Datalog rule instructs the reasoner to logically deduce some new triples from certain combinations of existing triples in the RDF graph. Rules can be executed one at a time and the inferred triples can be materialized incrementally as new data is added to the RDF graph. By extending the graph and generating a more complete dataset, more informed business decisions can be made efficiently and accurately.

The second mechanism for reasoning in RDFox is using the logic axioms that provide formal semantics of the terms within the ontology. Axioms are integral parts of a formal ontology. Once an ontology is added to the RDF data, then the ontology provides a semantic layer on top of the data. Reasoners then can use the logic axioms, in the form of necessary and/or sufficient conditions, to perform different types of reasoning including subsumption reasoning. RDFox uses the Functional Syntax for the OWL file. The following script shows the necessary and sufficient conditions for the *Unit With Leak Failure* class.

```
EquivalentClasses(woo:UnitWithLeakFailure
ObjectUnionOf(ObjectIntersectionOf(ObjectUnionOf(woo:FunctionalUnit woo:Machine)
ObjectSomeValuesFrom(woo:participatesIn woo:Leaking))
ObjectSomeValuesFrom(woo:hasPart ObjectIntersectionOf(ObjectUnionOf(woo:Component
woo:FunctionalUnit) ObjectSomeValuesFrom(woo:participatesIn woo:Leaking)))))
SubClassOf(woo:UnitWithLeakFailure woo:NonconformingArtifact)
```

Examples of rule-based and axiomatic reasoning are provided in the next section.

6. Work Order Knowledge Graph Generator Tool (KnoWo)

One of the core contributions of this work is to present a new, open-source tool, called KnoWo, for creating RDF knowledge graphs from text. KnoWo is a java-based tool for the human-assisted creation of knowledge graphs from maintenance work orders.

Before a work order can be analyzed, the user needs to upload the MD thesaurus (in RDF/JSON format) and the OWL file for the ontology. The user then enters the text related to a maintenance problem described in a work

[1] https://www.oxfordsemantic.tech/rdfox

order. The tool analyzes the text and automatically extracts the thesaurus concepts that appear in the text either through their preferred label or their alternative label (the first table in Fig 7- left). The tool automatically identifies the ontology classes related to the identified concepts. For example, *fitting* is an instance of `Component` in the provided text below. The concept extractor also provides a list of the concepts that are *related* to the concepts directly detected in the text (the second table in Fig 7- left). In the current example, *gasket* is suggested by the tool as a related concept that can be used in constructing the knowledge graph associated with the described maintenance situation. The user has the option of introducing new concepts that are not available in the thesaurus, but they are needed to build a complete graph for the selected work order (the third table in Fig 7- left: user-defined concepts). All concepts in the thesaurus are mapped to ontological classes. Therefore, each detected, or user-defined concept represents an instantiation of a class in WOO ontology. For example, *blown O-ring* is an instance of `Defective Artifact` class, and *leaking oil* is an instance of `Undesirable Behavior` class. The user can select the individuals that need to be transferred to the next step during which the selected individuals are linked together using ontological relationships.

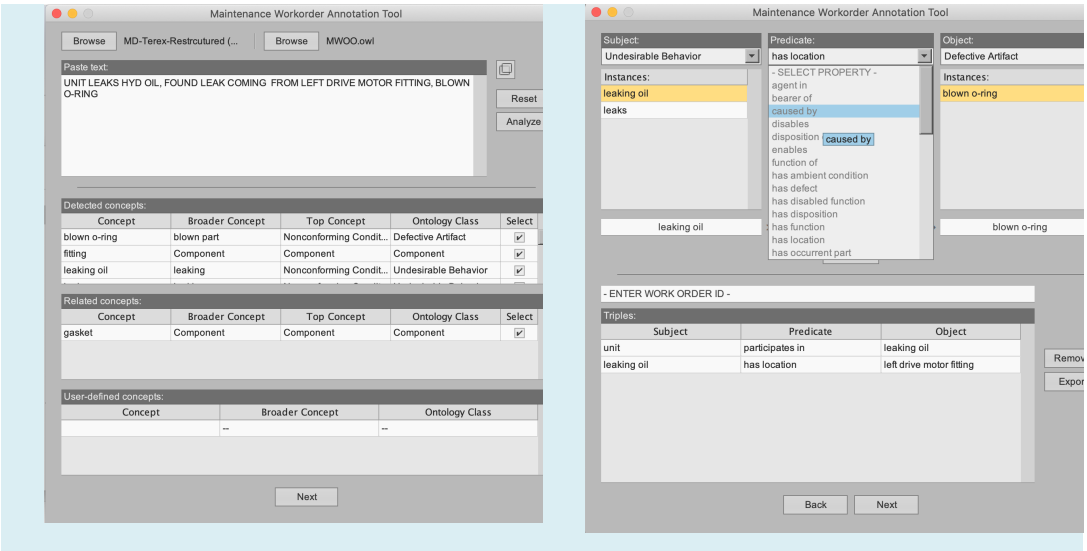

Fig 7. KnoWo interface for the first step: concept extraction and ontology instantiation (left), KnoWo interface for the second step: connecting the generated individuals using ontological classes (right)

The Subject and Object drop-down menus are populated by the classes that are represented by at least one individual. As can be seen in Fig 7 (right), the user can select the instances from the subject menu and connect them using the appropriate property to the instances under the object menu. For example, the user can specify that *leaking oil* `is caused by` *blown O-ring* based on the described observation. The permissible properties for the selected individual as the subject of the properties are highlighted and the rest are disabled to ensure triples are formed correctly. The triples are created one by one and added to the table at the bottom of the interface.

Once all necessary triples are built, the tool exports the final set of triples in turtle (.ttl) format, a file format for expressing data in the Resource Description Framework (RDF) data model. The graphs generated for each work order can be integrated into a larger graph that represents multiple work orders.

In this work we use the RDFox Console for visualizing RDF graphs and querying those graphs. Figure 8 shows the RDF graph related to the example work order as visualized in RDFox. In this figure, the green boxes represent classes, and the purple boxes represent individuals.

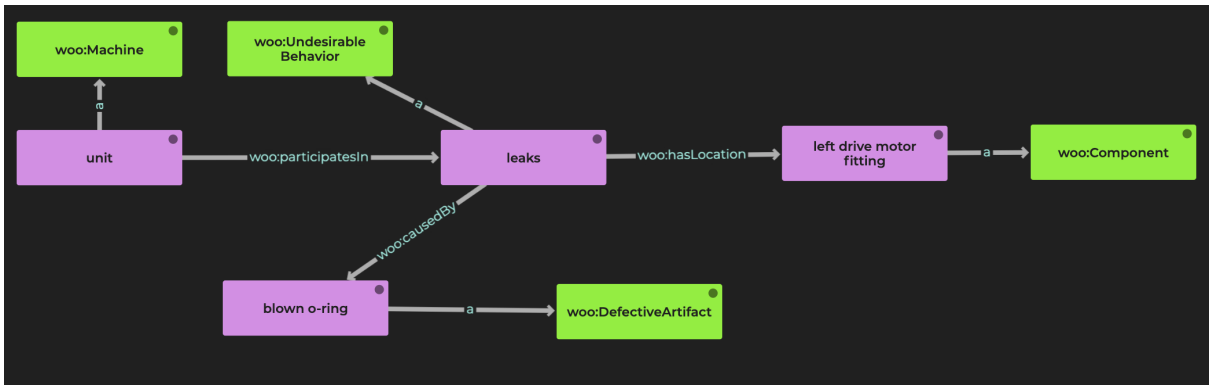

Fig 8. The final knowledge graph generated automatically based on the exported triples

Fig 9 shows the extended graph after two new triples were added to the graph automatically through reasoning. As can be seen in this figure, the reasoner has inferred that *unit* is also an instance of the class `Malfunctioning Artifact`. Also, *blown o-ring* is cause of *leaks* due since `is cause of` is the inverse property for the explicitly specified `caused by` edge.

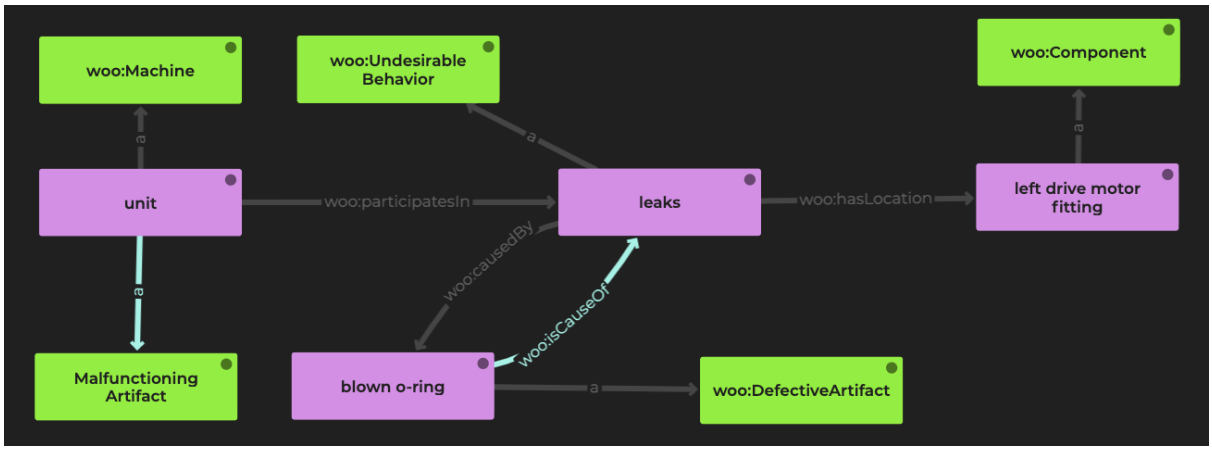

Fig 9. The extended graph after two new relationships were inferred

A pilot knowledge graph based on 100 work orders extracted from the CMMS of a construction equipment manufacturer was generated and used during the validation step. The generated graph contained 2511 triples. A domain expert in the maintenance of construction equipment was trained with the KnoWo toolkit and created the appropriate triples for each work order. All competency questions were formulated as SPARQL queries and executed against the test graph, and it was confirmed by the expert that the graph could correctly resolve those queries. The SPARQL query shown in Fig 10 returns all functional units that were in a defunct state at some time.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX woo: <http://infoneer.txstate.edu/ontology/MWOO/>

SELECT ?item ?state
WHERE {
    ?item woo:hasState ?state.
    ?state a woo:DefunctState.
    ?item a woo:FunctionalUnit.
}
```

Functional Unit	Defunct State
Winch	Not Holding Load
Generator	Not Working
Pump	Not Functioning
Fuel Pump	Inoperative
Engine	Not Functioning
Elect Motor	Not Turning On
Control Box	Not Operating
Seat Belt	Not Latching
ECM	Not Displaying

Fig 10. The example SPARQL query that returns all Functional Units that have participated in some Defunct State at some time

7. Conclusion

In this paper, a human-assisted method for the creation of RDF knowledge graphs from MWO data was proposed. A java-based tool, called KnoWo, was developed to facilitate the knowledge graph generation process. The main functions of KnoWo include extracting key concepts from MWO text (based on a list of known terms provided by a SKOS thesaurus), instantiating ontology classes according to the extracted concepts, and enabling the used to relate the generated instances using appropriate ontological relationships. One unique aspect of the proposed framework is to support the entity extraction process with a formal thesaurus that can be extended automatically using LLM.

This work particularly focused on formalizing the data related to observed symptoms and probable causes at the time of failure. However, a major component of MWO data is related to the action taken to address the cause of failure and restore an asset. In the future, the ontology and thesaurus will be extended to capture the maintenance treatment and link them to the failures and symptoms. In this way, the graph can be queried and reasoned over to retrieve or suggest potential solutions for different types of failures. Future enhancements will broaden the ontology to include maintenance actions, enriching the Work Order knowledge graph to cover the full spectrum of maintenance processes.

The integration of LLMs with authoritative knowledge models such as semantic knowledge graphs offers a novel approach by combining the power of neural language models with the ability to retrieve and synthesize information from a vast database of historical maintenance records. This allows for the generation of more nuanced and contextually relevant insights by querying the model with natural language, enhancing the analysis and decision-making processes. One future direction is to simplify querying the graph by translating natural language into SPARQL, making it more user-friendly [28]. Scaling efforts will focus on accommodating more extensive MWO data, ensuring the system's efficiency and responsiveness.

References:

[1] H. Melinda and H. Mark Tien-Wei, “Cleaning historical maintenance work order data for reliability analysis,” *Journal of Quality in Maintenance Engineering*, vol. 22, no. 2, pp. 146–163, May 2016, doi: 10.1108/JQME-04-2015-0013.

[2] M. Wienker, K. Henderson, and J. Volkerts, “The Computerized Maintenance Management System an Essential Tool for World Class Maintenance,” *Procedia Engineering*, vol. 138, no. Supplement C, pp. 413–420, Jan. 2016, doi: 10.1016/j.proeng.2016.02.100.

[3] M. Sharp, T. Sexton, and M. P. Brundage, “Toward Semi-autonomous Information Extraction for Unstructured Maintenance Data in Root Cause Analysis,” in *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing : IFIP WG 5.7 International Conference, APMS 2017, Hamburg, Germany, September 3-7, 2017, Proceedings, Part I*, H. Lödding, R. Riedel, K.-D. Thoben, G. von Cieminski, and D. Kiritsis, Eds., Cham: Springer International Publishing, 2017, pp. 425–432. doi: 10.1007/978-3-319-66923-6_50.

[4] I. Lopes *et al.*, "Requirements Specification of a Computerized Maintenance Management System – A Case Study," *Procedia CIRP*, vol. 52, no. Supplement C, pp. 268–273, Jan. 2016, doi: 10.1016/j.procir.2016.07.047.
[5] T. Sexton, M. Hodkiewicz, M. P. Brundage, and T. Smoker, "Benchmarking for keyword extraction methodologies in maintenance work orders," presented at the Proceedings of the Annual Conference of the PHM Society, 2018.
[6] M. Stewart and W. Liu, "Seq2KG: An End-to-End Neural Model for Domain Agnostic Knowledge Graph (not Text Graph) Construction from Text," *KR*, vol. 17, no. 1, pp. 748–757, Jul. 2020, doi: 10.24963/kr.2020/77.
[7] Z. Wang, X. Wang, Y. Zhang, Y. Yang, and Y. Li, "Information Extraction of Aircraft Maintenance Records for Knowledge Graph Construction," in *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, Yantai, China: IEEE, Oct. 2022, pp. 1–6. doi: 10.1109/PHM-Yantai55411.2022.9942091.
[8] Y. Ding, H. Li, F. Zhu, Z. Wang, W. Peng, and M. Xie, "A Semi-Supervised Failure Knowledge Graph Construction Method for Decision Support in Operations and Maintenance," *IEEE Trans. Ind. Inf.*, pp. 1–11, 2023, doi: 10.1109/TII.2023.3299078.
[9] Z. Peng, X. Meng, J. Du, and X. Xie, "A BERT-based Framework for Production Line Fault Maintenance Knowledge Graph Construction," in *Proceedings of the 7th International Conference on Computer Science and Application Engineering*, Virtual Event China: ACM, Oct. 2023, pp. 1–7. doi: 10.1145/3627915.3627917.
[10] M. Stewart, M. Hodkiewicz, W. Liu, and T. French, "MWO2KG and Echidna: Constructing and exploring knowledge graphs from maintenance data," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, p. 1748006X2211311, Nov. 2022, doi: 10.1177/1748006X221131128.
[11] A. Pedro, A.-T. Pham-Hang, P. T. Nguyen, and H. C. Pham, "Data-Driven Construction Safety Information Sharing System Based on Linked Data, Ontologies, and Knowledge Graph Technologies," *IJERPH*, vol. 19, no. 2, p. 794, Jan. 2022, doi: 10.3390/ijerph19020794.
[12] V. Ebrahimipour and S. Yacout, "Ontology-Based Schema to Support Maintenance Knowledge Representation With a Case Study of a Pneumatic Valve," *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 45, no. 4, pp. 702–712, Apr. 2015, doi: 10.1109/TSMC.2014.2383361.
[13] M. Ringsquandl *et al.*, "On event-driven knowledge graph completion in digital factories," in *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA: IEEE, Dec. 2017, pp. 1676–1681. doi: 10.1109/BigData.2017.8258105.
[14] F. Xu, X. Liu, W. Chen, C. Zhou, and B. Cao, "Ontology-Based Method for Fault Diagnosis of Loaders," *Sensors*, vol. 18, no. 3, p. 729, Feb. 2018, doi: 10.3390/s18030729.
[15] H. Hossayni, I. Khan, M. Aazam, A. Taleghani-Isfahani, and N. Crespi, "SemKoRe: Improving Machine Maintenance in Industrial IoT with Semantic Knowledge Graphs," *Applied Sciences*, vol. 10, no. 18, p. 6325, Sep. 2020, doi: 10.3390/app10186325.
[16] G. Zhang, X. Cao, and M. Zhang, "A Knowledge Graph System for the Maintenance of Coal Mine Equipment," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–13, Nov. 2021, doi: 10.1155/2021/2866751.
[17] E. Papadakis, T. L. McCluskey, H. Louadah, and G. Tucker, "Ontology-Guided Knowledge Graph Construction to Support Scheduling in a Train Maintenance Depot," 2023.
[18] M. Stewart, M. Hodkiewicz, and S. Li, "Large Language Models for Failure Mode Classification: An Investigation." arXiv, Sep. 15, 2023. doi: 10.48550/arXiv.2309.08181.
[19] R. Y. Farhad Ameri Kimia Zandbiglari, "SKOS Tool: A Tool for Creating Knowledge Graphs to Support Semantic Text Classification," in *APMS Advances in Production Management System*, Springer, Sep. 2020.
[20] T. B. Sexton and M. P. Brundage, "Nestor: A tool for natural language annotation of short texts," *Journal of Research of the National Institute of Standards and Technology*, vol. 124, p. 1, 2019.
[21] T. Sexton, M. P. Brundage, M. Hoffman, and K. C. Morris, "Hybrid datafication of maintenance logs from AI-assisted human tags," presented at the 2017 ieee international conference on big data (big data), IEEE, 2017, pp. 1769–1777.
[22] J. Ye *et al.*, "A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models." arXiv, Dec. 23, 2023. doi: 10.48550/arXiv.2303.10420.
[23] W. Yin, J. Hay, and D. Roth, "Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach." arXiv, Aug. 31, 2019. doi: 10.48550/arXiv.1909.00161.

[24] M. Uschold and M. Gruninger, "- Ontologies: Principles, methods and applications," - *The Knowledge Engineering Review*, vol. 11, pp. 93--136, 1996.
[25] R. Arp, B. Smith, and A. D. Spear, *Building Ontologies with Basic Formal Ontology*. The MIT Press, 2015.
[26] B. Smith *et al.*, "A First-Order Logic Formalization of the Industrial Ontologies Foundry Signature Using Basic Formal Ontology," presented at the Joint Ontology Workshop (JOWO 2019), Graz, Oct. 2019.
[27] T. Ajileye, B. Motik, and I. Horrocks, "Datalog Materialisation in Distributed RDF Stores with Dynamic Data Exchange," in *The Semantic Web – ISWC 2019*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 21–37. doi: 10.1007/978-3-030-30793-6_2.
[28] S. Yang, M. Teng, X. Dong, and F. Bo, "LLM-Based SPARQL Generation with Selected Schema from Large Scale Knowledge Base," in *Knowledge Graph and Semantic Computing: Knowledge Graph Empowers Artificial General Intelligence*, H. Wang, X. Han, M. Liu, G. Cheng, Y. Liu, and N. Zhang, Eds., Singapore: Springer Nature Singapore, 2023, pp. 304–316.