OPERATIONS RESEARCH CENTER
United States Military Academy
West Point, New York 10996

15 June 2020

Technical Report
USMA 2020 Corrosion Data Analysis Study: Estimating MADW Object Identification Accuracy and Corrosion Algorithm Improvement

Prepared By
MAJ John Case
Department of Systems Engineering
United States Military Academy
West Point, NY

Prepared For
Office of Corrosion Policy and Oversight,
Under-Secretary of Defense for Acquisition and Sustainment
Alexandria, VA

ORCEN-TR-2001

Abstract

Corrosion is estimated to account for over 20 billion dollars' worth of maintenance parts and labor each year in the Department of Defense. The Maintenance and Availability Data Warehouse (MADW) was developed to improve understanding of the corrosion problem, seeking to inform corrosion policy and funding decisions at the highest levels. Critical to our understanding of the corrosion problem is identification of the object receiving maintenance in each record, as well as identification of corrosion-related maintenance and supply transactions. In most cases, the unstructured text entries of the MADW provide the most information for these tasks. The purpose of this research is two-fold: to estimate the accuracy of the MADW objects identification algorithm and to identify ways to improve the corrosion tagging algorithm used in the MADW through use of Natural Language Processing (NLP) unsupervised and supervised deep learning algorithms. The first implication of this research is that expert systems need not be supplanted by deep learning methods at the expense of interpretability, instead, expert systems can be updated and improved by deep learning. Secondly, improving the algorithms that run on the MADW data is beneficial to improving the quality of insights gained from the data. Finally, this research identifies that a key future research area is improvement of data quality through improved data collection and cleaning techniques.

Key Words: Deep Learning; Natural Language Processing; Artificial Intelligence; Expert Systems; Corrosion; Data Analysis, Record Linkage

This Page Intentionally Left Blank

About Us

Established in 1988 inside the Department of Systems Engineering (DSE), the Operations Research Center (ORCEN) provides USMA with a dedicated analytical capability that routinely tackles problems of national significance for the Army and the Department of Defense (DoD). Leveraging the systems engineering and mathematical modeling skills of a small cadre of full-time analysts, the ORCEN executes an annual research portfolio of six to eight reimbursable projects across varied domains, including (but not limited to) operations assessment, human resource management, geospatial modeling, maintenance process improvement, and force structure decisions.

Immaterial of the domain, when the ORCEN analyst's abilities and the project sponsor's needs properly align, history has shown that the results can be quite good. Specifically, since 2005 the ORCEN has won the Dr. Wilbur B. Payne Memorial Award for Excellence in Analysis four times – an annual award, given under the signature of the Secretary of the Army, which recognizes the best operations research work that makes a significant contribution to the force. In addition to providing invaluable insights to their sponsors, ORCEN projects enrich cadet education, professionally develop Operations Research / Systems Analysis (ORSA) faculty, integrate emerging technologies into the Academic Program, and sustain ties between the Academy and the Army.

Beyond executing year-long, faculty-led research projects, the ORCEN also manages DSE's Academic Individual Advanced Development (AIAD or summer internship) program, where cadets are placed inside Army, DoD, industry, and academic organizations around the world for three to four weeks between May and August. This program broadens the cadets' perspectives, and it provides an opportunity for them to apply what they have learned in the classroom to the real-world. Cadet interns are typically rising juniors or seniors majoring in one of DSE's four majors: Systems Engineering, Engineering Management, Systems Decision Sciences, or Operations Research, and they work on an appropriately scoped project under the mentorship of the sponsoring organization. Without exception, the benefits to both the cadets and the sponsors are substantial. As one organization recently put it, "Not only did the internship program allow cadets to learn about acquisition and the processes we manage, but it also provided new ideas, thoughts, and creative concepts for us."

For more information regarding the ORCEN's reimbursable research or AIAD programs please contact:

Operations Research Center
Attn: LTC James Schreiner, PhD, Director
Department of Systems Engineering
Mahan Hall, Bldg. 752
West Point, NY 10996
james.schreiner@westpoint.edu
845-667-2685

This Page Intentionally Left Blank

Table of Contents

List of Figures

Acronyms

AI	Artificial Intelligence
Ao	Operational Availability
ASB	Aviation Support Battalion
BCT	Brigade Combat Team
CNN	Convolutional Neural Network
CPC	Corrosion Prevention and Control
DA	Department of the Army
DoD	Department of Defense
FEDLOG	Federal Logistics database
FY	Fiscal Year
GCSS	Global Combat Support System
GLoVE	Global Vectors of Word Representation
IPT	Integrated Product Team
LMI	Logistics Management Institute
LRC	Logistics and Readiness Center
LSTM	Long Short-Term Memory (unit)
MADW	Maintenance and Availability Warehouse
ML	Machine Learning
OCONUS	Outside the Continental United States
OEM	Original Equipment Manufacturers
ORCEN	Operations Research Center
OSD	Office of the Secretary of Defense
RNN	Recurrent Neural Network
SME	Subject Matter Expert
TAT	Turnaround Time
USD	Under-Secretary of Defense
USMA	United States Military Academy

Chapter 1
Introduction and Organization

1.1. Introduction

This report summarizes the work conducted by MAJ John Case in support of the Corrosion Policy and Oversight (CPO) directorate, Office of the Under-Secretary of Defense (Acquisition and Sustainment) from June 2019 to June 2020. The analysis in this technical report was enabled by the cooperation and assistance of the Logistics Management Institute (LMI), CPO's primary consulting firm. This research and its findings would not have been possible without the assistance of LMI.

1.2. Project Overview

This study supports the CPO's effort to understand the impacts of corrosion and how corrosion can be monitored and understood across the Joint Force. The first objective was to validate the MADW's accuracy through comparison of non-MADW data sources to MADW maintenance records. Specifically, the MADW algorithm that identifies what object is receiving the maintenance action in each record. Validating the accuracy of the MADW is important because understanding the accuracy of the algorithms can give CPO an understanding of how heavily to rely on insights derived from MADW data.

The second effort was to improve the MADW corrosion algorithm used to identify whether the maintenance action performed was related to corrosion or not. The research seeks to improve the corrosion algorithm by applying natural language processing (NLP) algorithms to the corpus of maintenance free text entries. Updating and improving the corrosion algorithm is important because improvements will affect both the cost and availability measures of the effects of corrosion, which were primary reasons for the MADW's development.

1.3. Organization

The remainder of this paper is organized as follows.

- Chapter 2 provides background on the problem and the currently used data management system.
- Chapter 3 explains the approach and methodology for assessing the object identification algorithm and modeling the corrosion record tagging.
- Chapter 4 provides the analysis findings and modeling results.
- Chapter 5 provides the impact and recommendations resulting from the study and possible areas for future work.
- Appendix A contains the references used in the conduct of the research.
- Appendix B provides access to the code developed in the research.
- Appendix C contains the full list of possible corrosion search words for potential inclusion in the MADW expert system.

Chapter 2
Background

2.1. Introduction

Corrosion presents a major challenge to system and equipment readiness across the Department of Defense (DoD). Corrosion is cited as having three main negative effects on the DoD: safety, availability, and cost. Safety concerns can be shown in corrosion-related structural cracking that has resulted in catastrophic failure and corroded electrical contacts have contributed to fires and electrocutions. Corrosion affects the availability of equipment when equipment becomes unavailable due to corrosion deficiencies.

According to a December 2003 DoD report to Congress, corrosion is defined as "the deterioration of a material or its properties due to a reaction of that material with its chemical [and physical] environment." The Government Accountability Office estimates DoD corrosion costs at $10 Billion to $20 Billion per year. In addition to the life-cycle cost implications, corrosion also reduces structural integrity, decreases Operational Availability (Ao) of equipment and facilities, and if left untreated, potentially results in materiel and possible system failure (U.S. Government Accountability Office, 2003).

Defense Acquisition's estimates are actually worse than GAO. They estimated that approximately 25% of all equipment and facility maintenance cost is attributable to corrosion, costing the DoD more than $20 billion annually (Defense Acquisition University, 2019). According to the Office of the Secretary of Defense (OSD) estimates, approximately 30% of current DoD corrosion costs could be avoided through investment in sustainment, design, and manufacture and other preventative measures such as paint and avoidance of dissimilar metals. Corrosion control mitigates the effects of corrosion through the use of sustaining engineering, product improvements, modifications, and design upgrades, as well as system monitoring, inspection, and maintenance practices (OSD, 2019).

Due to the high cost of corrosion, Congress enacted Public Law 107-314 Sec 1067 [codified in 10 U.S.C. 2228], titled "Prevention and mitigation of corrosion of military infrastructure and equipment," which implemented annual reporting by Services and DoD and promotes deliberate corrosion prevention and control (CPC) strategies. These include expanded emphasis on corrosion prevention and mitigation, testing and certification for new technologies to prevent and detect corrosion, and collection and sharing of information on corrosion in government, academia and industry. This legislation gave the Under Secretary of Defense for Acquisition, Technology and Logistics (USD(AT&L)) the overall responsibility for preventing and mitigating the effects of corrosion on military equipment and infrastructure. In carrying out that responsibility, the USD(AT&L) established the Corrosion Prevention and Control Integrated Product Team (CPC IPT), a cross-functional team of personnel from the military services and private industry. (Herzberg, 2015)

2.2. MADW Overview

The Maintenance and Availability Data Warehouse (MADW) is a data repository and management system designed by LMI to assess the effect of corrosion on the cost and availability of weapons systems. The intent for the MADW is to answer the question, "are we winning or losing in maintenance," however, it also answers a multitude of questions regarding maintenance cost and equipment availability across the services. To do this, the MADW targets measures of effectiveness and efficiency in terms of two corrosion-related categories:

1. Operational availability (Ao) of equipment, measured in terms of non-available days due to corrosion.
2. Cost per day of availability (C/DA), measured in terms of cost due to corrosion per day of equipment availability.

This is all possible because of LMI's work generating the MADW from data sources across the Services. They have consolidated over 1.5 billion maintenance records in total as of 2017 and likely near 2 billion at the time of this publication. To do this, the MADW draws from over 45 different DoD data sources to provide a comprehensive data repository for maintenance and supply actions. The MADW includes over 1.1 billion task-level transactional records and over 500 million supply or materials requisition records. This robust analytical capability and database provides availability, cost, inventory, and transactional data on every weapons system and readiness reportable piece of equipment within the U.S. DoD. The MADW identifies the maintenance action and object receiving the action for each maintenance record, depicted in Figure 2.1.

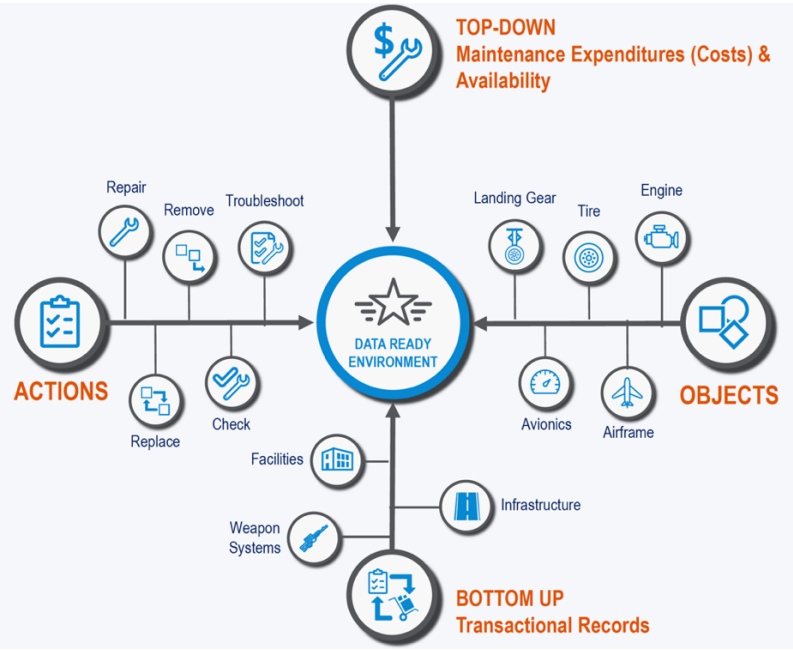

Figure 2.1: MADW Data Source (Figure Courtesy of LMI)

This research will focus on Army maintenance data. The MADW combines data from two levels of Army sources, the depot and OEM level, as well as the unit level. The depot and OEM levels are considered a "top-down" resource, while the unit level is considered a "bottom-up" resource.

2.3. Army Maintenance Data Collection Process

It is important to note that the MADW imports data differently for each Service, since each Service collects maintenance data differently. Additionally, within each service, different weapons platforms collect maintenance data differently. For example, Army units with primarily ground systems like tanks and wheeled vehicles collect maintenance data differently than Army aviation units. It is instructive to show how the maintenance data is collected from Army bottom-up resources. In Army ground systems units, this is the maintenance process:

1. Soldiers, the equipment operators, conduct preventative maintenance checks and services (PMCS) on the motorpool line in their units and record faults on DA5988's, the Equipment Maintenance and Inspection Worksheet.
2. The operators then get a mechanic, also usually a Soldier, to verify or correct faults from the DA5988. If the maintenance work requires parts to be ordered or significant labor, this is recorded on a DA2405, the Maintenance Management Record (MMR). The DA2405 (or 5989E in its electronic version) is used at both unit and support levels used to record all work requests received and handled by maintenance activities. It is supposed to capture cost of parts, hours of labor, descriptions of the word completed, and the objects that received the maintenance. The maintenance actions taken are summarized by the mechanic directly on the DA2405 for later digitizing of the document or are dictated to a clerk for input into the electronic version of the MMR
3. Parts are ordered and when on-hand, the work is completed and logged on the DA2405, closing out the maintenance action.
4. The digitized data from the DA2405 is uploaded through the Global Combat Support System – Army (GCSS-A), which is the data repository for maintenance and supply chain actions in the Army.
5. The MADW then imports the data from GCSS-A.

The maintenance data collection forms (DA2405 or DA2408-13-1) are filled out by hand or manually in a computer by Soldiers and maintainers, taking free text input data. Free text data is unstructured and highly variable, but often includes the most important information recorded on the maintenance record. Additionally, in the name of efficiency at the unit level, it is typically a clerk that inputs maintenance data as recorded on the maintenance forms or as dictated by maintainers, meaning that the person who performed the maintenance or observed the work is not necessarily inputting the descriptions into the data system. As a result, many details can be lost in the data collection process, as well as introduce shorthand, acronyms, and slang terms into the data system.

Army aviation units have an improved data collection process over Army ground units. In general, aviation units have more experienced and trained maintainers that specialize in aircraft maintenance compared to their general-purpose ground maintainer counterparts. Most significantly, the aviation industry is highly regulated, with increased logging, maintenance, and safety requirements, and as a result the data automation efforts have been a focal point for much longer than in Army ground units. In addition to the Army ground maintenance procedures, aviation units record all airframe usage on the DA2408-12, which is the Aviator's Flight Record logbook. Any time a part is removed, repaired, installed, or otherwise gained or lost from an aircraft, the action is recorded on the DA2410, the Component Removal/Repair/Install/Gain/Loss Record. Aviation units also use the DA2408-13-1 instead of the DA2405, but it serves the same purpose. The MADW primarily imports maintenance data from DA2408-13-1 records, but also has access to DA2408-12 and DA2410 records. Therefore, the bulk of MADW Army aviation data still comes from the free text entry portions of the DA2408-13-1 form.

Because Army aviation maintenance data tends to be more complete and more accurate than ground unit maintenance data, this study used Army aviation maintenance records generated for the UH-60 Blackhawk helicopter in the forthcoming analysis.

2.4. MADW Algorithms of Interest

There are a multitude of algorithms that run over the data brought into the MADW. In this study, two algorithms were of interest: the object identification algorithm and the corrosion-tagging algorithm.

2.4.1. Object Identification Algorithm

The MADW seeks to identify the object, or functional group of system components, that receives the maintenance action for each maintenance record. For example, in a maintenance record indicates that the oil was changed in a truck, the object that received the maintenance would be the engine. Although the maintenance records are supposed to include an item National Item Identification Number (NIIN), which uniquely identifies the part or component repaired or replaced, many records erroneously do not include a NIIN or they include multiple NIINs. Even when a NIIN is present, it is desirable to know the larger functional component of the equipment receiving maintenance, like an engine or drive system.

To do object identification, LMI developed a machine learning (ML) and natural language processing (NLP) algorithm which identifies the Tier 1 Object and Tier 2 Object for each maintenance record (LMI, 2019). The Tier 1 Object is the higher-level object acted on for each maintenance record, with roughly 44 different Tier 1 objects existing in the utility helicopter MADW records, for example. Tier 1 Objects are meant to capture the major functional assemblies of the vehicle or end item receiving the maintenance. Examples of the Tier 1 Objects are shown in Table 1.

Table 1: Example of MADW Tier 1 Objects for Utility Helicopters

Tier 1 Objects (Subset of 44 Total)
'AIR CONDITIONING HEATING OR PRESSURIZING EQUIPMENT'
'ALARM OR WARNING SYSTEM'
'ARMAMENT MISSILE'
'ARMAMENT NON MISSILE'
'AUTOMATIC DIRECTIONAL CONTROL'
'BODY FRAME OR HULL DOOR'
'BODY FRAME OR HULL NOT DOOR'
'CARGO HANDLING EQUIPMENT'
'COMMUNICATION EQUIPMENT'
'COUNTERMEASURE EQUIPMENT'
'CRYPTOGRAPHIC/SECURITY EQUIPMENT'
'DATA PROCESSING EQUIPMENT'
'DRIVE SYSTEM'
'ELECTRICAL SYSTEM'
'ENGINE OR PROPULSION'
'ENVIRONMENTAL CONTROL SYSTEM'
'FIRE CONTROL EQUIPMENT'
'FIRE FIGHTING EQUIPMENT'
'FUEL SYSTEM'
'GEARBOX'
'GENERATOR'

In addition to identifying the Tier 1 Object, the MADW also identifies the subcomponent or "child" of the Tier 1 Object that received the maintenance action. This is known as the Tier 2 Object. There are accordingly many more Tier 2 Objects, including 864 of them for the utility helicopter. Examples of the Tier 2 Objects are shown in Table 2.

Table 2: Example of MADW Tier 2 Objects for Utility Helicopters

Tier 2 Objects (Subset of 864 Total)
'BORESCOPE'
'TEST EQUIPMENT'
'CABIN'
'ENTRYWAY'
'PRESSURE REGULATOR'
'OIL GAUGE'
'COMPASS'
'SUSPENSION'
'DEBRIS SHIELD'
'DISCHARGER'
'STEERING SYSTEM'
'MISCELLANEOUS COMPONENTS'
'ROOF'
'COMMUNICATIONS PANEL'
'THERMOCOUPLING'
'HILOCK'
'HYDRAULIC PUMP'
'DRAIN PLUG'
'TACTICAL DATA SYSTEM'
'CART'
'CUTTER ASSEMBLY'

Many of the MADW data fields rely on analysis of the free text entries for each record. The MADW creates a single data field which combines the text in from the fault detail, fault summary, and maintenance operation fields from the original data sources (LMI, 2019).

2.4.2. Corrosion-Tagging Algorithm

Of critical importance in understanding the impacts of corrosion on DoD maintenance activities is the identification of which maintenance actions are corrosion-related. To do this perfectly, each record would have to be evaluated by a knowledgeable maintenance professional or professionals, which is prohibitively time consuming and likely impossible to complete given the total number of records is in excess of 1.5 billion.

However, the maintenance professional is using judgement accrued over many years of experience, which can be distilled to the rules or logic that the maintenance professional would use to classify maintenance actions as corrosion-related or not. These rules and logic can be written into computer code and the classification process automated. This concept is what is called an “expert system” in the field of artificial intelligence because a machine has been programmed to complete human decision-making tasks. This is the most direct way to solve the problem of tagging billions of records in a short period of time. In order to do this in the MADW, LMI attempted to capture all of the rules and logic used by maintenance subject matter experts (SME) from across the Services and programmed this expertise into a computer system.

The algorithm development took considerable time and effort, resulting in a capability that did not previously exist, and provided valuable new insights to DoD's understanding of the corrosion problem. However, one major difficulty in this method is knowing with certainty that *all* SME rules and logic were captured in the algorithm

The LMI corrosion algorithm was built by first collecting a list of keywords from maintenance SMEs that, if found in the free text entry, indicate some proportion of corrosion-related maintenance work was completed for that maintenance record. The free text entry fields are then searched over for these corrosion keywords, and if the keyword is found, then the record is tagged with a corrosion tag, which is the keyword that was found in that record's free text. Maintenance SMEs also helped identify certain "exclusionary" or "poison" words, which would negate any corrosion keyword found in a record's text. Some examples of these keywords and exclusionary words are shown below in Table 3.

Table 3: Corrosion-Tagging Algorithm Example Keywords and Exclusionary Words

Corrosion Keywords	**Exclusionary Words**
"POWDERCOATE"	"CORROSION FREE"
"STRIPPPING"	"NOT CORROSION"
"EVALU"	"NO CORROSION"
"CRACXK"	"CONTACT/CONNECTION DEFECTIVE*"
"TI-HEADBOARD"	"INSULATION BREAKDOWN*"
"BIENNAUAL"	"TEETH BROKEN ON GEAR*"
"GRING"	"BRITTLE FRACTURE*"
"DETERMNED"	"INTERNAL FAILURE*"
"QUALTY"	"CONTAMINATION*"
"INTIAL"	"CORONA EFFECT*"
"SAND"	"CONTAMINANTS*"
"DEPOSITION"	"CRYSTALLIZED*"
"PROTRECTION"	"INTERMITTENT*"
"PROTEC"	"INSULATION*"
"COROSSIVE"	"PUNCTURED*"
"PRESERVATIONXXXS"	"EMBRITTL*"
"CL-I"	"BRITTLE*"
"PRSOV"	"CHIPPED*"
"STRIPES"	"BURRED*"
"OVERPAINT"	"CONTAM"
"IN-PROC"	"FUSED*"

The corrosion tags that currently exist in the MADW were tagged with this expert system. One can quickly see some of the challenges of using a word search system to classify records. First, if the search word "corrosion" is being searched for across all records, then all misspellings and

abbreviations of this word also need to be searched. In the current search word list, there are over 100 spelling variations of "corrosion," shown by this sub-sample from the MADW search word list, which does not include any possible abbreviations:

"COREROSION", "CORODEDXXX", "CORORADING", "CORORATION", "CORREOSION", "CORRIOSION", "CORRISION*", "CORRISOION", "CORRISSION", "CORRROISON", "CORRROSION", "CORRSOSION", "CPORROSION", "EACORRODED"

Second, new words or abbreviations may be introduced by maintainers over time, potentially not included in the search keywords. This would cause the algorithm to lose accuracy and effectiveness over time and would require it to be updated on a routine basis. Finally, it is difficult to assess accuracy of the MADW corrosion algorithm because there is no clear data set of "ground truth." Each record has to be evaluated on its own by a knowledgeable maintenance professional, which is prohibitively time consuming. Therefore, the use of other methods to classify maintenance actions or to improve the expert system is needed.

2.5. Rationale for the Study

Understanding the true financial cost of corrosion, as well as the impact on equipment and facility availability is of critical importance for decision makers in the DoD. The better that the organization understands how corrosion affects the services, both in financial terms and availability measures, will improve solution strategies for preventing and controlling corrosion in the future. The only way to understand the impact of corrosion on the Services is through data analysis of financial and maintenance data collected across the many levels of the enterprise. If insights are to be derived from data, two conditions must exist:

1. The input data must be accurate.
2. Effective algorithms must be used to glean insights from the data.

These two conditions are the objects of this study. Both are investigated with the intent of informing decision makers of their ability to rely on the data within the MADW.

Although there are over 1.5 billion records in the MADW, the usefulness of the data is dependent upon the way it was entered by a maintenance worker, Soldier, Sailor, Marine, or Airman into the system. In many cases, the most useful features in a maintenance record are the maintenance action code, which is a service code depicting the maintenance action performed, and the free text entry inputted by the maintainer at the unit level. While the maintenance action code is useful, many times it does not describe the complete work done in a maintenance action or it is not present in the data at all. The free text entry usually provides a more complete description of what was actually done in the maintenance action.

There are also many complicating factors of using the free text entry for corrosion analysis. Text data is unstructured by nature. Additionally, for the same maintenance action performed, two different maintenance workers may enter different text entries to describe what work was

performed. Broadly speaking, the free text entry field's format, spelling, grammar, and topic is highly variable, even for common weapon systems and common maintenance actions. The free text analysis presents a source of continual effort to improve how maintenance data is analyzed to produce corrosion-related insights.

2.6. Problem Statement

The MADW has been a monumental effort to bring maintenance and financial data together from across the Services. Additionally, the MADW's many algorithms running on the data provide insights into the current state of corrosion and maintenance. When decision makers are relying on the data within the MADW and the insights of the algorithms, some of the questions that arise include:

- "How accurate are the corrosion estimates?"
- "How much can I rely on this data?"
- "What is our level of confidence in this algorithm's result?"

Specifically, CPO desired to know the accuracy of the MADW's determination of the object receiving maintenance, the Tier 1 and 2 Objects. This is the study's first objective.

The second area CPO desired to have research target is if the MADW insights can be improved. Of critical importance in understanding the impacts of corrosion on DoD maintenance activities is the identification of which maintenance actions are corrosion-related. As previously discussed, LMI uses an expert system based off of keyword and exclusionary word searches in the free text entries of the maintenance records. Given this, decision makers were interested in these questions:

- What, if any, are the search words that are missing from the currently used keywords?
- How can accuracy be assessed for the MADW corrosion tags?
- Can a level of confidence be calculated for each corrosion tag?

Therefore, the second research effort is to identify ways to improve the corrosion-tagging algorithm in the MADW. This is the study's second objective.

The restated problem statement for this study is this:

Given that CPO needs to understand to what degree they can rely on the MADW data and its algorithms' insights, (Objective 1) what is the accuracy of the MADW Tier 1 and 2 Objects and (Objective 2) can the corrosion-tagging algorithm be improved to ensure that the expert system is accurate and up to date?

2.7. Literature Review

In order to address this study's problem statement, two data analysis fields are pertinent to pursuit of the two objectives:

- Objective 1: record linkage (or data matching)

- Objective 2: natural language processing (NLP)

2.7.1. Record Linkage

Record linkage, or data matching, is a field of computer science in which the task is to identify, match, and merge records that correspond to the same entities in one or several databases. The challenge is the lack of common entity identifiers in the data sources to be matched. The creates the challenge of matching records using attributes that contain only partially identifying information, due to partial availability of data, typographical variations present in the data, or changing information input over time (Christen, 2012).

The record linkage process typically requires five steps.

1. Data pre-processing ensures that the data used for record linkage are in the same format, structure, and content. This includes removing unwanted characters and words, expanding abbreviations, correcting misspellings and splitting or combining attributes.
2. Indexing ensures that a computationally tractable record linkage task can be executed. Potentially, each record from one database will need to be compared to every record in the other database to allow record similarities to be calculated. Thus, the number of record pair comparisons is quadratic in the size of the databases. Typically, the vast majority of pair-wise comparisons are between non-matches, so indexing is employed to reduce the number of comparisons to be made. Blocking is the typical indexing approach used to filter or reduce computational complexity in the comparisons.
3. Record pair comparison ensures that potential record linkages are measured in a robust manner, allowing for relative comparison to other potential matches. Typically, exact string matching cannot be employed with success and some measure of similarity is needed to identify how similar non-identical records are. Edit distance metrics are a common way to calculate similarity scores and compare potential record linkages.
4. Classification bins each record pair into a match, non-match, or potential match. Probabilistic techniques and machine learning techniques are typical approaches to classification.
5. Evaluation assesses the quality of matches identified in the classification step. Precision and recall are typical metrics to evaluate match quality. To evaluate completeness and accuracy of the matching process, some form of ground truth data is needed to identify the true matches and compare these to the algorithmically applied matches. In some cases, a human must evaluate the algorithmic matches if no ground truth data exists (Christen, 2012).

In order to assess the accuracy of the MADW, a record pair comparison method that is suited to the data is required. In the case of the MADW, the Tier 1 and 2 Objects are multi-word strings, which will be compared to platform-specific functional components, which are also multi-word strings. In general, edit distance algorithms that handle multi-word strings, also known as multi-

token algorithms, are best suited to the task. The basic Levenshtein edit distance algorithm, used in conjunction with some text pre-processing, was the best suited algorithm for MADW record matching, although more were explored in the course of this study, including Jaro-Winkler, Editext, Longest Common Substring, Bag Distance, Ratcliff-Obershelp, Monge-Elkin, and Jaccard algorithms.

The basic Levenshtein edit distance function computes the smallest number of single character insertions, deletions, and substitutions that are needed to convert the first string into the second string. This edit distance calculation is used in approximate string matching to compute similarity scores for string comparisons. The computation for Levenshtein edit distance is given by computing a matrix with the characters of string one (s_1) along the rows and string two (s_2) along the columns. Then the entries of the matrix are completed using dynamic programming with the edit distances according to each pair of characters from the two strings, given by:

$$LED_{s_1,s_2}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0 \\ min \begin{cases} LED_{s_1,s_2}(i-1,j) + 1 & \text{a deletion,} \\ LED_{s_1,s_2}(i,j-1) + 1 & \text{an insertion} \\ LED_{s_1,s_2}(i-1,j-1) + 1 & \text{a substitution} \end{cases} & \text{otherwise} \end{cases}$$

(Christen, 2012 p. 104)

The distance calculation can be normalized to a given scale, thus allowing use of a common similarity score over many comparisons of varying lengths, as is needed in the MADW assessment. This is computed on a scale of 0 to 100 with the following formula:

$$\frac{(|s_1| + |s_2|) - LED_{s_1,s_2}(i,j)}{|s_1| + |s_2|} \times 100,$$

where $|s_1|$ and $|s_2|$ are the lengths of the two sequences being compared.

A question that arises is whether a probability of a match can be computed when using edit distance functions, given a specified number of possible characters and total pattern length to be matched. In general, an exact analysis of probability of matches is only useful for fixed patterns, however empirical analysis has been fruitful in determining the probability of a match given the total number of possible characters in the text and the number of errors in the comparisons (Navarro, 2001). Bounding the probability of a match allowing for errors in the text is a useful method to avoid spending time verifying potential matches, which is shown to require polynomial time on order of m^2, where m is the total length of the strings being matched. If we let $f(m,k)$ be the probability of a match of a random pattern of length m matching a given text position with k errors or less under the Levenshtein edit distance, and let:

- $\sigma \equiv$ number of possible characters found in the text.
- $\alpha = \frac{k}{m} \equiv$ error level for which an approximate string match is found.
- $\alpha^* \equiv$ the maximum error level for which approximate string matches are found.

It can be shown that $\alpha^* > 1 - \frac{e}{\sqrt{\sigma}}$ is a conservative condition on the error level that ensures few matches. However, Navarro showed empirically that $\alpha^* \approx 1 - \frac{1.09}{\sqrt{\sigma}}$ for m in the hundreds and σ = 32. Additionally, for small m, $\alpha^* \approx 0.70$. This result assumes that all characters can occur with the same probability across the entirety of the text, although a significant assumption, is reliable in practice (Navarro, 2001). The similarity score, calculated on a scale of 0 to 100 relates to Navarro's findings by:

$$Sim\ Score = (1 - \alpha) \times 100$$

2.7.2. Natural Language Processing

NLP is a multi-disciplinary field, including areas such as computer science, computational linguistics, and artificial intelligence, which is focused on building models and applications that enable interaction between machines and the natural languages created by humans. NLP is related the field of Human-Computer Interaction (HCI), which enables humans and machines to interface in an optimal way, providing advancements in many different fields. Natural language is captured as data for processing by a computer in the form of text, audio signals, or image data. This area is of interest because natural language is generally stored in unstructured form, which is difficult for a machine to process. Natural language data is one of the most prolific forms of data on the internet; according to IBM, approximately 80% of all data on the internet is unstructured text data. This provides a significant opportunity for improving HCI and the way we process and understand text (Sarkar, 2019). Over the past decade, NLP has become tightly coupled with machine learning methods.

Machine learning is best introduced by comparison to classical programming methods, in which a computer is explicitly programmed how to complete a given task. In machine learning, the system is trained to complete the task at hand by being shown many examples relevant to the task. The machine is able to find statistical structure in the data and map the inputs to the outputs, in a way that was not dictated explicitly in computer code. Thus, through iteration of a program, the computer learns how to map inputs to outputs, essentially learning the "rules" rather than being explicitly told the "rules" through computer code. This comparison is shown in Figure 2.2.

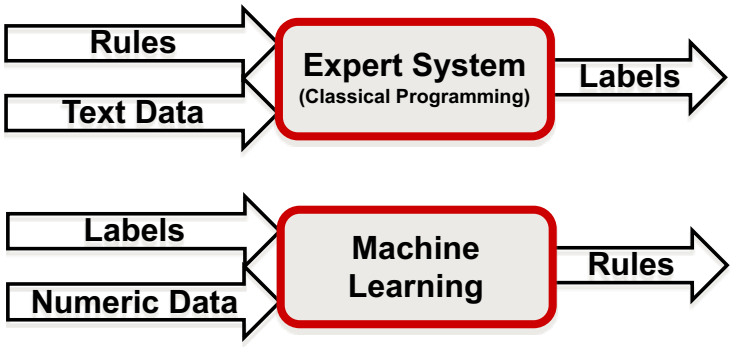

Figure 2.2: Machine Learning Compared to Classical Programming (Chollet, 2018)

At a very high level, machine learning takes input data points and examples of the expected output and learns how to map the inputs to the outputs through some error-measuring algorithm. The machine adjusts its mapping in order to reduce the error associated with its current output and the expected output in a process called learning. Thus, the computer learns useful representations of the input data such that it can produce the expected output data with minimal error. There are three main problem areas in machine learning, each with multiple subfields not discussed here:

- Supervised learning, in which the "answers" or true labels of the input data are known. The task may be regression or classification.
- Unsupervised learning, in which the "answers" or true labels of the input data are unknown. The main task is to find useful structure in the input data.
- Reinforcement learning, in which a software agent learns to take actions within its environment to achieve some objective.

For this study, a combination of unsupervised and supervised learning was most appropriate, specifically in the deep learning sub-category of machine learning. Deep learning methods are applications of machine learning algorithms that leverage successive layers of mathematical representations of the input data to realize increasingly meaningful representations. The layered representations are typically learned with neural networks, which are machine learning frameworks inspired by abstracting the functions of the brain. Each layer of a neural network performs feature extraction, engineering, and transformation of the previous layer's output, allowing the model to build a hierarchical representation of the data at different layers of abstraction (Sarkar, 2019). This framework allows the trained model to represent highly non-linear and complex relationships found in the data. A high-level view of a simple two-layer neural network is shown in Figure 2.3, which depicts how the computer learns to adjust the model weights in order to reduce its error on the model predictions.

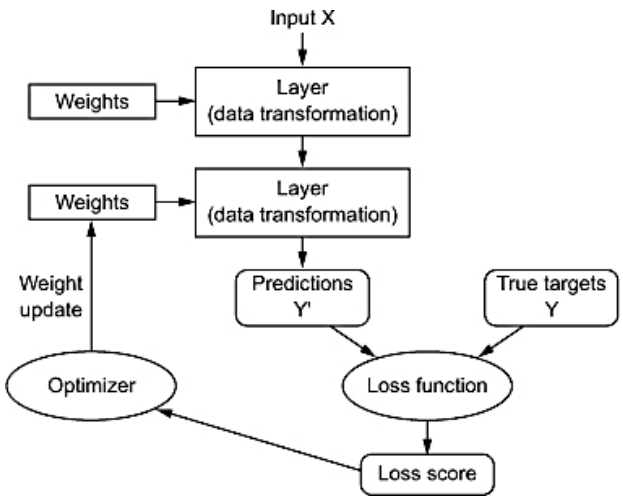

Figure 2.3: High Level Depiction of a Two-Layer Neural Network (Chollet, 2018)

Deep learning has achieved significant breakthroughs in recent years, including near-human level image classification, speech recognition, handwriting transcription, language translation, text-to-speech conversion, autonomous driving, and natural language question-answering. The state of the art in NLP is currently in deep learning methods (Chollet, 2018).

1.2.7.2 Word Embeddings in NLP

Two critical components of text classification are word and document representations and the use of language models. The task of representing words and documents is central to text classification and many other NLP tasks. In general, representation of words and documents as numeric vectors has been the most useful and intuitive ways to represent words and documents. Because computers can only process numeric data, natural language must be converted into numbers so that it can be processed by a machine. There are multiple ways to transform text into numeric vectors so that NLP algorithms can be used to process the text data:

- Tokenize text into words, then convert each word into numeric vectors
- Tokenize text into characters, then convert each character into numeric vectors
- Extract "n-grams" of words or characters, then transform each n-gram into a vector.

N-grams are groups of consecutive words or characters that appear together, thus allowing for multiple overlapping n-grams to be extracted from a single sentence or document (Chollet, 2018). There are also multiple ways to associate tokens with numeric vectors, of which two are very prominent:

- One-hot encoding, in which each token is encoded as a unique binary vector of the total size of the vocabulary.
- Token embedding, in which each token is represented as a dense floating-point vector and related in a geometric way with the other token embeddings in an embedding space.

The tasks of tokenization of natural language is computationally intensive and is executed algorithmically, thus pairing with the fields of machine learning and deep learning, where the combination of language representations and neural language models were brought together to realize significant advances in NLP (Almeida and Xexeo, 2019).

Token embedding methods can broadly be categorized into prediction-based models and count-based models. Prediction-based embeddings can best be seen in their use as the first layer of neural language models. An effective technique that applies well to this study is the negative sampling procedure developed by Mikolov et. al. in which the famous Word2Vec algorithm was developed (2013). The Skip-Gram implementation of Word2Vec is an unsupervised learning technique that generates labeled training data from the corpus of documents, such that a supervised approach can be taken to generate the dense word embeddings. The algorithm proceeds by running over the text and creating positive and negative context samples. Positive samples are pairs of a target word with its context words in the same n-gram window, while negative samples are pairs of a target word with random words drawn from the corpus. Positive samples are labeled with a "1" while negative samples labeled with a "0," in effect creating the input data and known label such that a supervised classification approach can be used to train the word embeddings that minimize the total error.

The Skip-Gram implementation of Word2Vec can be shown by an example. Context sampling is generally shown by the structure [(*target word*, *context words*), *label*]. The sample maintenance record is: "PMD INSPECTION DUE COMP IAW TM-1-1520-280-PMD" Given a context window of 4, a positive sample would be [(DUE, INSPECTION COMP IAW), 1], while a negative sample is [(DUE, ENGINE MODULE BRACKET), 0] since the context words are not found with the target word, "DUE." This is depicted in Figure 2.4, in which a context window of length four is passing over the text to be analyzed.

Figure 2.4: Depiction of Labeled Data Generation in the Skip-Gram Implementation

Also developed by Mikolov et. al. is FastText, an algorithm based upon Word2Vec, which improves the algorithm by incorporating *subword* embeddings, which are embeddings based upon character-level n-grams rather than treating each word as unique entities. Because rare words are typically not modeled well by Word2Vec, the inclusion of *subword* n-grams allows rare word *subwords* to be found in the corpus more often. Mikolov et. al. recommend that all n-grams be extracted for three to six character n-grams, which helps capture all prefixes and suffixes from the corpus (Sarkar, 2019).

Count-based methods leverage word-cooccurrence relationships in the text corpus, relying on global word-context statistics to calculate the embeddings. This is best shown in the well-known Global Vectors (GLoVe) embedding model developed by Pennington et. al. (2014). This algorithm proceeds by generating a large word-context cooccurrence matrix that represents how

often a target word occurs with its context words. Then matrix factorization is used to approximate this matrix with a word-feature matrix and a feature-context matrix, which are multiplied to produce the word-context cooccurrence matrix. The word-feature matrix gives the intended word embeddings once it has been trained successfully with stochastic gradient descent (Sarkar, 2019). The word embedding development is still a very active research area in NLP (Almeida and Xexeo, 2019).

2.2.7.2 Text Classification in NLP

Text classification is a subfield of NLP in which the primary objective is to use machines to successfully categorize or classify text documents based upon their natural language content. Documents of text are classified into pre-defined categories, which can be binary or multi-category. Text classification begins with a training set of documents and their corresponding class or category, which are the true labels. A supervised learning algorithm is employed to build a classifier, which is trained through some form of back propagation algorithm to adjust the model weights in order to minimize the cost, or total error of the model mappings. This trained model can predict the class of new and unseen documents. A supervised learning algorithm requires manually labeled training data, meaning that at some point in the process of data collection, a human had to label the documents with their true classes. Although time consuming and costly, the benefits of labeling data is significant, given that once an accurate classifier is trained, text classification of new documents is automated. Now new documents can be classified with minimal human effort or supervision.

The general text classification process includes multiple important steps to yield a successful model.

1. Prepare the data set. This entails gathering the data and labeling documents with their classes, if required.
2. Preprocess the text documents. Text pre-processing is extremely important in natural language processing tasks because it reduces the dimensionality of the data prior to applying training algorithms to the text data. Critical pre-processing steps include:
    a. Text tokenization consists of splitting text data into its smallest meaningful components, like words and symbols.
    b. Removing unnecessary tokens and stop words consists of removing words that have little or no significance, like “a,” “the,” or “and.”
    c. Expanding contractions consists of locating contractions in the text and replacing them with the full words, like replacing “couldn’t” with “could not.”
    d. Making all text uniformly lowercase and removing special characters.

e. Correcting spelling errors consists of replacing words that have character(s) in the wrong location, added erroneously, or omitted entirely with the correct spelling of the word.

f. Stemming and lemmatization consists of finding the root form, or lemma, of each word by removing affixes from the word, like replacing "cleaning" with "clean."

g. Part of speech tagging consists of labeling each word in a sentence with its proper lexical category in order to apply structure to the data.

h. Parsing or chunking consists of organizing text into language structures like phrases and sentences.

i. The data is finally split into training, validation, and testing data sets. The train and test split is dependent upon the size of the data set, but typically can be a 70:30 or 80:20 split.

3. Generate word or character embeddings. This process was described previously.

4. Train the model. The classification model parameters are adjusted in order to minimize the output error on the true classes in the training process. Depending on the size of the data set and the algorithm, this step can require minutes to weeks of runtime. In this step, hyperparameter optimization is required to tune the model parameters that are not adjusted as part of the algorithmic training process. The training process is typically iterated many times to find hyperparameter settings that help the algorithm minimize its error.

5. Evaluate the model. In this step, the testing data set is fed into the model in order to assess how the model performs with data it has not used in the training process. A well-trained model is generalized to data it has not seen in the training process; thus the test set is used to evaluate model accuracy and generalizability. In this step, the model's mis-classified documents are analyzed to understand where the model is making mistakes. If required, the text classification process is iterated to produce a better model (Sarkar, 2019). Typical metrics used to evaluate model performance on classification tasks include measures of accuracy, precision, recall, and F1 score.

Although the algorithms and hardware advances have allowed deep learning NLP advances, the programming frameworks to implement these algorithms has also improved the accessibility of deep learning. The Keras deep learning library, developed primarily by Francois Chollet, provides a highly developed and convenient programming interface to the Tensorflow backend, which is a Google-developed tensor-optimized and differentiable programming computation engine that increases the speed of training neural networks significantly. This package allows the designer to build and train neural networks in an efficient manner (Chollet et. al., 2015). Keras provides an application programming interface through Python, which was utilized in the course of this study. There are many other useful deep learning frameworks available, but Keras was selected as the most accessible and well-developed option.

2.8. Hypotheses

Given the two objectives of this study, there were multiple hypotheses made in order to focus the research efforts. These hypotheses are investigated in order to target the problem statement for the client.

2.8.1. Hypothesis 1 (Objective 1):

Although the MADW pulls data from many sources and multiple levels of the Services' maintenance processes, data sets exist that are not directly imported into the MADW. These external data sets can be used to validate the MADW Tier 1 and 2 Objects and assess the object identification algorithm's accuracy. Specifically, approximate string-matching methods can be used to identify whether or not the Tier 1 and 2 Objects are the true receiver of the maintenance action if the MADW records and external data set records are mapped to the platform's major functional components. If the MADW Tier 1 and 2 Objects are accurate, then each true maintenance action performed (external data set) should be found in the MADW. Additionally, if the MADW Tier 1 and 2 Objects are accurately mapping to the higher-level assemblies on the platform receiving maintenance, then the Tier 1 and 2 Objects should map to an assembly *up the tree* from the removed or installed part in the platform's parts tree (as captured in the external data set).

2.8.2. Hypothesis 2 (Objective 2):

Given that the keywords were compiled from subject matter experts and that free text entry is the corpus being searched over for these keywords, it is unlikely that every spelling variation and abbreviation of the search words are accounted for in the keyword index. Additionally, since the MADW is being populated with new data monthly, new phrases may be used to describe new or changing conditions in maintenance records, which are not accounted for in the keyword index. The hypothesis is that natural language processing methods can identify words that are semantically similar to the corrosion algorithm search words. Specifically, a vector space representation of all words, also known as dense word embeddings, can be trained from the corpus of maintenance text entries. This vector representation also captures the context and semantic meaning of words in the corpus. The corrosion search words can then be used to find similar words from the trained word embedding space by identifying all words that have similar semantic meaning and context, or those words with the smallest Euclidean distances from the corrosion search words.

2.8.3. Hypothesis 3 (Objective 2):

Advances in the use of deep learning algorithms in natural language processing tasks has improved dramatically in recent years. The hypothesis is that these algorithms can be used to improve upon the accuracy of tagging maintenance actions as corrosion-related or not. Specifically, neural network text classification models can be developed that provide an increased level of accuracy in tagging maintenance records.

2.9. Study Limitations

As in any data-centric problem, the study outcomes are limited by the quality of the original data. If the data collected is inaccurate or incomplete, then any analytics or modeling effort will not provide valuable insights. As previously discussed, the free text entry in the maintenance records is highly variable, with descriptions that lack detail, contain shorthand and slang, and sometimes errant information. This is a significant limitation in training a highly reliable and accurate model.

As a natural language processing problem, the maintenance text representation and corrosion classification problems are difficult because there are many irregular words. Consider the following examples of actual free text entry maintenance record descriptions:

PMD INSPECTION DUE COMP IAW TM-1-1520-280-PMD

M/R HUB TQ CHK ON ZONE 1 REQ FOR PMI 2 TQ CHK COMPLETED TQ TO 128FT LBS TW SN:2369

VG21014356705 GEN ST DSL MEP-802A VG2/LUP/I-90D/A-365D/B-720D

The first text entry only contains two English words out of eight total words. The second entry has 6 complete words out of 20 terms, where most terms are abbreviations or shorthand. The third entry lacks any complete English word. The corpus of maintenance text is almost a different dialect of English because it contains different words, sentence structure, and punctuation than standard English.

A significant limitation exists when considering use of supervised learning classification methods, because there is no “ground truth” corrosion-tagged maintenance actions. Training a model to use the free text entries to tag records as corrosion-related or not is fully based upon the expert system’s tagged records. Therefore, the models can only learn from what the existing methodology has already done. This implies that if the expert system has incorrectly tagged any maintenance record, the model will learn this incorrect mapping. As is shown in the methodology and findings chapters of this study, a data set “ground-truth” corrosion-tags were identified under Objective 1. Because this data set is not large enough to train a deep learning model, it was used as a test set of data to evaluate model performance.

Chapter 3
Approach and Methodology

3.1. General Approach

The general approaches of this study were split in support of the two broad research objectives. Under Objective 1, quantifying the accuracy of the MADW's Tier 1 and 2 Objects, the research approach was the application of record linkage and approximate string-matching techniques to identify the accuracy of MADW objects. Under Objective 2, identifying ways to improve the corrosion-tagging algorithm in the MADW, the research approach was the application of deep learning NLP methods to generate word embeddings which help identify terms related to the current search words used in the MADW corrosion-tagging algorithm. Additionally, text classification deep learning models were trained, using the previously mentioned word embeddings, to identify which words were most important in determining the corrosion tag, learned from the data rather than the expert system rules.

3.2. Research Sample

A data set of Army aviation maintenance data was provided by the U.S. Army Aviation and Missile Command as part of the Reliability, Availability, Maintainability (RAM) data curation effort in the Aviation System Assessment Program (ASAP). The data provided included approximately 44,500 DA2410 records collected from 2000 to 2019, which are generated any time a part is removed, repaired, installed, or otherwise gained or lost from an aircraft. This data was specific to the UH-60 Blackhawk helicopter platform. As part of the ASAP RAM effort, any DA2410 parts that come through Corpus Christi Army Depot (CCAD), the Army depot that conducts all aircraft maintenance, are coded as one of several categories, including deteriorated, fungus effect, moisture saturation, corroded, pitted, flaking, blistered, broken, wet, or crystallized. A sample of this data is shown in Table 4. Select Army aviation parts have been categorized by CCAD since 2000, providing human-verified data for the DA2410 UH-60 parts. The MADW has access to the originally inputted DA2410 data but does not access the ASAP-RAM categories applied by CCAD.

Table 4: ASAP-RAM DA2410 Data Sample

DATE_2410	PN_NOUN	DESCRIPTION	SYS_CAT	UIC	CITY	QTY	YEAR	NOUN	PRICE
22-Mar-17	5078T29G06/ELECTRICAL CONTROL UNIT	070-Broken	T700	W0U911	FT RUCKER	1	2017	ELECTRICAL CONTROL UNIT	$19,591.00
22-Mar-17	5130T00G01/T700-GE-701D ENGINE ASSY	170-Corroded	T700	W0H95J	KILLEEN	1	2017	T700-GE-701D ENGINE ASSY	$678,067.00
27-Mar-17	3046T17G02/O/S DRAIN VALVE ASSY	070-Broken	T700	WDUYB0	FT RILEY	1	2017	O/S DRAIN VALVE ASSY	$3,376.00
28-Mar-17	6071T27G01/ACCESSORY MODULE	117-Deteriorated	T700	W0H9AA	HUNTSVILLE	1	2017	ACCESSORY MODULE	$77,878.00
30-Mar-17	4046T52G27/FUEL CONTROL	170-Corroded	T700	W1J7AA	YAKIMA	1	2017	FUEL CONTROL	$31,618.00

In order to identify "ground truth" objects, an actual parts tree was needed for the UH-60 Blackhawk helicopter. This was provided by the Utility Helicopter project office within Program Executive Office Aviation (PEO Aviation). This data set includes 1,146 UH-60 parts in a hierarchical structure according to how the parts function and are installed on the airframe. The hierarchical nature of the UH-60 parts lends itself directly to identifying what was intended by the Tier 1 and 2 Objects in the MADW.

Table 5: UH-60 Parts Tree Data Sample

SYSTEM_ID	RFG	DESCRIPTION
__-60_	'03	Landing Gear
__-60_	'03A	Main Landing Gear (MLG) Installation
__-60_	'03A01	MLG Shock Strut LH/RH
__-60_	'03A02	MLG Brake Installation
__-60_	'03A02A	MLG Brake LH/RH
__-60_	'03A02B	Master Brake Cylinders LH/RH
__-60_	'03A02C	Main Brake Disk
__-60_	'03A02D	Friction Lining
__-60_	'03A03	Parking Brake Installation

The MADW data used in this study is one year of maintenance records from the Army Aviation branch. This sample of data has 1.4 million records and includes all text entries associated with each record, the words and/or exclusionary word the expert system search algorithm found for each record, as well as corrosion cost percentage and non-available time assigned to that record by the cost and availability corrosion algorithms. A subset of the features available in the MADW data is shown in Table 6.

Table 6: MADW Data Sample

New Column Name	User-Friendly Name	Description	Sample Value	Calc/Import?
AVAILCD	Availability Code	Code which signifies the record resulted in NMC time	Z	Import
CORROCATCD	Corrosion Category Code	Corrosion category code - used if a service code exists	C	Import
ISCORROSION	Corrosion Related	A flag that denotes whether work depicted in the labor record is corrosion-related or not.	TRUE	Calc
CORROTAT	Corrosion Turnaround Time	Corrosion turn around time - if AvailCD flags record, it is the reconciled non-availability multiplied by corrosion percent	21.5	Calc
UnitCD	Unit Code	A code identifying the unit (UIC)	W0Y407	Import
MAINTCOMPLDATE	Maintenance Completion Date	The date the maintenance operation was completed.	11/23/18	
MAINTSTARTDATE	Maintenance Start Date	The date the maintenance operation was started.	11/23/18	Import
TIER1OBJECT	Tier1 Object	The main "parent object" being maintained. The are approximately 50 Tier 1 objects.	ENGINE OR PROPULSION	Calc
TIER2OBJECT	Tier2 Object	The "child" object of the identified Tier 1 object. There can be up to 500 Tier 2 objects for a Tier 1 object.	CIRCUIT CARD	Calc

3.3. Objective 1: Measuring Object Identification Accuracy

3.3.1. Research Procedures

In Objective 1, the goal is to identify the accuracy of MADW objects through assessing whether each ASAP-RAM part could be found in the MADW, and whether the Tier 1 and 2 Objects could be mapped to an assembly *up the tree* from the ASAP-RAM part in the UH-60 parts tree. The method was to use a non-MADW data source, DA2410 ASAP-RAM data, to verify the accuracy of the MADW Tier 1 and 2 Objects. The hypothesis was that if the MADW is accurate, then each DA2410 ASAP-RAM record should be able to be joined to its corresponding MADW maintenance record. This research proceeded as follows:

1. Join the ASAP-RAM data to the MADW by maintenance date range and unit identification code (a many-to-one join).

2. Join the best matching UH-60 parts tree part to each record on the DA2410 part. As part of this step:
    a. Different approximate string-matching techniques were tested for linking the object description strings.
    b. The UH-60 parts tree was processed into a hierarchical data structure to allow searching for string matches *up the tree* from the ASAP-RAM DA2410 part.
    c. The distribution of similarity scores for the DA2410 parts to the UH-60 parts tree were generated. The classification technique to identify matched and non-matched records was developed.
3. With the best string-matching and classification technique form step 2, generate the distribution of similarity scores for the MADW Tier 1 and 2 Objects to the UH-60 parts tree for the part matched to the ASAP-RAM part and any part up the tree in the hierarchy.
4. Evaluate the similarity score distributions and apply the classification technique from step 2 to assess the accuracy of the MADW objects.

The record linkage procedure is depicted in Figure 3.1.

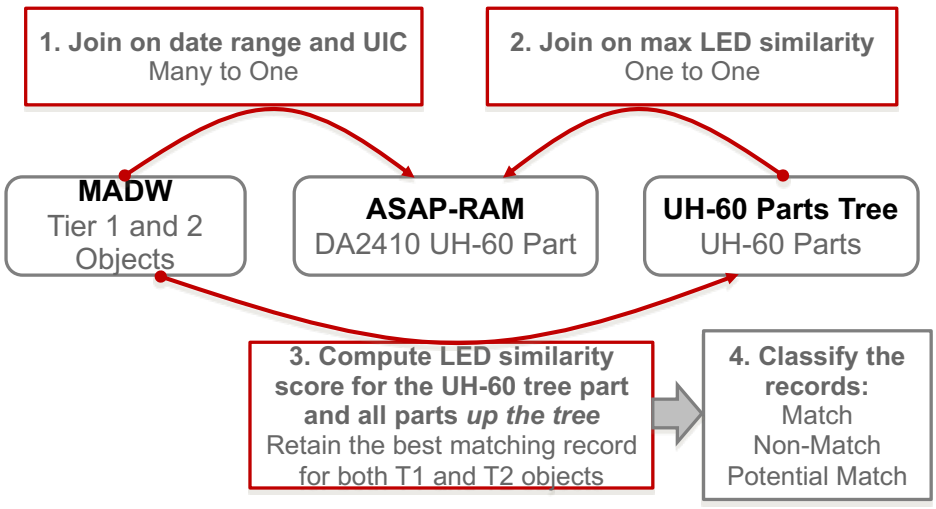

Figure 3.1: Record Linkage Procedure

1.3.3.1 Treatment of the Data

The record linkage process was followed to explore the first objective. Data pre-processing was completed to make the text uniformly lowercase, remove unwanted characters and words, expand abbreviations and splitting or combining attributes. Correcting misspellings was not done, because initial trials were correcting maintenance-specific terms into more common words, essentially removing important information from the data.

Next, the "ground truth" data sets, the ASAP-RAM and UH-60 parts tree, would be compared. Blocking was not used as an indexing technique when comparing the ASAP-RAM parts to the UH-60 parts tree. These data sets were small enough that every record of each data set could be compared to every record of the other.

Record pair comparison scores were calculated using Levenshtein edit distance, Jaro-Winkler, Editext, Longest Common Substring, Bag Distance, Ratcliff-Obershelp, Monge-Elkin, and Jaccard algorithms. The highest scoring ASAP-RAM part was retained for each similarity metric as the potential match to the UH-60 parts tree part. Upon manual inspection of the result of each approximate string-matching run comparing UH-60 parts tree to the ASAP-RAM parts, the Levenshtein edit distance (LED) was selected as the metric that produced the most logical matches. A combination of pre-sorting of the part terms and applying LED yielded the best results. The similarity scores for LED were calculated on a 0 to 100 scale, with 100 being an exact match, according to:

$$\frac{(|s_1| + |s_2|) - LED_{s_1,s_2}(i,j)}{|s_1| + |s_2|} \times 100,$$

where $|s_1|$ and $|s_2|$ are the lengths of the two sequences being compared.

The classification step bins each record pair into a match, non-match, or potential match. However, for matching the ASAP-RAM part to the UH-60 tree part, it was critical to identify matches and non-matches. Therefore all compared records of ASAP-RAM parts and the UH-60 parts tree were manually verified to assess the true label, match or non-match. The evaluation step assesses the quality of matches identified in the classification step. If we consider the MADW tier objects and the part names considered in this study, we use $\sigma = 40$ to consider the 26 letters, digits 0-9, and four additional special characters: commas, dashes, and parentheses. Given this, a conservative bound for similarity score where few matches exist is $\alpha > 1 - \frac{e}{\sqrt{\sigma}} = 1 - \frac{e}{\sqrt{40}} = 0.5702$, which corresponds to a similarity score of 43 by the maximum error rate's relationship to similarity score, given by $(1 - \alpha) \times 100$. Thus, any LED similarity scores at or below 43 were classified as non-matches.

All potential matches of ASAP-RAM parts and the UH-60 parts tree were manually verified to assess the true label, allowing the true sample distributions of similarity scores to be known for both matches and non-matches. The finding of Navarro above was consistent with the sample distributions of similarity scores, shown in Figure 3.2. The lowest similarity score that yielded a match was 48, which is greater than the non-match threshold of 43. The greatest similarity score that yielded a non-match was 74, thus this was chosen as the threshold above which we considered all record linkages to be matches.

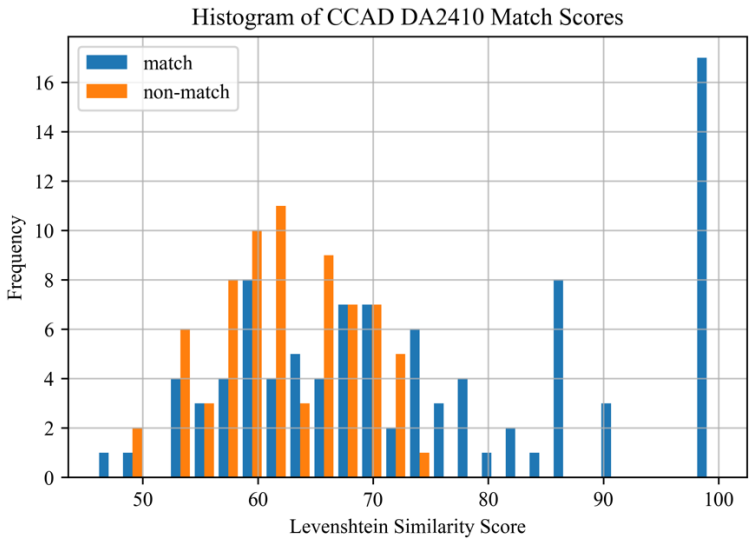

Figure 3.2: Distribution of Similarity Scores for Matches and Non-Matches (ASAP-RAM to UH-60 Parts Tree)

For the similarity scores between 43 and 74, which are the remaining potential matches, the sample proportion matching was 0.429 with a corresponding 95% confidence interval on the proportion of matches of (0.342, 0.515). These thresholds and calculated proportion of non-matches can be used in comparing the MADW Tier 1 and 2 Objects with the UH-60 parts tree. These are summarized as shown in Table 7.

Table 7: Match Categorization Summary

Match Category	Similarity Score Interval	Remarks
Non-Match	[0 , 43)	-
Potential Match	[43 , 74]	Match proportion is between (0.342, 0.515) with 95% confidence.
Match	(74 , 100]	-

Using the above categorization metrics, the overall accuracy of the MADW objects can be estimated as follows:

$$\frac{total\ matches}{total\ objects} = \frac{count\ of\ scores\ in\ (74,100] + p * count\ of\ scores\ in\ [43,74]}{total\ count\ of\ objects},$$

$$where\ p\ indicates\ the\ match\ proportion.$$

This calculation would be conducted at each limit of the 95% confidence interval on the potential match interval.

The next step was to join the MADW data to the ASAP-RAM data by time and unit variables. The date of the DA2410 from the ASAP-RAM data was compared to the maintenance start and end date window from the MADW and only those records from the MADW that had overlapping date ranges as well as matching unit codes (UIC and UnitCD) were joined to the ASAP-RAM records. This blocking technique reduced the computational complexity of comparing each MADW record to each ASAP-RAM record. To perform this join, the MADW data was chunked into 25-megabyte objects, then blocked on the maintenance start and end dates, followed by blocking on the Unit Codes. These chunks were processed in parallel on a multi-core machine to ensure a reasonable run time in the data matching process. This generated the data set that would contain potential matches of the UH-60 parts tree to the MADW Tier 1 and Tier 2 Objects. For each ASAP-RAM record, there were many potential MADW matches to be assessed by the record linkage process.

The blocking used to join the ASAP-RAM data to the MADW sufficiently reduced dimensionality such that when later calculating the LED similarity scores, every joined record could be evaluated. Because the best matching UH-60 tree part had been previously matched to the ASAP-RAM parts, the MADW Tier 1 and 2 Objects could now be compared to the proper "branch" of the UH-60 parts tree. The LED similarity scores were then calculated for each joined record of the ASAP-RAM data, comparing the corresponding Tier 1 and 2 Objects to the best matching UH-60 parts tree part, as well as every UH-60 tree part *up the tree* from the matching ASAP-RAM part. If the ASAP-RAM DA2410 part was the "whole" maintenance action, then the Tier 2 Object should have a one-to-one match to the ASAP-RAM DA2410 part. If the ASAP-RAM DA2410 part was just one step of a larger maintenance action, then the Tier 2 Object should match somewhere up the UH-60 parts tree from the DA2410 part. An example of the relationship of the ASAP-RAM part and Tier 1 and 2 Objects is shown in Figure 3.3. The ASAP-RAM part should be at the same level or at a lower level beneath the Tier 2 Object, as depicted in the figure.

For each unique ASAP-RAM record, the MADW record with the highest similarity score for both Tier 1 and 2 Objects was retained in the data and all other records dropped. Thus, the final data set contained a unique ASAP-RAM DA2410 record in each row, with its corresponding best-matched MADW record, along with the corresponding Tier 1 and 2 Objects and their similarity scores to the best matching UH-60 tree part.

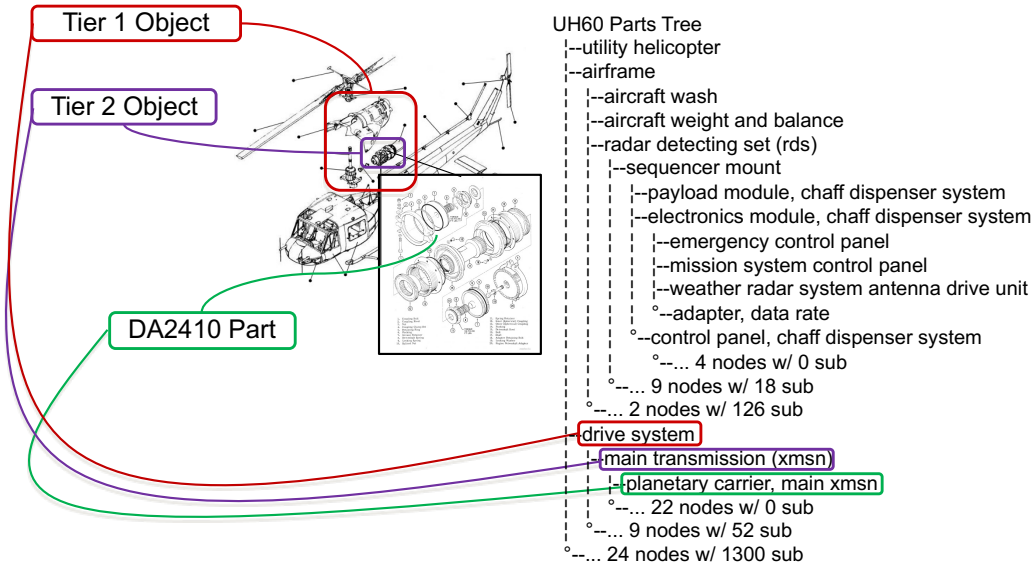

Figure 3.3: Example of the ASAP-RAM DA2410 Part, Tier 1 and Tier 2 Objects

Finally, to evaluate the completeness and accuracy of the matching process, ground truth data as assessed by a human was used to assess precision and recall on a subset of the testing data, or the MADW Tier Object matches. One hundred samples were inspected within each match category to ensure the assumptions of the matching strategy held.

3.4. Objective 2: Improving Corrosion-Tagging

3.4.1. Research Methods and Procedures

In Objective 2, the goal is to improve the MADW corrosion record-tagging algorithm by identifying ways to keep the expert system keywords up to date and by assessing the current algorithm's accuracy. The method was to use deep learning NLP techniques, specifically dense word embeddings and neural network text classification modeling approaches. The hypothesis was that deep learning would provide an improved approach to modeling maintenance language and word embeddings would provide a way to identify terms used in the maintenance records that are similar to the current expert system keywords. This research proceeded as follows:

1. The maintenance free text entries and their corresponding corrosion tags (as tagged by the MADW expert system) were extracted from the 1.4 million MADW records.
2. Text pre-processing was conducted to reduce the dimensionality of the data.
3. Word embeddings were generated through use of skip-grams (Word2Vec) and word co-occurrences (GLoVE).

4. The embedding model parameters were trained in a series of experiments to find hyperparameter values that produced useful word embeddings.
5. Using the trained word embeddings, cosine similarity was used to identify which words were most similar to the current expert system keywords.
6. Text classification neural networks were constructed, leveraging the word embeddings, and trained to classify records as corrosion-related or not.
7. The models were evaluated to assess performance with data not used in the training process. Error analysis was conducted to understand how to improve the text classification and embeddings.
8. The text classification models were run over records identified under Objective 1 as being a true match to the ASAP-RAM human-labeled data.
9. The neural network model performance was compared to the expert system model performance, using metrics of accuracy, precision, recall, and F1 score.

3.4.2. Research Sample

The data used under this objective is the same MADW data set used in Objective 1. Again, this sample of data has 1.4 million records and includes all text entries associated with each record, the words and/or exclusionary word the expert system search algorithm found for each record, as well as corrosion cost percentage and non-available time assigned to that record by the cost and availability corrosion algorithms. The two variables of interest were the free text entries and the corrosion tags for each maintenance record.

3.4.3. Treatment of the Data

Text pre-processing was conducted prior to applying training algorithms to the text data. Pre-processing was used to the extent that the data allowed. The steps taken with MADW data are listed below.

1. Text tokenization was used to split text data into individual words.
2. Unnecessary tokens and stop words were removed from the data. This step consists of removing words that have little or no significance, like “a,” “the,” or “and.”
3. Expanding contractions consists of locating contractions in the text and replacing them with the full words, like replacing “couldn’t” with “could not,” was completed with the maintenance data.
4. Making all text uniformly lowercase and removing special characters was completed with the maintenance data.

5. Correcting spelling errors consists of replacing words that have character(s) in the wrong location, added erroneously, or omitted entirely with the correct spelling of the word, was found to be more detrimental than beneficial in the case of the maintenance text descriptions because of how many non-English words, abbreviations, and maintenance slang terms were included in the text corpus.
6. Stemming and lemmatization consists of finding the root form, or lemma, of each word by removing affixes from the word, like replacing "cleaning" with "clean." This was done with the maintenance data.
7. Part of speech tagging consists of labeling each word in a sentence with its proper lexical category in order to apply structure to the data. In the case of the maintenance text, this was not possible due to the lack of adherence to English language rules when inputting text into the maintenance data system.
8. Parsing or chunking consists of organizing text into language structures like phrases and sentences. This was not possible due to the same reason part of speech tagging was not done.
9. Removal of duplicate records. This was used to allow the model to treat each maintenance action equally, without keying on the most frequent actions, which are typically routine services, like quarterly or annual services.

Once the data was preprocessed, the vocabulary consisted of 41,403 unique single-word tokens. An example of the pre-processing and tokenization step is shown in Figure 3.4.

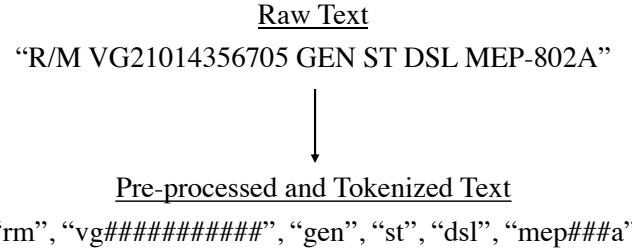

Figure 3.4: Pre-processing and Tokenization Example

Each token was assigned a unique integer such that the numeric vectors can be linked back to the token text. Next, the labeled data was generated via the skip-gram generator, which is a function that labels words that appear in the same n-gram window with a "1" and words that do not appear in the same n-gram window with a "0." This function extracts all the n-grams from the text data, then uses positive and negative sampling to generate the labeled data. The labeled data is what allows the neural network to train and adjust the word embedding vectors such that similar words will have embeddings that are very close to one another in Euclidian space.

The Skip-Gram architecture was then constructed in Python using the Keras deep learning framework. This model consisted of two parallel stacks of layers, consisting of the input layer, embedding layer, followed by the dot product layer to re-connect the two parallel stacks. Following a dense fully-connected layer which utilized a sigmoid activation function, the true labels are compared to the predicted labels and error or loss was calculated. The errors were then backpropagated to adjust the embedding weights in order to reduce the errors. Initially, the embedding weights are randomly initialized, then iteratively adjusted a small amount in the training process, such that error is minimized. The Skip-Gram architecture used is depicted in Figure 3.5.

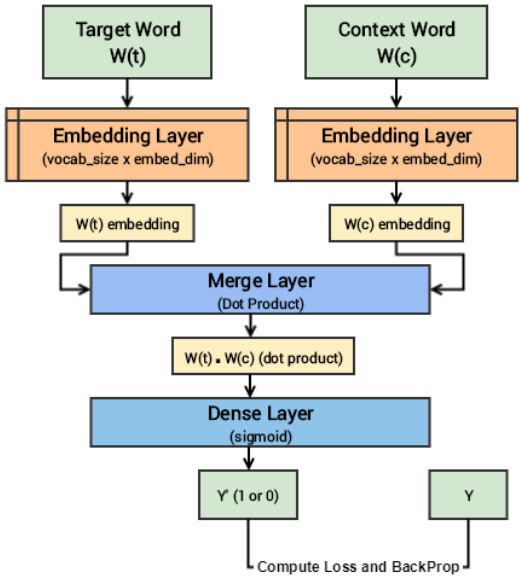

Figure 3.5: Skip-Gram Architecture (Sarkar, 2019 p. 251)

The neural network was trained in a series of experiments in order to find the hyperparameters, the "un-trainable" model parameters, which allow the learning algorithm to minimize the final model error. The model with the lowest error would provide the best representation of the maintenance language semantics, accurately associating common words with each other through use of the words' vector embeddings. The hyperparameters tested in the deep learning experiments are shown in Table 8.

Table 8: Word Embedding Experiments

Hyperparameter	**Values**
Epochs	1 to 15
Embedding Dimension	50, 100, 300
Architecture	GLoVE, SkipGram Word2Vec (2)
Window Size (n-grams)	2 to 10
Vocabulary Size	10k, 20k, 40k

With the best performing word embeddings, the expert system keyword embeddings were compared to all other word embeddings found in the vocabulary of the 1.4 million maintenance records. The top ten similar words were then identified using a Euclidean distance metric, retaining the words with the smallest distance to the expert system search words. Any similar words that were already accounted for in the expert system keywords were then dropped, leaving only the potentially similar words that needed to be considered for use in the expert system.

Finally, the records identified as a match between the ASAP-RAM data and the MADW under Objective 1 methods could be used as the true labeled data for the corrosion tagging algorithm. This data set could be used to compare the performance of the expert system to that of the neural network models. The neural network model was run over these records and the accuracy of each method compared.

The text classification neural networks were constructed, trained and evaluated according to the following steps:

1. The input text was pre-processed in accordance with the same steps identified above for the word embedding models, then tokenized for input into the neural network.
2. The data was split into train, validation, and test sets.
3. The model architecture was defined using the Keras API in Python.
4. The models were trained until the first epoch in which the training accuracy surpassed the validation accuracy, to avoid overfitting on the training data and causing the model to generalize poorly.
5. Each model was evaluated for accuracy, precision, recall, and F1 score on the hold-out test set.
6. The hyperparameters were tuned with a combination of automatic search and hand-tuning methods based likely optimal values found in the literature.

Two types of neural network architectures were developed and trained on the text classification task. The literature has shown the recurrent neural networks (RNNs) and convolutional neural networks (CNNs) can both be used to process sequence data like time series data and text data. Furthermore, RNNs that make use of long short-term memory (LSTM) units within layers were shown to have state of the art performance on NLP tasks, including text classification (Rome, 2018). 1-dimensional CNNs have been shown to provide an efficient and accurate ability to classify text data, performing very near LSTM models and training much more quickly (Chollet, 2018). Therefore, the two model architectures developed in this study were the LSTM and CNN networks.

For each of the LSTM and CNN architectures, models would be tested using different treatments of the embedding layer in each model. First, each model would utilize an embedding layer trained from scratch on the corpus of maintenance text in order to optimize the text classification

task of the neural network (corrosion-tagging). In this case, the embedding weights are randomly initialized at the start of training, then adjusted through back propagation in order to reduce the model error rate in the classification task. Second, each model's embedding layer would be initialized with the embedding weights developed through the SkipGram Word2Vec implementation. These weights were then fine-tuned to the text classification task during the training process, meaning the embeddings were adjusted through back propagation in order to reduce error on the text classification task. Finally, each model's embedding layer would be initialized with the embedding weights developed through the SkipGram Word2Vec implementation, but the embedding weights were fixed during training, meaning the Word2Vec embeddings were not adjusted at all during model training. Essentially, what was being tested in these three treatments of word embeddings is the degree to which the models benefit from pre-training the maintenance language token embeddings. The architectures for these models were built using the Keras deep learning framework and the Python programming language. The LSTM and CNN architectures are shown in Figure 3.6.

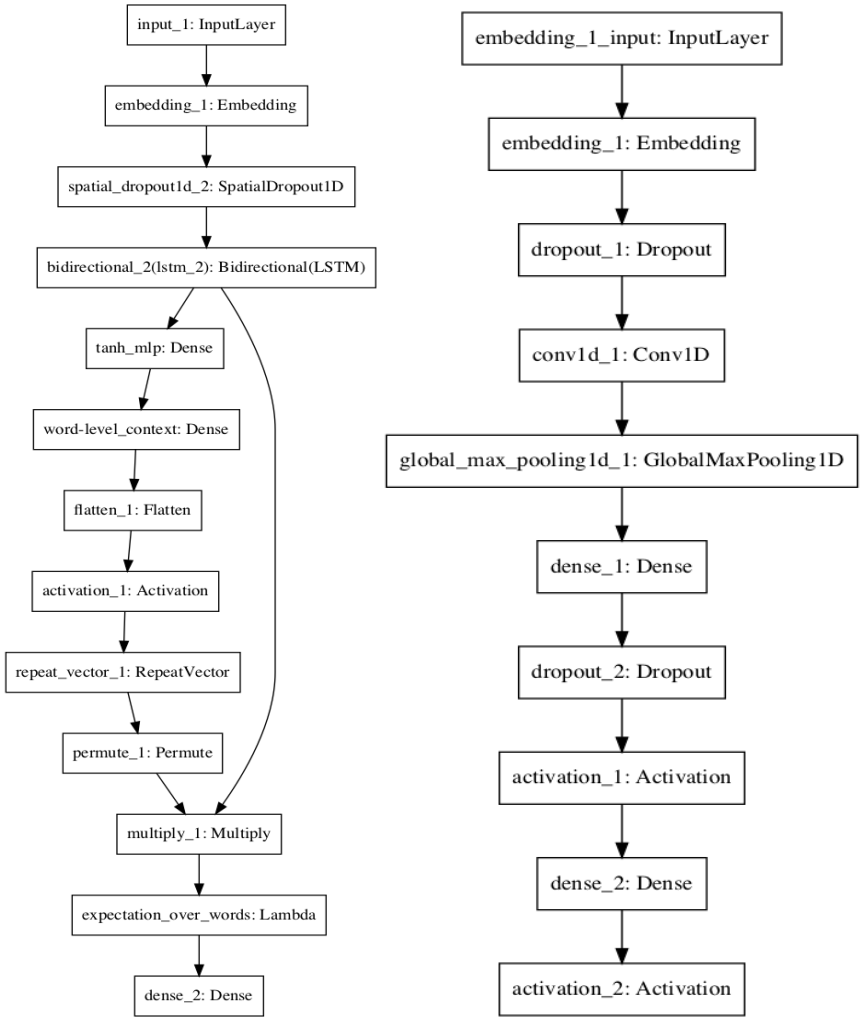

Figure 3.6: Model Architectures: LSTM on Left (Rome, 2018), CNN on Right (Chollet, 2018)

It is important to note that the text classification models are being trained to predict corrosion labels *as labeled from the MADW expert system*. These labels cannot be considered the "true" labels, unless verified by a human SME. Thus, a true labeled test set was needed to evaluate model performance against reality, not just against the expert system predictions. The results of Objective 1 record linkage could now be used for the true labeled test set. The matched records from Objective 1 were used as a second test set of "ground truth" data to evaluate the performance of the neural network models compared to the MADW expert system's performance on corrosion-tagging.

The evaluation metrics used to evaluate model performance were accuracy, precision, recall, and F1 score. Accuracy is measured as the ratio of correctly predicted records to the total number of records. Precision is measured as the ratio of correctly predicted positive records (corrosion) to total number of positive predictions. Therefore, if precision is high, the model is not making mistakes when it predicts that a record is corrosion-related, however the model may be omitting the corrosion label for records that should be classified as corrosion related. Recall is measured as the ratio of correctly predicted positive records (corrosion) to the total actual positive records. Therefore, if recall is low, the model is mislabeling records that should have the corrosion label applied. Finally, F1 score is the harmonic mean of precision and recall. This is useful because it combines the two metrics of precision and recall, which measure different components of the model's performance. In cases where the true proportions of positive and negative labels are not equal, as in this study, the F1 score is the most useful metric of a model's performance.

3.5. Critical Assumptions

3.5.1. Best Similarity Score Selection

When using the LED for record linkage of the ASAP-RAM data to the UH-60 parts tree, it was assumed that the highest scoring similarity score was the best match available. It is possible in cases where the highest scoring pair yielded a non-match, that there was in fact a lower scoring match elsewhere in the data. This could occur because of the nature of Army parts descriptions, in which some parts are listed with additional description information, while others are only listed with their generic nomenclature. If the matching additional description information is longer than the actual part name, then this causes problems using the max similarity score. This is shown in the below example:

Table 9: Best Similarity Score Assumption Example

UH-60 Parts Tree	ASAP-RAM Part	Similarity Score	Actual Match Category
"winch, aircraft mounted"	*"winch, external"*	44	Match
"winch, aircraft mounted"	*"skid, aircraft mounted"*	84	Non-Match

3.5.2. Potential Match Proportion

In calculating the proportion of matches within the potential match category interval, [43,74], the MADW was assumed to yield a distribution of matches and non-matches that was not significantly different than that of the ASAP-RAM to UH-60 parts tree comparisons. The proportion of matches to non-matches in this similarity score interval is thought to be jointly dependent up both the input data (ASAP-RAM) and the approximate string-matching function (Levenshtein edit distance). When using the calculated proportion on the MADW, although the string-matching function remained the same, the source of input data changed. Therefore, the assumption that the distribution of matches to non-matches in the sim score interval [43, 74] rests upon the condition that the MADW and ASAP-RAM strings are not significantly different in distribution.

Chapter 4
Findings

4.1. Objective 1 Findings: MADW Object Accuracy

The Levenshtein similarity scores for the best matching UH-60 tree part on the same UH-60 parts branch as the ASAP-RAM DA2410 part is shown in Figure 4.1. The color code indicates to which category of record linkage the similarity score corresponds. Figure 4.1 shows the Levenshtein similarity scores for Tier 1 and Tier 2 objects if the best matching Tier 2 object determines which record is selected for the respective ASAP-RAM DA2410 part.

Figure 4.1: Distribution of Similarity Scores for Best-Matching Records According to Tier 2 Object Match

According to the accuracy assessment calculations identified in Chapter 3, the 95% confidence interval of the true Tier 1 Object accuracy is estimated at (0.262 , 0.350), while the true Tier 2 Object accuracy is estimated at (0.592 , 0.691).

Figure 4.2 shows the Levenshtein similarity scores for Tier 1 and Tier 2 objects if the best matching Tier 1 object determines which record is selected for the respective ASAP-RAM DA2410 part. According to the accuracy assessment calculations identified in Chapter 3, the 95% confidence interval of the true Tier 1 Object accuracy is estimated at (0.502 , 0.631), while the true Tier 2 Object accuracy is estimated at (0.273 , 0.370).

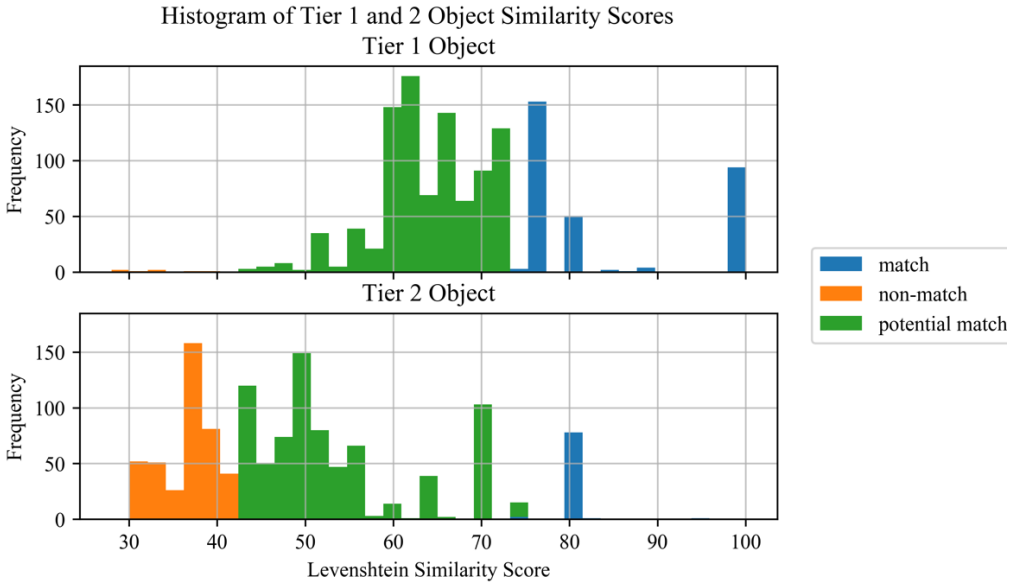

Figure 4.2: Distribution of Similarity Scores for Best-Matching Records According to Tier 1 Object Match

From these results, it is clear that if the Tier 2 Object score is prioritized, the Tier 1 Object similarity score suffers. The reverse is true if the Tier 1 Object similarity score is prioritized over the Tier 2 Object score. Because the Tier 2 Object is the lower-level identification of the object receiving maintenance, Tier 2 Object matches should be prioritized for declaring matched ASAP-RAM to MADW records. Therefore, the estimated accuracy, shown as a 95% confidence interval, of the MADW object identification algorithm is (0.592 , 0.691). It is important to note that this estimate is only based upon Army aviation maintenance records from the given time period and estimates for other maintenance sectors of the Army would need to be estimated using maintenance data specific to the targeted unit and equipment types.

With those records identified as a match when prioritizing Tier 2 Object similarity scores over Tier 1 Object scores, the MADW corrosion tags were compared to the ASAP-RAM part condition descriptions, which indicate the presence or absence of corrosion. The MADW corrosion tags were correct in 41.6% of the 495 matching records for 2017 ASAP-RAM records. A 95% confidence interval on the corrosion tag accuracy for all ASAP-RAM maintenance data is (0.373, 0.460). This estimate of the corrosion algorithm accuracy applies to the Army aviation maintenance data from the given time period only. The low accuracy estimate of the corrosion algorithm enhances the importance of Objective 2, improving the corrosion algorithm's accuracy.

4.2. Objective 2 Findings: Corrosion Record Tagging

4.2.1. Identification of Similar Corrosion Search Terms

The efforts to generate meaningful word embeddings leveraging NLP deep learning techniques was effective in identifying corrosion search terms that are not currently present in the MADW expert system. Using the methods outlined in Chapter 3, 2,422 potential search words were identified for possible inclusion in the MADW expert system search words. Most of these words are additional misspellings of other known corrosion-related terms that were not previously accounted for in the expert system keywords. However, there are also some synonyms for corrosion-prevention tasks that were identified in the data which were not previously included in the MADW algorithm. A sample of these terms are shown in Table 10. Clearly, not all of these terms should be included in the expert system, however, a corrosion SME could quickly identify which terms relate to corrosion and which do not. The full list of search terms is shown in Appendix C.

Table 10: Sample of Corrosion Keywords Identified with Word Embeddings

Current Corrosion Keyword	Nearest Neighbors to Corrosion Keyword (Not Found in Current Keywords)
disbond	['penetrate', 'delaminating', 'anhedral', 'delaminated']
debond	['region', 'erode', 'blending', 'meob']
debonde	['rebonde', 'deteriorate', 'nutplate', 'chafe', 'marked']
corrioson	['vg6corrosion', 'dsic', 'enchange', '3debonding', 'cap6']
inspeciton	['inspectrp', 'add1', 'inspectreset', 'conjucntion', 'preventitive']
treatment	['primere', 'gusset', 'inspectl', 'fret', 'flapper']
treating	['cplf', 'demond', 'authoriztion', 'repaierd', 'piitte']
coat	['blended', 'bare', 'painting']
cpc	['follower', 'application', 'compound', 'stabilization', 'prevention']
preparation	['ply', '3na', 'primere']
inpection	['inspectioninspection', 'differ', 'conjucntion', 'complette']
insection	['complette', 'completeed', 'inspectin', 'wit']
quarterly	['semiannual', 'semi', 'gauging', 'm2', 'hb']
periodic	['inspectioninspection', 'inspw']
monthly	['weekly', 'vrc', 'trc', 'cxp']
inspc	['cmplete', 'conjution', 'complet', 'coplete', 'conjucntion']
inspe	['visualcomplete', 'baffel', 'completen', 'dinsp']
inspt	['wtih', 'december', '6jul', 'conplete', 'inspection4']
test	['ok', 'good', 'turn', 'inop', 'reset']
ins	['faqult', 'egi2', 'fault6', 'compltete', 'adn']
isnp	['spiveyserviceability', 'tuckerserviceability', 'inspserviceability', 'mw', 'packserviceability']

4.2.2. Identification of Improved Corrosion-Tagging Algorithms

For each of the LSTM and CNN neural network architectures, three sub-models were evaluated for performance on the ASAP-RAM matched records:

1. Word embeddings learned from scratch (randomly initialized at the start of training).
2. Word embeddings pre-trained on the SkipGram Word2Vec implementation which were fine-tuned to the text classification task during training, meaning the embeddings were adjusted in order to reduce error on the text classification task.
3. Word embeddings pre-trained on the SkipGram Word2Vec implementation which were fixed during training, meaning the Word2Vec embeddings were not adjusted during training.

The performance metrics are based upon the models' ability to classify the maintenance records in accordance with the expert system tags, although the expert system tags are not known to be correct. Thus, the expert system can be thought of as a labeling function to provide the neural network with labeled data for training. Then a "ground truth" data set, identified through the record linkage procedures in Objective 1, is used to compare these models to the MADW expert system to assess relative performance.

1.4.2.2 Neural Network Model Performance (*Test Data as Labeled by the Expert System*)

Table 11 depicts the performance metrics of the neural network models trained on the expert system labeled data and tested on hold-out test data which was set aside from the same data set.

Table 11: Text Classification Model Performance Comparison

Model Architecture	Embedding Layer	Accuracy	Precision	Recall	F1 Score	Training Time
Bi-Directional LSTM with Attention	Learned from scratch	0.876	0.909	0.927	0.918	*180 min*
	Pre-trained and fine-tuned	0.876	0.927	*0.905*	0.916	29 min
	Pre-trained and fixed	0.765	0.766	0.987	0.863	26 min
1-Dimensional CNN	Learned from scratch	**0.957**	**0.976**	0.970	**0.973**	166 min
	Pre-trained and fine-tuned	*0.747*	*0.747*	**1.0**	*0.855*	18 min
	Pre-trained and fixed	*0.747*	*0.747*	**1.0**	*0.855*	**11 min**

The CNN was more effective at learning to classify the maintenance records (as predicted by the MADW expert system) than the LSTM, as shown by its maximum performance on all training metrics. The models with the best performance on the text classification problem were the models with embedding layers learned from scratch in conjunction with the text classification task. However, the training time on both these models was much longer than the pre-trained

embedding models. It is important to reiterate that these models are training on data *as labeled by the expert system*, thus model performance is measured in terms of how closely their corrosion-label predictions match that of the expert system. Performance on the ASAP-RAM verified data set yielded different results.

2.4.2.2 Neural Network Model Performance (*Test Data from ASAP-RAM true labels)*

Following the evaluation on the hold-out test data from the MADW records, the LSTM and CNN models were subsequently compared to the expert system through use of the 495 matched records identified in Objective 1, which were identified as true matches from the ASAP-RAM data to the MADW, based on the similarity score of the Tier 2 Objects. The performance metrics of the tested models are depicted below in Table 12. The LSTM model with pre-trained and fixed embeddings yielded a 58% improvement in accuracy and an 82% improvement in F1 score over the expert system on the matched records and was significantly better than the MADW expert system on all performance metrics. Additionally, the LSTM with embeddings trained from scratch was significantly better than the expert system in all categories and achieved the highest performance on the precision metric of all models. However, the LSTM with pre-trained and fixed embeddings was significantly better than the LSTM with embeddings learned from scratch on the F1 metric.

Table 12: ASAP-RAM Test Set Model Performance Comparison

Model Architecture	**Embedding Layer**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
MADW Expert System	N/A	0.416	0.588	*0.333*	0.425
Bi-Directional LSTM with Attention	Learned from scratch	0.549	**0.745**	0.464	0.572
	Pre-trained and fine-tuned to task	0.444	0.632	0.343	0.444
	Pre-trained and fixed	**0.659**	0.674	0.916	0.777
1-Dimensional CNN	Learned from scratch	*0.402*	*0.565*	0.340	*0.424*
	Pre-trained and fine-tuned to task	0.648	0.648	**1.0**	**0.787**
	Pre-trained and fixed	0.648	0.648	**1.0**	**0.787**

Interestingly, the CNN models which used the pre-trained embeddings produced identical model performance for both the fine-tuned and fixed embedding models. Both of these models outperformed all other models on both recall and F1 score, achieving 100% recall and 78.7% F1 score. The CNN models with pre-trained embeddings achieved a 200% improvement on recall and an 85% improvement on F1 score over the expert system. The CNN with embeddings learned from scratch yielded a performance not significantly different from the MADW expert system and actually performed worse in terms of accuracy, precision, and F1 score.

The neural network models were able to better predict against the test set likely due to the benefit of the information contained in the embedding layer. While the MADW expert system can only search for explicitly identified words in the text entries, the neural networks can make predictions based off of all the semantically similar important words found in the vocabulary. This allows the deep learning models to better generalize to maintenance data, as was seen in the ASAP-RAM matched data set.

The bi-directional LSTM model has interpretability advantages over the CNN. One drawback of deep learning models is that they are not explainable to someone unfamiliar with the algorithms, as compared to a statistical model like linear regression or analysis of variance. The "attention" layer used in the LSTM neural network helps to solve the interpretability shortcoming, whereby identifying the terms which were most important to the model prediction. This is identified in Figure 4.3, where the model uses the *get_word_importances()* function to predict the probability that the text is corrosion-related and the relative word importance in making that prediction. In the example, "rust" was identified as the most important word, resulting in a prediction of 99.6% probability of corrosion for the maintenance record.

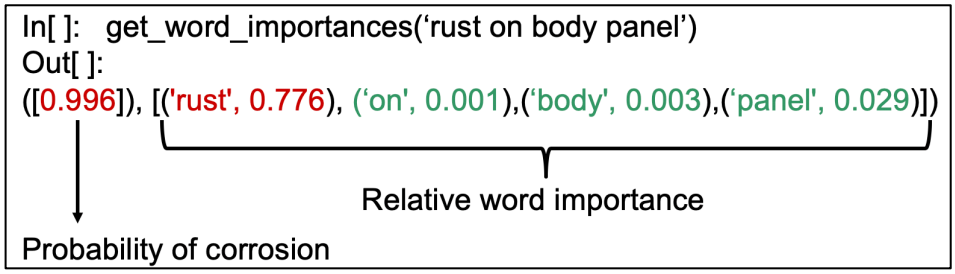

Figure 4.3: Attention Layer Interpretability Example

Because the neural network models proved to be significantly better at classifying maintenance records, use of the neural networks should be considered by decision makers if the interpretability features are acceptable in comparison to the expert system. The idea that the neural network model was essentially trained by the expert system could provide confidence in the model output and interpretability features as well.

4.3. Unanticipated Results

4.3.1. Non-Matching ASAP-RAM Records to UH-60 Parts Tree

An important unanticipated result was how many non-matches existed for comparing the human-verified ASAP-RAM parts to the UH-60 parts tree parts. The highest scoring LED similarity score was a non-match in 43% of the attempted record linkages. The distribution of LED similarity scores and their match or non-match category is shown in Figure 4.4. This indicates that even beyond free text entry at the point of maintenance action, maintenance data is variable and difficult to match even in cases where a human has verified the data inputs. The implication is that analysis based on low quality data will yield low quality insights. Additionally, this may

indicate that maintainers are not using the technical nomenclatures for the parts and the proper terms for the maintenance action, which would increase the variability of the maintenance data.

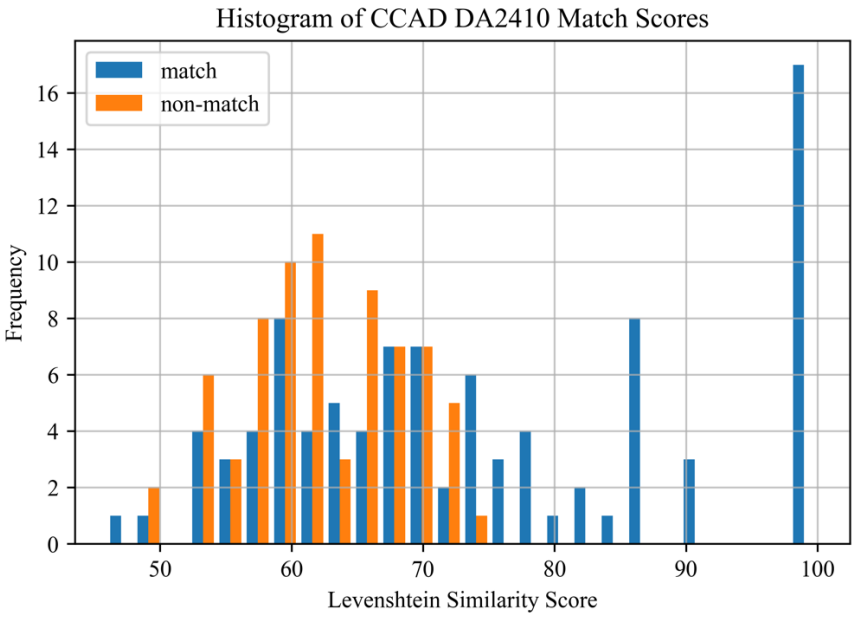

Figure 4.4: Distribution of Similarity Scores for ASAP-RAM parts to UH-60 Tree Parts

4.3.2. Testing Performance on ASAP-RAM Maintenance Records

A second unanticipated result was in the difficulty of corrosion-tagging records from the "ground truth" ASAP-RAM test set. These ASAP-RAM records were identified as matches in the record linkage efforts under Objective 1 of this study, which demonstrated a 41.6% corrosion-tag accuracy and a 42.5% F1 score. Although the neural network models achieved an accuracy significantly higher than that of the MADW expert system, the best neural network accuracy was only 65.9% and best F1 score was only 78.7%. This is indicative of the low quality of data and the high variability of the maintenance text entries.

Chapter 5
Conclusions and Implications

5.1. Summary of Findings

The estimated accuracy of the MADW object identification algorithm for Army Aviation records is between 59.2% and 69.1%, with 95% confidence. For those records identified as a match when prioritizing Tier 2 Object approximate-string matching similarity scores, the MADW corrosion tags were correct in 41.6% of the 495 matching records for 2017 ASAP-RAM records. Corrosion tag accuracy for all ASAP-RAM maintenance data is between 37.3% and 46.0%, with 95% confidence.

NLP deep learning techniques to generate word embeddings were effective in identifying corrosion search terms that are not currently present in the expert system. A total of 2,422 potential search words were identified for possible inclusion in the MADW expert system search words. NLP deep learning techniques to build an improved text classification system for corrosion-tagging were effective as well. The LSTM model with pre-trained and fixed embeddings yielded a 58% improvement in accuracy and an 82% improvement in F1 score over the expert system on the matched records and was significantly better than the MADW expert system on all performance metrics. The CNN models with pre-trained embeddings achieved a 200% improvement on recall and an 85% improvement on F1 score over the expert system. However, the overall performance of the classification models indicates the poor quality of the maintenance data within the MADW.

5.2. Conclusions Based on Findings

The findings of Objective 1 indicate that at the high end, the MADW's identification of the object receiving maintenance is 69% accurate. This accuracy level is attributable to many factors, the least of which may be the MADW algorithm which identifies the Tier 1 and Tier 2 objects. Although not directly investigated, it is the author's opinion that the majority of the error rate of the MADW object algorithm is due to the raw data collected at the point of maintenance.

The findings of Objective 2 indicate that NLP deep learning approaches are powerful ways to understand maintenance text data and draw insights about corrosion in the Services. Deep learning has been shown as an effective tool to improve the existing expert system on the NLP task of text classification by up to 200% improvement on recall. While expert system models may be preferred due to their interpretability, this research showed that there are ways to allow deep learning model predictions to be better understood, as in the application of an "attention" layer in this study. Although the improvements of deep learning were large, the overall model F1 score only rose to 0.787, demonstrating the limitations of modeling this maintenance data. The foundation of meaningful analytics is the data itself and this study has implicated the MADW raw source data as problematic.

5.3. Impact of the Study

The accuracy level of the object identification algorithm and the corrosion-tagging algorithm affects any analytical insights gained from models or summary statistics that rely on this data. Although NLP deep learning approaches can significantly improve the MADW corrosion-tagging algorithm, the underlying data is so noisy that building models with high accuracy may not be possible. No matter how appropriate a modeling technique may be for maintenance text nor how well it is applied, if the input data is missing information or is inaccurate, the model results will still be poor.

Algorithms that run over data with missing information, extraneous data, and data entry errors will have limited performance and the insights drawn from the data must be used with caution. Analysts and decision makers should understand that policy and funding decisions based upon the MADW object data should not solely rely on data analysis from the MADW but take into account other forms of knowledge and insight gathered from across the Services. This is not an affront on the MADW algorithms, rather it is an attribution of the underlaying low quality data that is collected across the Services.

5.4. Recommended Future Work

The low accuracy of the object receiving maintenance and the low accuracy of the corrosion tagging algorithm, both attributable to low data quality, suggest that future efforts should be put into improving the data collection and data cleaning process. Because maintenance data is collected at the lowest levels across the Services, standardization of data collection methods is very important to reduce the variability in the maintenance records. A set of common data collection tools and processes would be very helpful to allow users of varying knowledge, skills, and abilities contribute meaningful and accurate data entry into the maintenance system.

Future work should include incorporating AI into the point of maintenance and data collection, such that the MADW input data is more accurate and easier to collect at the unit level. This could include mobile applications to document maintenance actions that limit free text entry and incorporate other forms of data, like images of the equipment receiving maintenance. Descriptions of maintenance work done could be compiled from PMCS procedures and the equipment technical manuals such that NLP AI systems could complete the input of a user as they begin to describe the maintenance action through voice or text. This technology already exists in the civilian sector and could be adapted for use in the military.

5.5. Conclusion

This study has delivered valuable insights into the accuracy of MADW algorithms, as well as the quality of the underlying maintenance data collected in the Army. The purpose of this research was to estimate the accuracy of the MADW object-identification algorithm and to identify ways to improve the corrosion tagging algorithm used in the MADW through use of deep learning

applications in NLP. Record linkage techniques were shown to estimate the MADW Tier 1 and Tier 2 Object identification accuracy. Deep learning NLP methods were shown to improve and update the expert system, allowing the continued use of an interpretable expert system algorithm in the MADW, which may be preferable to a more effective but less understood "black box" deep learning model. The first implication of this research is that expert systems need not be supplanted by deep learning methods at the expense of interpretability, instead, expert systems can be updated and improved by deep learning. Secondly, improving the algorithms that run on the MADW data is beneficial to improving the quality of insights gained from the data. Finally, future research efforts should focus on improvement of data quality through improved data collection and cleaning techniques.

Appendix A
References

Almeida, F. and Xexeo, G. (2019). Word Embeddings: A Survey. Retrieved from https://arxiv.org/abs/1901.09069

Chollet, F. (2018). *Deep Learning with Python*. Shelter Islands: Manning.

Chollet, F. and Others. (2015). Keras. Retrieved from https://keras.io

Cohen, A. (2020). FuzzyWuzzy: Fuzzy String Matching in Python. Retrieved from https://pypi.org/project/fuzzywuzzy/

Christen, P. (2012). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin: Springer.

Defense Acquisition University. (2019). Corrosion Prevention and Control. Retrieved from https://www.dau.edu/acquipedia/pages/articledetails.aspx#!507

Herzberg, E. (2015). Determining Corrosion's Effect on the Cost and Availability of DoD Weapon Systems and Equipment: Methodology. (Publication No. SAL4ITI). Arlington: LMI.

Herzog T.N., Scheuren F.J. & Winkler W.E. (2007). *Data quality and record linkage techniques.* New York: Springer.

Logistics Management Institute. (2019). Maintenance and Availability Data Warehouse: Authoritative Data Sources. Arlington: LMI.

McDonald, J. (2018). *An Introduction to Probabilistic Record Linkage.* London: Institute for Education.

Navarro, G. (2001). A guided tour to approximate string matching. ACM Computing Surveys 33(1), 31–88.

Rome, S. (2018). Understanding Attention in Neural Networks Mathematically. *Blog: A Math Ph.D. who works in Machine Learning.* https://srome.github.io/Understanding-Attention-in-Neural-Networks-Mathematically/

Sarkar, D. (2019). *Text Analytics with Python: a Practitioner's Guide to Natural Language Processing*. New York: Apress.

Skansi, S. (2018). *Introduction to Deep Learning: from Logical Calculus to Artificial Intelligence*. Cham: Springer.

U.S. Department of the Army. (2016). Army Data Strategy. Retrieved from http://ciog6.army.mil/Portals/1/20160303_Army_Data_Strategy_2016.pdf

U.S. Department of the Army. (2018). TRADOC Pamphlet 525-3-1: The U.S. Army in Multi-Domain Operations 2028. Retrieved from https://www.tradoc.army.mil/Portals/14/Documents/MDO/TP525-3-1_30Nov2018.pdf

U.S. Department of the Army. (2019). Army Transition Team: Winning Matters. Retrieved from https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/

U.S. Government Accountability Office. (2003). Opportunities to Reduce Corrosion Costs and Increase Readiness. (Publication No. GAO-03-753). Retrieved from http://www.gpo.gov/fdsys/pkg/GAOREPORTS-GAO-03-753/pdf/GAOREPORTS-GAO-03-753.pdf

U.S. Government Accountability Office. (2019). Observations on Changes to the Reporting Structure of DoD's Corrosion Office and Its Implementation of GAO Recommendations. (Publication No. GAO-19-513). Retrieved from http://www.gpo.gov/fdsys/pkg/GAOREPORTS-GAO-19-513/pdf/GAOREPORTS-GAO-19-513.pdf

Appendix B
Code

The code used in this analysis is available on the author's private GitHub repository. Contact MAJ John Case at john.j.case10.mil@mail.mil to request access to the code.

Appendix C
Additional Corrosion Search Words for Consideration

Current Corrosion Keyword	Nearest Neighbors to Corrosion Keyword (Not Found in Current Keywords)
hardener	['stabilizatn', 'ibet', 'mrke', 'fabricatedshorted', 'completepaint']
hardness	['parked', 'blanet', 'noticeably', 'damagesoundproofe', 'lbds']
preheat	['hvms', 'fuelfrombnut', 'meaurement', 'illistration', 'satcon']
debounding	['intalled', 'repairment', 'visualy', 'jung', 'turntable']
disbonding	['cloth', 'layer', '3na', 'errosion']
disponding	['defeere', 'hdfs', 'tb2a', 'fcrmma', 'update2']
debonding	['layer', 'lip', 'kevlar', 'leading']
disbonded	['disponde', 'cloth', '3na']
debonded	['4t', 'inspectionblue', 'mtpp', 'securinig', 'weatherproof']
debound	['olsen', 'darla', 'ripps', 'dshrd', 'preforme']
disbond	['penetrate', 'delaminating', 'anhedral', 'delaminated']
debond	['region', 'erode', 'blending', 'meob']
debonde	['rebonde', 'deteriorate', 'nutplate', 'chafe', 'marked']
baterie	['replac', 'sfa', 'u2', 'seviceability', 'usual']
ndi	['qty', '1ea', 'rl', 'pit', 'mike']
ndt	['penetrate', 'ch7', 'dzue', 'repaie']
contamination	['gbx', 'sampling', 'dpi', 'residual', 'technique']
polarization	['antirotation', 'fabicate', 'conns', 'alve', 'caustion']
regeneration	['whem', 'cecil', 'okvor', 'symboldowngrade', 'screws4']
assyleaking	['adjment', 'gathnee2', 'downlaod', 'prehoist', 'dampnerpitch']
cavitation	['pucture', 'serviceableleft', 'sgtmill', 'aphache', 'fuelboost']
insulation	['stack', 'wall', 'damaged', 'repaired']
blistered	['dogleg', 'sight1', 'incing', 'completeahrs', 'rhwheelhouses']
corriosion	['hairline', 'bladed', 'qdr', 'taxing', '2side']
corrision	['mildew', 'jamb', 'crosstube', 'penetrate', 'remv']
corrission	['inspwand', 'vaccume', 'fuild', 'side2', 'updtated']
corrrosion	['fgw', 'negetive', 'entry7', 'langing', 'clearwater']
microbial	['utlilty', 'janssen', 'linnk', 'witnin', 'b3k']
shrinkage	['refuelin', 'oilaft', 'nerve', 'sight4', 'venthoist']
coorosion	['conpsumption', 'oilaft', 'compadence', 'combinder', 'dgbb']
corossion	['groud', 'rule', 'requiredfor', 'idd']
correoded	['bldae', 'emphasis', 'suspense', 'radartrouble', 'aplifier']

corresion	['coast', 'testedwe', 'stripblack', 'resuscitator', 'testdisplay']
corrioson	['vg6corrosion', 'dsic', 'enchange', '3debonding', 'cap6']
corroded	['reinstale', 'wraparound', 'combuster', 'repaie', 'exaust']
degrease	['comanche', 'bookfcr', 'xsmission', 'troyaart', 'repairedinsp']
dryrotted	['bxb', 'loom', 'inspreplace', 'tt', 'verifacation']
extender	['lip', 'finger', 'radius', 'receptical']
fretting	['negligible', 'penetrate', 'propagate', 'dzue']
sandblast	['ndiphase', 'photographic', 'leavng', 'grossly', 'remolal']
sanderson	['tanksunit', 'broehm', 'ibg', 'prioe', 'amunition']
cathode	['sustem', 'rosetts', 'coordnination', 'unequal', 'hundreths']
classiii	['hemtt', 'cnv', 'marketing', 'simulation', 'shwr']
coroded	['reselecte', 'exmark', 'eppenage', 'attendantseat', 'gnh']
corosion	['seperating', 'soap', 'loom', 'borken', 'reinforcement']
correded	['tese', 'sprage', 'vidr', 'orme', 'ofwiper']
corrided	['yellowtail', 'couting', 'aftd', 'ibet', 'lightintg']
corrison	['holden', 'winsheild', 'replacent', 'requir', 'pck']
corrsion	['heel', 'jamb', 'midcowl', 'spaghetti', 'hindge']
corrsoin	['replacemol', 'boor', 'throttlesticke', 'ile', 'armourd']
crrosion	['wkcgd0', 'panelefab', '5fn', 'rhis', 'torquee']
erosion	['hysol', 'abrasion', 'adhesive', 'trailing']
flaking	['primere', 'meob', 'preppe']
leackage	['graound', 'adjusted6', 'comppleted', 'serrate', 'comleted']
leacking	['fmhigh', 'erronously', 'anvil', 'succession', 'fr9']
leakage	['consumption', 'seep', 'leaking', 'seeping', 'qd']
sanders	['lach', 'wss', 'dgse', 'tg', 'ry']
sanding	['windshild', 'reuqire', 'poly', 'bht', 'ctl']
arcing	['runway', 'initialization', 'stiffner', 'xmsm', 'heatshrink']
blaster	['unibal', 'inconjunctionwith', 'gearpmi2', 'daysh', 'reoccurr']
classii	['1inal', 'repairmultiple', 'dmm', 'conjonction', 'ssda']
cliii	['1vf', 'lookup', 'rg', 'mine']
corrion	['sealent', 'aardvark', 'boroscope', 'borscope', 'fatboy']
dryrot	['groud', 'dispo', 'errossion', 'penetration', 'boken']
galling	['doesent', '0hr', 'researvice', 'misfire', 'groundspee']
peeling	['delaminated', 'penetrate', 'negligible', 'hysole']
pitted	['matosbearing', 'kvalebushing', 'spiveybushing', 'hawkin', 'inspectionworn']

pitting	['scoring', 'beyound', 'disponde']
rotted	['remv', 'nicked', 'chaf', 'aplie', 'inproperly']
rotting	['weindshield', 'portrude', 'install2', 'strighten', 'canter']
shroud	['cracked', 'wall', 'grommet', 'orange', '5r']
silica	['servic', 'allot', 'xndcr', '4b1', 'punch']
solder	['ckt', 'initialize', 'traverse', 'glad', 'traction']
blast	['flip', 'interfere', 'supressor', 'jett', 's1']
clii	['koma', 'inus', 'hrse', 'gpa', 'stan']
clsiii	['sandba', 'vech', 'sdwfn', 'parcar', 'b2x']
corode	['eppenage', 'deecor', 'fuslage', 'stabs', 'tpo']
decay	['reliable', 'engie', 'trck', 'tcom', 'mr3']
flake	['meob', 'primere', 'deep']
oxide	['pmiphase', 'defferment', 'gpw4', 'dault', 'raplace']
rotten	['velco', 'football', 'ch9', 'mntg', 'saddlebag']
sanded	['eroded', 'ply', 'region', 'hisol']
sander	['insptreset', 'inspsmall', 'inspectionnondestructive', 'clarkreset', 'traughbermulitple']
scuff	['eroded', '3na', 'topside']
timimg	['againt', '1july', 'downinside', 'ttail']
timing	['4bb', 'headspace', 'unser', 'aey', 'liw']
cl3	['bt', 'uns', 'mdl', 'gladhand', 'djv']
cl2	['duew', 'compelete', 'inspectionpmi', 'omp', 'completeed']
cllll	['zvu', 'vspfm', 'aftt', 'assbly', 'swbd']
corr	['feature', 'kickpanel', 'apt', 'mccormick', 'negitive']
decal	['dpw', 'automobile', 'tailgate', 'tl', 'freightliner']
faded	['batteryivhum', 'withiaw', 'rtorques', 'bel', 'modife']
flux	['stby', 'recreate', 'momentarily', 'confidence', 'rapidly']
leack	['thermostatic', 'omit', 'repacement', 'ofthe', 'utilty']
leak	['hydraulic', 'line', 'pressure', 'hyd', 'fluid']
leans	['iem', 'bookfcr', 'mccroskey', 'depanle', '6pk']
resin	['cloth', 'ply', 'primere', 'topside']
rust	['x2', 'welding', 'pipe', 'rpr']
rusty	['donor', 'burr', 'oper', 'assem', 'parallel']
sprin	['compenster', 'ce7', 'pbo', 'sensetive', 'j1j3j5j7j9j']
cl1	['completetd', 'duplicte', 'inspectionl', 'assymbly']
cli	['hyunda', 'torne', 'wstc', 'side2', 'cools']

cls	['mdl', 'uns', 'jcb', 'glad']
fade	['insulator', 'color', 'moody', 'lw']
iii	['steering', 'bg', 'bobcat', 'rpl', 'gator']
leek	['releaed', 'installedl', '1para', 'lander', 'qq']
peel	['delamination', 'separation', 'mesh']
pmb	['york', 'asst', 'anp', 'botl']
rott	['bolf', 'lupi', 'faith', 'testingnp2', 'assyfail']
salt	['mile', 'volcano', 'nautical', 'tropic', 'extremely']
sand	['bare', 'hysol', 'blended', 'intrusion']
cl	['jt', 'steam', 'mtd', 'pltfm', 'steering']
lek	['comilete', '5miles', 'recevier', 'r1dtd2oct', 'detectoron']
lkg	['providedrequest', 'reinstalledrmvd', 'observerd', 'emtm1', 'inop5']
rot	['dog', 'recepticle', 'mate', 'repaired']
wet	['braid', 's1', 'acce', 'discolored', 'easily']
lk	['reinspecte', 'quill', 'bw', 'winshield', 'eroded']
ll	['reinstlle', 'wil', 'sutley', 'locs', 'dvep']
exfoliation	['okayheat', 'c9omp', 'accelarator', 'reservoirinsp', 'duecracke']
sheetmetal	['radius', 'lip', 'doghouse', 'delaminate']
deactivate	['incorect', 'duplicte', 'plateform', 'acutator', 'ltg']
substrate	['hydraulicdeck', 'pullstart', 'stickcollective', 'dema', 'toilerance']
abrasive	['penetrate', 'anhedral', 'blending', 'hysole']
shotpeen	['scorre', 'studaclc', 'ecessive', 'grtr', 'weightrequ']
stripping	['velco', 'jettsion', 'rn1', 'ln1', '1side']
grinding	['cease', 'pushbutton', 'fact', 'dopler', 'cptr']
shorted	['freeplay', 'xmsns', 'sustain', 'excedence', 'nt']
solvent	['freshwater', 'bracket2', 'replacedfail', 'amunition', 'ivhmssoftware5']
striping	['quintus', 'windsheildl', 'parkingbreak', 'sain', 'collectinve']
stripped	['sgr8', 'lac', 'sdi', 'dyd', 'filiment']
stripper	['kebeh', 'after', 'imus', 'dothan', 'diaganal']
bleach	['amplifer', 'fdma', 'seeral', 'elct', 'wrtie']
chassis	['elect', 'proc', 'teca', 'lrm', 'amplifier']
reweld	['accelerometer', 'eastman', 'duplicatebft', 'abf', 'rebuildfreewheele']
scrape	['usual', 'blur', 'runway', 'slice', 'adell']
striped	['williams', 'lefthand', 'name', 'beside', 'badly']
chasis	['dcm', 'initialization', 'cvfdr', 'times', 'vu']
disarm	['phillop', 'flohr', 'nosebay', 'lekae', 'engagedb']

grind	['sanuc', 'initially', 'downward', 'exhibit', 'barely']
pickle	['cago', 'mechanically', 'intrustion', 'consule', 'eppenage']
short	['fall', 'pick', 'difficult', 'adjusted', 'qce']
strip	['worn', 'edge', 'adhesive', 'hole']
stripe	['torq', 'lwr', 'hdw', 'saftie', 'nuts']
stripp	['ddate', 'met', 'maintenancepotentiometer', 'fmt', 'engineupper']
stript	['foault', 'normilehose', 'trainmission', 'moo', 'assosciated']
blend	['blended', 'bare', 'fiberglass', 'expose', 'nick']
dearm	['apendix', 'swadge', 'anywhere', 'wt3', 'bnc']
grnd	['ecl', 'parameter', 'mocs', 'spike', 'student']
scale	['articulation', 'rbc', 'collet', 'in', 'oz']
weld	['stand', 'vacuum', 'b4', 'welding']
body	['unser', 'unservicable', 'satellitecommunication', 'aff', 'vnc']
cure	['bond', 'touch', 'paddle', 'hysol', 'abrasion']
strp	['phillip', 'procce', 'corrisponde', 'drone', 'externally']
bod	['calper', 'repalcedphase', 'linsley', 't6o', 'inoop']
grn	['yel', 'pvh', 'ylw', 'rotation', 'ob']
wld	['grainy', 'pramac', 'asterisk', 'unmounted', 'mam']
determination	['winsheild', 'qaxial', 'filght', 'confirmation']
polyeurathane	['remov', 'scheduled9', 'aebi', 'alll', 'titanuim']
polyeurethane	['spe8ej', 'airfrae', 'sheen', 'news', 'sibily']
polyurethane	['bubbling', 'lifting', 'mesh', 'leading']
predeployment	['broken6', 'perofrm', 'voidsrp', 'sdscc', 'packingload']
preinspection	['hrness', 'broken2tail', 'demilfaile', 'gallagher', 'repairmultiple']
thermocouple	['arch', 'midframe', 'wat', 'patterson', 'locknut']
inspepection	['offect', 'sevric', 'bl4', 'intermittetly', 'outburd']
polyurithane	['evidance', 'snsrs', 'replece', 'retake', 'stripblack']
preflighting	['floater', 'substructure', 'inspectsocket', 'gja', 'peadal']
preservation	['serviceing', 'unitl', 'errror', 'falut', 'preasure']
reinspection	['outbaord', 'rotorhead', 'reinspecte', 'adjusment']
iinspection	['dqc', 'coimplete', 'inward', 'repairedprime']
initialinsp	['camer', 'facilit', 'lmit', 'ths', 'invaild']
innspection	['onreplacement', 'intercostol', 'mocmonthly', 'simuator', 'maximim']
inproccess	['timeinsp', 'c0mplete', 'finalinsp', 'ru', 'scheduledinsp']
inpsection	['packbushing', 'connecting', 'spiveyserviceability', 'eb', 'inspbeare']
inpspection	['tony', 'conducive', 'permanently', 'reinforce', 'boomhead']

inscpection	['snsrs', 'sighns', 'competed', 'reqremoval', 'intalled']
insepection	['electrolyte', 'torqe', 'regauge', 'vibrex', 'esm2']
insopection	['cciection', 'struttail', 'airleak', 'flud', 'upfaire']
inspcection	['timed', 'reviewpmi2', 'comand', 'foggy', 'fcd']
inspecation	['accellerate', 'dsouza', 'bottlerequest', 'positionrd', 'jurrisen']
inspecction	['anf', 'balence', 'completedno', 'afx']
inspecdtion	['completedgenerator', 'tighened', 'inedx', 'agt', 'benchtest']
inspecetion	['requiredi', 'rotationfault', 'cfompleted', 'insp3ection', 'confidenceni']
inspecition	['bifliar', 'escaping', 'finalblue', 'bburg', 'grather']
inspeection	['surpass', 'boxes', 'versilok', 'ductair', 'inspremote']
inspenction	['dialy', 'entir', 'detectable', 'breaver', 'mountsrequire']
inspoection	['dryrote', 'duestab', 'prome', 'incerrectly']
insprection	['dipenser', 'damdper', 'screew', 'pressre', 'emp']
inspsection	['glhtjfnm', 'suppert', 'holer', 'fritt', 'clk']
instpection	['prox', 'gga', '3july', 'inspok', 'rn']
reinspected	['externally', 'las', 'evaporater', 'nbg', 'eyebolt']
ultrasonic	['nom', 'usmh', 'sonic', 'vd', 'depos']
braketest	['installationoverhaul', 'touich', 'optimal', 'syem', 'chipped']
breechlock	['handrail', 'ferguson', 'rein', 'apt', 'peice']
camouflage	['1zp', 'grizzly', 'harne', 'gro', 'recep']
determined	['bacteriosttaic', 'roughly', 'blalnce', 'aek']
inapection	['badepusha', 'ecypiele', 'beggine', 'prome', 'completeeediaw']
inepection	['g2', 'aca', 'bennett', 'propane', 'hange']
insepction	['5hrs', '3july', 'inspok', '6jul']
insoection	['attacment', 'reinstal', 'ciele', '5semiannual', 'klesert']
inspcetion	['o1', 'hbfs', 'aog', 'afq', 'okayl']
inspeciton	['inspectrp', 'add1', 'inspectreset', 'conjucntion', 'preventitive']
inspestion	['stic', 'ringfuel', 'footpeg', 'mello', 'lockpinlube']
inspetcion	['supportreplace', 'nrts1reset', 'supervior', 'acfthour', 'required1']
instection	['mair', 'svcinsp', 'tubeaclc', 'tpo', 'mainteneance']
isnpection	['gp3', 'um', 'mcl', 'cuffphase']
isopropyl	['nutplatesfwd', 'engwash', 'overtempe', 'hur', 'snag']
overspray	['toque', 'corrossion', 'numer', 'fittings', 'sissor']
preflight	['cw2', 'discover', 'johnson', 'postflight', 'spc']
prelflight	['camply', 'exposed', 'wyc', 'cover2', 'acsent']
protecting	['cupple', 'arlam', 'hardwareheater', 'proffit', '1transition']

protection	['ply', 'penetrate', 'chaf', 'delaminated']
protective	['plastic', 'shrink', 'wall', 'minor', 'repaired']
repainted	['usual', 'unbonded', 'kickpanel', 'cutout', 'plexiglass']
repainting	['xed', 'lie', 'bng', 'sturt', 'balence']
scheduler	['2vpm', 'indicatorseal', 'fms1', 'm4k', 'retie']
undercoat	['cw1', 'zts', 'veiw', 'bruin', 'trunniontbo']
abraision	['bonded', 'bagging', 'spanwise', 'dft', 'countersunk']
lubricate	['wiper', 'rotary', 'lubed', 'reassemble']
polyamide	['metalfuel', 'elbowapu', 'primarily', 'reassembleinsp', 'copmletion']
powerwash	['inspectionhot', 'det1', 'shortshaft', 'safteyt', 'bammedevac']
protector	['gladhand', 'filament', 'thermo', 'dmg', 'mulitple']
repaint	['meob', 'primere', 'cloth']
silicone	['rn1', 'landyard', 'jettsion', 'crewcheif']
smoothing	['practicable', 'goal', 'chck', 'dau', 'cadu']
treatment	['primere', 'gusset', 'inspectl', 'fret', 'flapper']
acetone	['cordwise', 'alcohol', 'expand', 'trb', 'backstrap']
aerosol	['cardel', 'repairedcopilots', 'catellation', 'gathhcec', 'elongtate']
alodine	['mecf', 'tj', 'gusset', 'cj']
alodined	['replaceddash', 'ko', 'sercieve', 'discriminator']
inhibit	['strat', 'deselecte', 'succesfully', 'bienl', 'offload']
lacquer	['okayheat', 'soke', 'companygood', 'manifoldutility', 'dkaf']
perserve	['popwell', 'wirh', 'potentiomete', 'inspectionradio', 'recptical']
preserve	['iw', 'auxilary', 'thermostatic', 'utilty']
primered	['assesment', 'dgse', 'tpwd', 'cyl1', 'elimination']
priming	['integrity', 'intersystem', 'disconnecting', 'bsump']
primming	['corrsosionm', 'marry', 'installednew', 'lbb', 'echu']
protect	['grab', 'lightly', 'usual', 'blur', 'upcome']
recoated	['areasoundproofe', 'bealt', 'aaij', 'mdp1', 'intermittetly']
recolor	['slipmark', '2map', 'counld', 'glennrequest', 'assemblydue']
restaint	['busy', 'srvo', 'bearne', 'inspectionfwd', 'asg']
topcoat	['inspectionremoval', 'hovering', 'foodtruck', 'dfc', 'nonlighte']
treating	['cplf', 'demond', 'authoriztion', 'repaierd', 'piitte']
washdown	['cauiton', 'complonn', 'inprogresscord', 'lens6', 'cornert']
washrack	['rosan', 'rockets', 'clasp', 'dispersion', 'airborn']
coating	['bare', 'fiberglass', 'applied', 'shrink']
reprime	['conjuuction', 'undervoltage', 'groundspeed', 'itm', 'mas']

rewash	['grat', 'bushinginspection', 'accoradnce', 'inspectionsrequest', 'uppermost']
touchup	['hisol', 'anhedral', 'penetrate', 'chipping', 'dzue']
washing	['diag', 'polar', 'hp4dl', 'hpx', 'cv']
booth	['curtis', 'logged', 'garcia', 'verifacation', 'birdstrike']
breach	['refit', 'dmp', 'frnt', 'selctor', 'jammed']
breech	['helical', 'backplate', 'bur', 'aey', 'headspace']
coated	['replacedblue', 'deadband', 'disassebly', 'spilt', 'pope']
coatng	['blach', 'reninstalle', 'reass', 'jasonreset', 'completearl']
dirty	['reservoir', 'amount', 'seape', 'glass', 'leaking']
epoxy	['mecf', 'blended', 't6']
gloss	['ltq', 'oillb', 'inicate', 'wwtr', 'strife']
latex	['extchange', 'batella', 'comntaminate', 'refulle', 'incomplinace']
paint	['0h', 'rl', 'worn', 'surface']
panint	['ok3', 'statation', 'bobbyreset', 'inspectedinspect', 'allllllll']
panted	['hadswind', 'keepalive', 'ottom', 'completemtf', 'equalize']
preser	['initiation', 'noc', 'baremetal', 'inlns', '4v']
prime	['applied', 'blended', 'expose', 'core', 'comb']
primed	['appeared', 'lhsrhs', '1v', 'pdp1', 'zbh']
primer	['hysol', 'bare']
repain	['downcrack', 'resrvoir', 'farah', 'volage', 'leank']
repait	['halvesrequeste', 'blanketing', 'sheld', 'bqb', 'mainoil']
squeak	['combingramp', 'fairingscrew', 'troublrshooting', 'reassemble9', 'sep6']
squeal	['intermitten', 'scroll', 'wiggle', 'occasionally', 'grey']
squeel	['inus', 'seater', 'src', 'trasmit', 'rush']
treat	['fiberglass', 'bare', 'blended', 'expose']
wahser	['scheulde', 'operatoinal', 'htdraulic', '5semiannual', 'requiredscissor']
washed	['pior', 'perfomre', 'minspbearing', 'smaple', 'utiltity']
washes	['outbroad', 'zeuss', 'testpilot', 'assemblyswaged', 'iconj']
acid	['donor', 'gga', 'bold', 'hourly', 'fittings']
coat	['blended', 'bare', 'painting']
coats	['rebuildfreewheele', 'ordonio', 'progamatic', 'bb6', 'evaportator']
eddy	['rail', 't6', 'gusset', 'lighten', 'example']
haze	['blur', 'usual', 'withno', 'copliot', 'hazy']
piant	['giving', 'blurred', 'patchedaclc', 'inicate', 'moored']
prote	['perso', 'stake', 'vnc', 'aff', 'palletize']

prsov	['ats', 'fip', 'abort', 'pressurize', 'override']
squel	['rush', 'deselecte', 'distinct', 'cu']
washe	['sampleinsp', 'mcg', 'pullover', 'falult', 'toche']
washo	['enterreed', 'replave', 'csae', 'iceassist', 'tachmoeter']
clean	['around', 'open', 'need', 'leave']
camo	['iveco', 'accelerometr', 'wbxjaa', 'sunroof', 'spport']
cass	['magnectic', 'mfk', 'inu1', 'hinderence', 'updtated']
cpc	['follower', 'application', 'compound', 'stabilization', 'prevention']
dirt	['eapps', 'installle', 'adjacent', 'slider', 'underside']
film	['lubricant', 'solid', 'dusty', 'bifilar', 'expandable']
pain	['ciscous', 'ventaltion', 'ckeare', 'patchtemp', 'carswell']
pair	['ifm2', 'normaly', 'swith', 'unlatch', 'consol']
pant	['reshime', 'bieng', 'seventh', 'commutator', 'iba']
prsv	['inspectionpressure', 'coj', 'forget', 'labyrinth', 'requireservice']
wahs	['8x2', '3aw', 'tility', 'padie', 'hsge']
wash	['cci', 'coastal', 'fresh', 'storage', 'conjuntion']
ion	['aek', 'groud', 'hn', 'pul', 'adjsute']
mma	['lru', 'rfi', 'sply', 'psp', 'rcvr']
pnt	['installeation', 'aircraftrib', 'acsent', 'mrefer', 'acculate']
cln	['reliefe', 'retriction', '7fgcu', 'manufa', 'tb2a']
compliance	['reflect', 'copy', 'hard', 'delete']
complaince	['reinforce', 'curser', 'trasmission', 'omission', 'combination']
evaluation	['begin', 'reach', 'never', 'jump', 'percent']
evaulation	['turrect', 'accomadate', 'wrtie', 'iformation', 'vy']
reevaluate	['inob', 'forcrack', 'dn0k', 'halveswheel', 'emerency']
sanitation	['superceede', 'replacedcopilot', 'zipp', 'cutoff', 'act1']
fracture	['wearstrip', 'jammer', 'aplie', 'spoon', 'suport']
polishing	['inspectioncom', 'sheathe', 'moee', 'ultilty', 'idicate']
caulking	['ohv', 'wirute', 'electgricalconnector', 'buchanan', 'servor']
plating	['orderly', '3t', 'hyrdraulic', 'unravel', 'rerequire']
polished	['vick', 'vg2low', 'auxzillery', 'prevelant', '4kj']
refinish	['layard', 'vest', 'develpoment', 'undone', 'upwards']
sanitize	['repairmultiple', 'enf', 'rivte', 'ofnewir', 'nfault']
anneal	['servicedat', 'nozlle', 'coolijng', 'axc', 'andelbit']
chrome	['grove', 'machinist', 'disbounde', 'vertex', 'rebond']
finish	['meob', 'deep', 'layer']

polish	['unglue', 'rn1', 'reattache', 'aplie', 'velco']
silver	['gc', 'snake', 'psta', 'pouch', 'adhere']
stress	['someone', 'acm', 'feature', 'cluster', 'designate']
crack	['rivet', 'sm', 'mark', 'top', 'bracket']
nickel	['leading', 'layer', 'errosion']
nickle	['erode', 'meob', 'layer', '3na']
powder	['crewseat', 'fuild', 'restrainer', 'firwall', 'satus']
reface	['flying', 'arrivel', 'ivc', 'lau3', 'complain']
refine	['sidetone1', 'panelsn', 'retainor', '4eas', 'derow']
spray	['halfway', 'cloudy', 'buss', 'noisy', 'smoothly']
crach	['r0d', 'occonner', 'lxy4', 'seqpc', 'appllie']
crak	['missdrill', 'collett', 'usability', 'nitrogn', 'ateral']
crake	['hindge', 'nacell', 'mulitple', 'heel', 'enge']
crck	['sk', 'dep', 'pocc', 'inspl', 'balancetail']
creak	['leakikng', 'dissconnete', 'fms1damage', 'transimssion', 'cabale']
seal	['rubber', 'apply', 'washer', 'around', 'metal']
zinc	['backrest', 'wt1', 'lark', 'uneconomical', 'amalguard']
clad	['bol', 'pedestal', 'wrining', 'ashley', 'corisponde']
mask	['chemica', 'biolo', 'canister', 'chemical']
preparation	['ply', '3na', 'primere']
preperation	['supspect', 'likely', 'taxing', 'bale', 'elimination']
inprocess	['soum', 'inspectioninsp', 'inital', 'mo', 'yr']
inprrocess	['glhtlc6k', 'followie', '5miles', 'longbowcompleted', 'braken']
scheduling	['dtuslot', 'tfd', 'isdraine', 'accelerometor']
assurance	['assessment', 'hangteste', 'reinstillation', 'zimmerman', 'gov']
detected	['swch', 'cip', 'accurately', 'tacmap', 'blanking']
detection	['fadec', 'det', 'five', 'reinstalation']
determin	['lbmr', '6fc', '4x6', 'errored', 'amog']
determine	['reach', 'source', 'begin', 'restart', 'notice']
evaluated	['ppos', 'colwell', 'whnrc0', '3blue', 'headlower']
evalution	['fligth', 'instability', 'repeater', 'hardin', 'yield']
evauation	['wkc7aa', 'exxcessive', 'spport', 'purpo', 'rend']
inpection	['inspectioninspection', 'differ', 'conjucntion', 'complette']
insection	['complette', 'completeed', 'inspectin', 'wit']
inspction	['4aug', 'iw', 'comgenne', '3july', 'anotate']
inspecion	['inspectin', 'logbow', '5hrs', 'starp', 's0f']

inspetion	['defrre', 'inspectin', '6jul', '8sep', 'inspectionw']
ispection	['thermostatic', 'comgenne', '5hrs', 'inp']
loadtest	['wildland', 'sfa', 'rupture', 'motoring']
moisture	['almost', 's1', 'emer', 'restore', 'readable']
nspection	['comgenne', 'complette', 'inspectin']
prefilght	['auido', 'supplyinsp', 'initialndi', 'scedule', 'gathm6f0']
prefligth	['troublingshooting', 'gathhcec', 'supplu', 'leiberman', 'spindle9']
prelfight	['linkm', 'makre', 'requiredlateral', 'ietmmain', 'calculator']
pullout	['knurle', 'onyango', 'asvc', 'baselined', 'allowd']
quarterly	['semiannual', 'semi', 'gauging', 'm2', 'hb']
reinspect	['outbaord', 'thousandth', 'seapage', 'dir', 'omit']
roadtest	['burnish', 'yearber', 'henry', 'gng', 'casey']
scheduale	['fautls', 'dynacorp', 'edtu', 'certain', 'gorney']
schedule	['bays', 'deflector', 'anomaly', 'since', 'requirment']
scheduled	['inatalle', 'february', 'add2', 'completeiaw']
tirepair	['acccordence', 'phelps', 'requencie', 'colleen']
validtion	['dimention', 'seqpc', 'cpa', 'groundsystem', 'proeseal']
biennial	['unserv', 'recoverable', 'inital', 'a4', 'weldin']
bienniel	['gpa', 'hustler', 'colorado', 'koma', 'intern']
biennual	['ims', 'tacom', 'w6kg', 'wpf2c0', 'quartly']
complie	['by', 'eror', 'dowel', 's0f', 'links']
detecter	['boubble', 'runner', 'felming', 'skls', 'efrsii']
detrmine	['enryty', 'rdeice', 'functionallity', 'wie', 'okco']
evaluate	['spike', 'parameter', 'downward', 'climb', 'oscillation']
iinitial	['castelate', 'engen', 'unstick', 'stailite', 'phaserequeste']
iinspect	['dfif', 'visualminspection', 'kuusalu', 'recir', 'leash']
inpected	['vg2low', 'broehm', 'rsesulte', '4v', 'everthe']
inpsect	['outgoing', 'scft', 'saaf', 'bushingsinsp', 'lake']
inpspect	['defectice', 'informatonial', 'patriot3b', 'stacked', 'utplate']
inrocess	['spsdp1', 'incerrectly', '2ph', 'completemultiple', 'iawe']
insepct	['paddlere', 'harrdware', 'anilysis', 'iem', 'workinsp']
inspacet	['localizor', 'reviously', 'serv0', 'replacmeent', 'stitchweld']
inspeced	['unclogged', 'delaminiate', 'appllie', 'occonner', 'rotable']
inspecio	['atthis', 'chafte', 'reppel', 'onyellow', 'panelsl']
inspect	['damage', 'housing', 'fit', 'line']
intitial	['completerequeste', 'errode', 'progess', 'luke', 'o2']

periodic	['inspectioninspection', 'inspw']
prefight	['pdmanufacture', 'dmgd', 'tothe', 'yf', 'scheduledmain']
preflght	['attenant', 'pilotrequ', 'glhtlc6k', 'removide', 'faultsiaw']
prefligh	['resevour', 'rfq', 'researvice', 'orque', 'releasae']
prefligt	['latch2', 'autoengage', 'wrtench', 'testedflight', 'assemblyrequest']
prefliht	['refelect', 'saki', 'bealt', 'reinsterte', '2days']
prelight	['inbound', 'gorman', 'asap', 'ertel']
prepare	['finding', 'inspheat', 'hfs', 'workplatform']
quality	['accel', 'nulle', 'ski', 'etc', 'petroleum']
schedued	['attactchment', 'pressusre', 'spat', 'as6', 'hovers']
testunit	['troubleshootingduring', 'ringer', 'barn', 'aviaonic', 'pylonwill']
annaual	['meassage', 'tubbing', 'assemeble', 'armourd', 'initialy']
annnual	['confidenceni', 'replacedlongitudinal', 'swaqge', 'requencie', 'conteh']
annuual	['hatts', 'onreplacement', 'wd6uaa', 'imoroper', 'envriment']
biennal	['hedspace', 'linnk', 'canope', 'wwtr', 'completeqca']
cheange	['quarterlly', 'emergencty', 'phelps', 'zoose', 'intercomp']
comply	['amam', 'swage', 'asam', 'pendant', 'tr']
curciut	['ajacent', 'seqpc', 'distributor2nd', 'w6hkf9', 'codder']
curcuit	['readable', 'significantly', 'coordination', 'lgt', 'intermitantly']
detect	['controller', 'det', 'avr', 'il', 'operating']
detects	['yelloq', 'cornert', 'd3bf', 'wrj', 'm6v4']
filler	['hysol', 'abrasion', 'touch']
inpect	['preventitive', 'add1', 'icjw', 'conjucntion', 'conjution']
insepec	['skinndi', 'rresample', 'diffusr', 'sensorintermitent', 'repairmultiple']
inspdue	['completeinspect', 'inspectionlau', 'tiresreplace', 'snowplow', 'testedlow']
inspece	['isdraine', 'tide', 'servor', 'bitch', 'hangat']
intital	['reqrdservere', 'henry', 'sm1', 'servicesreset', 'intermediated']
isnpect	['utvsl', 'ngbdomops1', 'ks', 'nkr', 'sleave']
monthly	['weekly', 'vrc', 'trc', 'cxp']
months	['generater', 'reatarde', 'ncm', 'deterriorate', 'yw']
onspect	['servicale', 'subsiquent', 'serialp', 'clipclip', 'rpaddle']
prepair	['rivte', 'bhh', '1tunnel', 'iformation', 'nazeam']
prevent	['profile', 'soon', 'secured', 'become', 'unless']
reteste	['colletive', 'brief', 'fallback', 'pst', 'intend']
testiaw	['uunder', 'inspectiocomplete', 'met', 'illisheim', 'maggard']

testinf	['adhvesive', 'ratchting', 'checkreplacement', 'day5', 'hardwarde']
testing	['teca', 'amplifier', 'atg', 'amp', 'ciu']
testion	['lapto', 'shootingfm2', 'repairmultiple', 'fuslt', 'lipot']
testset	['eda', 'thoroughly', 'deselecte', 'endoscopic']
warped	['cu', 'deselecte', 'inus', 'impsa', 'iot']
annaul	['sd', 'reflector', 'eyecup', 'biannual']
annuel	['incompatable', 'missng', 'elar', 'adjusted2', 'reguarly']
assure	['chattering', 'replacedreaction', 'salve', 'repetitive', 'indiate']
insect	['5july', 'extente', 'balancetail', 'perorm']
inspct	['routere', 'unistalled', 'redrille', 'piv', 'remval']
inspec	['vg6', 'inspectionw', 'nco', 'transit', 'sat']
inspet	['lonb', 'footswitche', 'ndiphase', 'replacementpilot', 'manutfacture']
intial	['joh', 'classification', 'site', 'mmh', 'inital']
ispect	['shaped', 'cmwws', 'relaxed', 'ksli', 'r8dated8jul']
moistu	['intm', 'extinsible', 'metalrequeste', 'wornbeare', 'sli']
month	['bldg', 'rqrd', 'tub', 'overdue', 'week']
monthy	['bdate', 'overborad', 'illisheim', 'uunder', 'jsb']
montly	['acftpmd', 'goodic', 'incorrectley', 'insppreform', 'disperse']
nspect	['poly', 'reuqire', 'inoard', 'mfr', 'replacent']
preflt	['chaffed', 'beneath', 's1', 'sightglass', 'easily']
prefor	['positionleft', 'sybmbol', 'unscheduledleft', 'rotorengine', 'impeed']
qualty	['rotorengine', 'entryinsp', 'ohase', 'sstem', 'axis2']
reinsp	['stablization', 'scouring', 'rerequire', 'intermittitently', 'deficancie']
retest	['exe', 'differentiate', 'panell', 'replacedphase', 'umark']
screen	['blower', 'cbox', 'vent', 'behind']
tesion	['initialrev', 'maginifye', 'voiv', 'metlic', 'anttenas']
tested	['puck', 'jh', 'mw', 'ed', 'mcfadden']
tester	['tmde', 'jcb', 'ef', 'bii', 'internati']
testng	['hutcheson', 'nrts1fail', 'draq', 'emaill', 'gff5dtjc']
annua	['startsignal', 'airsource', 'aceptance', 'ratio', 'lugreset']
anual	['deselecte', 'inus', 'decontaminiation', 'baldor']
daily	['awrs', 'preventive', 'specify', 'bays', 'accordance']
eval	['abort', 'observe', 'fluctuate', 'nr', 'rise']
iinsp	['gt', 'phenelic', 'partsdisassemble', 'serrate', 'blabe']
injsp	['routte', 'cplamp', 'ginn', 'hoistpendant', 'antrotation']
inpsd	['duetoggle', 'linehyd', 'replacedmast', 'pclrequeste', 'ldds']

insop	['nrtspressure', 'complye', 'replacementbroken', 'excessivelly', 'storeroom']
insp	['a', 'x', 'upgrade', 'due', 'status']
inspc	['cmplete', 'conjution', 'complet', 'coplete', 'conjucntion']
inspe	['visualcomplete', 'baffel', 'completen', 'dinsp']
inspo	['xchange', 'initiation', 'brittle', 'unil', 'klesert']
insps	['pm', 'schd', 'unscheduled', 'defib']
inspt	['wtih', 'december', '6jul', 'conplete', 'inspection4']
inswp	['dcr', 'compliede', 'initiation', 'repairedload', 'tansmitter']
inti	['fnbx', 'dlk', 'intermentally', 'bushinglag', 'maintemance']
isnpc	['wtub', 'hanset', 'picuture', 'trial', 'lockscrew']
loadt	['zeuss', 'indictator', 'seterioration', 'sealand', 'outgoing']
moist	['appen', 'llg', 'bulbe', 'corrodde', 'stripppe']
phase	['reinst', 'facilitate', 'completion', 'removed', 'turret']
pmi1	['btwn', 'viscous', 'distributor', 'maintenace']
pmi2	['disassembly', 'distributor', 'btwn', 'depanel']
pmi3	['pmsacft', 'deamand', 'experice', 'prescribedin', 'apapche']
teset	['4f9quarterly', 'vernier', 'grtu', 'initate', 'popswhen']
test	['ok', 'good', 'turn', 'inop', 'reset']
teste	['lhrear', 'bushinglag', 'succeptabe', 'specifiy', 'leank']
testi	['reocket', 'incicator', 'g4', 'appt', 'johsnon']
tests	['barn', '3possible', 'repairedby', 'aircrraft', 'housing3']
tilt	['suspension', 'mdl', 'freightliner', '2fy']
aoap	['lab', 'abnormal', 'resample', 'content', 'ppm']
dail	['emg', 'jh6annual', 'waring', 'tiesday', 'iinstalle']
inps	['compleed', 'complette', 'completeed', 'duew']
ins	['faqult', 'egi2', 'fault6', 'compltete', 'adn']
isnp	['spiveyserviceability', 'tuckerserviceability', 'inspserviceability', 'mw', 'packserviceability']
nisp	['assemblyhole', '2a7', 'lonhgbow', 'imprperly', 'sercurity']
prep	['indirect', 'supportrequeste', 'nco', 'sats', 'olr']
prev	['tention', 'cel', 'conplete', 'cks', '5hrs']
ti	['technical', 'pid', 'a4', 'dump', 'wa']
te	['ra', 'yc', 'trc', 'classification', 'millimeter']
tesr	['spirometer', 'bootred', 'sofadd2', 'lsb', 'repairmultiple']
tess	['exercise', 'training', 'october', 'reconfiguration', 'instalation']
tid	['preflighte', 'colletive', 'orginal', 'incompatible', 'fg2']

ae	['bfa', 'biannual', 'fss', 'retae', 'handguard']
ann	['sd', 'djv', '4ag', 'reg']
hsc	['lmdbh', 'ecdh', 'indx', 'events', 'tti']
lti	['multiquip', 'internati', 'cor', 'inf', 'transcraft']
noe	['cleanse', 'touching', 'opertation', 'unistalle', 'pior']
nsp	['completeed', 'conjnction', 'compleed', 'preventitive']
pmi	['progress', 'disassembly', 'disassemble']
qc	['m1', 'recovery', 'school', 'records']
tet	['eej', 'lightbar', 'mounti', 'trash', 'apt']
til	['defrre', 'complette', 'cojunction', '3july', 'wih']
qa	['hard', 'reflect', 'copy', 'school']
nan	['complwte', 'reprovision', 'breare', 'completeq', 'watertight']

Department of Systems Engineering
United States Military Academy
West Point, New York 10096