3-2021

Contract Information Extraction Using Machine Learning

Zachary E. Butcher

CONTRACT INFORMATION EXTRACTION USING MACHINE LEARNING

THESIS

Zachary E. Butcher, Captain, USAF

AFIT-ENS-MS-21-M-145

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-21-M-145

CONTRACT INFORMATION EXTRACTION USING MACHINE LEARNING

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Zachary E. Butcher, BS

Captain, USAF

March 2021

AFIT-ENS-MS-21-M-145

CONTRACT INFORMATION EXTRACTION USING MACHINE LEARNING

Zachary E Butcher, BS

Captain, USAF

Committee Membership:

Dr. Jeffery D. Weir
Chair

Capt Phillip R. Jenkins, PhD
Member

Abstract

The Air Force Sustainment Center assisted by the Data Analytics Resource Team and the Defense Logistics Agency has collected four million contracts onto one of the Air Force Research Laboratory's high power computers. Many efforts are underway to utilize this new database. This thesis focuses on the effort to determine if parts are available through existing contracts. Parts are anything that is expendable, such as a B-52 tire or an annual JMP license.

To determine availability, this thesis implements the process outlined below:

a. Identify contracts containing parts and the part
b. Determine which organization initiated the contract
c. Determine which organization is supplying the part
d. Determine the contract start and end date
e. Discover details about the part, including name and category
f. Create easy to understand visualization of information

The process tasks are accomplished using a variety of tools and techniques. Some information is extracted from the contracts using machine learning (ML) in combination with natural language processing. Specifically, two types of ML models are used, named entity recognition (NER) and classification. Where ML methods are unsuccessful or inappropriate, more text mining techniques, such as pattern recognition and rules, are used. Upon completion, the information is combined into a Gantt chart for quick evaluation.

While each step in the process is generally successful, there is little overlap in successes. As a result, only 21% of the contracts have their information correctly extracted with this process. To provide an accurate depiction of availability for every part, an improvement is needed. There are several adjustments which could provide better results. The likely most effective improvement is to develop a custom NER model.

Acknowledgments

I would like to express my sincere appreciation to my advisor, Dr. Jeffery Weir, for his guidance. I would also like to thank Derrick Chiwinsky, from the DART.

Zach Butcher

Table of Contents

Page

List of Figures

List of Tables

CONTRACT INFORMATION EXTRACTION USING MACHINE LEARNING

I. Introduction

General Issue

The Air Force Sustainment Center (AFSC) executes hundreds of thousands of contracts each year. For the past decade, these contracts have been written under changing policies, in a myriad of formats, and stored in several disconnected systems. In the current climate of data capitalization, an effort is being made to bring together every contract to provide useful insights.

Objective

From the compilation of contracts several products are desired. The specific goal of this effort is to determine part availability. If a part is covered by an active contract, it is available, if the part is not, a contract would need to be created to provide said part. This goal requires several steps to accomplish. Those steps are outlined below:

1. Locate “parts” contracts
2. Extract relevant information from those contracts
3. Compile information by part

From this effort it can be determined if specific parts are currently covered by a contract. Due to the complexities of data retrieval, machine learning (ML) techniques are used in conjunction with natural language processing (NLP). The immense size of the data sets makes the use of high-powered computing necessary.

This paper provides a framework developed on a sample set of contracts. The framework is successful if it can be applied to several data sets and return the necessary

information. A proven framework could then be applied to every contract, accomplishing one of AFSC's primary goals.

In Chapter 2 the necessary background on each process in the framework is presented. In addition, similar efforts are reviewed, focusing on their similarities and success or failure. With the processes explained, their application to the project is detailed in Chapter 3. Chapter 4 reports the individual process successes and other findings. The implications of the entire project are then explored to determine if it is effective. Finally, the understanding obtained from this endeavor is summarized in Chapter 5 to provide value to the reader and AFSC.

II. Literature Review

Chapter Overview

The purpose of this chapter is to provide a description of NLP, discuss applications of ML to natural language, and highlight other's efforts along these lines.

Natural Language Processing (NLP)

NLP is a method of taking written language, such as a contract, and turning it into a format which can be quickly analyzed (Isahara, 2007).

The first step is to identify the item of interest. In this case, it will be the content of the contracts. This content is already somewhat structured, but not for ML techniques. As such, the content must be prepared for machine learning (Marinov & Efremov, 2019).

Some preparation is easy such as removing punctuation, stop words, and other unnecessary characters. Other processing, such as tokenization and lemmatization, are more difficult.

Stop words are high frequency words such as "I", "a", and "the" (Patel & Shah, 2013). These words routine usage causes them to have almost no significance. Removing them reduces noise and unnecessary processing. There is not a universal list of stop words. The advantage of this fact is task specific words such as "Air Force" can be considered stop words and removed to improve processing.

Tokenization is the process of breaking a larger entity down into smaller pieces, or tokens (Li, Ma, & Lee, 2013). In this case, converting the entire contract text into individual words to evaluate. One largely complex variable changed into many simpler variables. Once text is tokenized, processing and analysis is straight forward.

Lemmatization is the procedure of reducing a word to its root form or lemma (Han, Shen, Wang, & Liu, 2012). It involves removing inflections on a word such as the "d" on "united" to provide the base word "unite." It is more useful than stemming (a similar process) because lemmatization uses disambiguation to help reduce each word to its dictionary term. Using a dictionary provides a predefined vocabulary list, allowing for more consistent application.

Disambiguation is a product of syntactic analysis (Bessmertny, Platonov, Poleschuk, & Pengyu, 2016). It uses the context of the word to determine what it is. For example, it could infer if "bat" was the animal or the sport equipment. In general, syntactic analysis is any evaluation of a word's nature, placement, and/or use to provide more information.

Part-of-speech tagging is one component of syntactic analysis. It involves labeling, or "tagging," each word with the speech category it belongs to. Categories include verbs, nouns, and adjectives.

Dependency parsing is another component of syntactic analysis (Park & Kang, 2019). It evaluates how words are related to each other. Figure 1 shows the relationship of each word to the other words in a sample sentence.

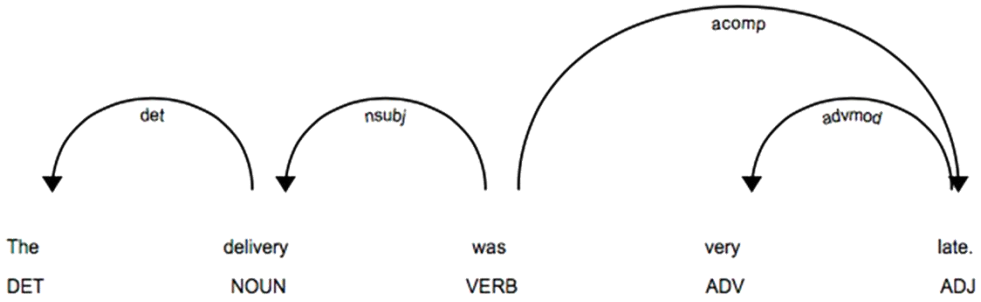

Figure 1. Demonstration of Word Relation and Identification

Once the words have been processed, ML can be applied. At this point ML can treat each word as a feature.

Machine Learning (ML)

ML uses features as inputs to train a model to produce correct outputs (James, Witten, Hastie, & Tibshirani, 2015). "Learning" in ML refers to the training portion of the process since the machine "learns" the most effective model.

There are several types of learning: supervised, unsupervised, reinforcement, and semi-supervised. In supervised learning, the outcome for a specific input is known. Given the inputs, the machine can learn/train a model to approximate the output. It can then apply the model to another input-output set, reinforcing what has been learned or forcing a change to the model to better accommodate all the data. If a change is needed, the change is determined by an optimization algorithm. The algorithm determines the adjustment(s) to the model needed to reduce the difference between the predicted outcome and the true outcome.

Classification is a typical application of machine learning (Bulbul & Unsal, 2011). It involves sorting entities into different groups, known as classes.

Deep Learning (DL)

DL is a subset of ML in the same way that ML is a subset of artificial intelligence (AI). It allows the computer to learn complex concepts from simple systems (Goodfellow, Bengio, & Courville, 2016). DL utilizes artificial neural networks (ANNs) which are capable of developing hundreds of connections and nodes. Conceptually it is difficult to explain what each element of the network does, but the aggregate allows complex understanding to be applied to every input.

Embeddings

DL can be used to create embeddings for each word. These word embeddings are vectors used to signify words while providing additional information (Goodfellow, Bengio, & Courville, 2016). They are referred to as distributed representations because each vector element provides context for the word. However, vector elements are similar to principal components, their meaning is not explicit. They are determined using neural networks with unsupervised learning.

The benefit of embedding is instead of matching a word by its characters, words can be compared on an element-by-element basis. This allows words with multiple meanings, such as “pass,” to be differentiated and similar words, such as “king” and “ruler,” to be associated.

Many embeddings are generated by Recurrent Neural Networks (RNNs) (Salim, Ghanshyam, Ashok, Mazahir, & Thakare, 2020). RNNs are neural networks that work for sequential data. This is important since sentences provide meaning through word

sequence. Unfortunately, simple RNNs are not enough since words can be referenced from multiple points within a sentence and differently from sentence to sentence. As such, a special RNN called a RNN-LSTM is used, where LSTM stands for long short-term memory. This RNN allows the model to remember how a word was used in a sentence at the beginning of a document as well as at the end.

ML, like many modeling techniques, considers more samples better than less. DL, unlike most modeling techniques, will continue to improve as the number of samples increases (Halevy, Norvig, & Pereira, 2009). As a result, the more samples that are provided, the better the model performs. Typically, embeddings are learned from a corpus with millions of words.

The learned embedding is a square matrix. A word is passed to it as a one-hot vector of all possible words; an example vector is in Figure 2. The result of multiplying the one-hot vector with the embedding matrix is an embedding vector for the word; example shown in Figure 3. This vector can then be compared to other word's vectors. It is important to note that the context represented by each element of the vector is notional and not able to be explained as completely as it is in the figure. Once the embeddings are generated, they can be used in models as a layer or preprocessing step. This entire mathematical process is summarized well in Figure 4.

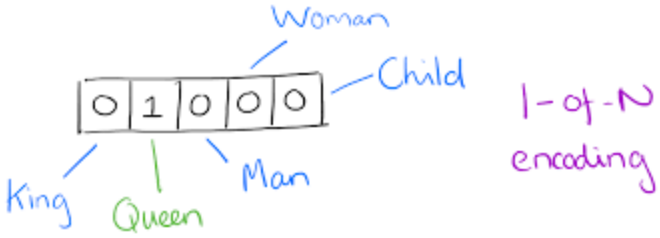

Figure 2. One-hot Word Vector Example (Ali, 2019)

Figure 3. Notional Word Embedding Example

$$\underbrace{\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}}_{\text{One-hot vector}} \times \underbrace{\begin{bmatrix}8 & 2 & 1 & 9\\ 6 & 5 & 4 & 0\\ 7 & 1 & 6 & 2\\ 1 & 3 & 5 & 8\\ 0 & 4 & 9 & 1\end{bmatrix}}_{\text{Embedding Weight Matrix}} = \underbrace{\begin{bmatrix}1 & 3 & 5 & 8\end{bmatrix}}_{\text{Hidden layer output}}$$

Figure 4. Embedding Vector Generation (Agrawal, 2019)

Named Entity Recognition (NER)

NER utilizes supervised, deep ML to classify words as specific entities (Shokripour, Anvik, Kasirun, & Zamani, 2013).

To conduct supervised learning, a set of words labeled with their corresponding entity must be provided. Entities include “person”, “place”, “date”, etc. This labeled dataset serves as the corpus. An example corpus is in Figure 5 (Terry-Jack, 2019).

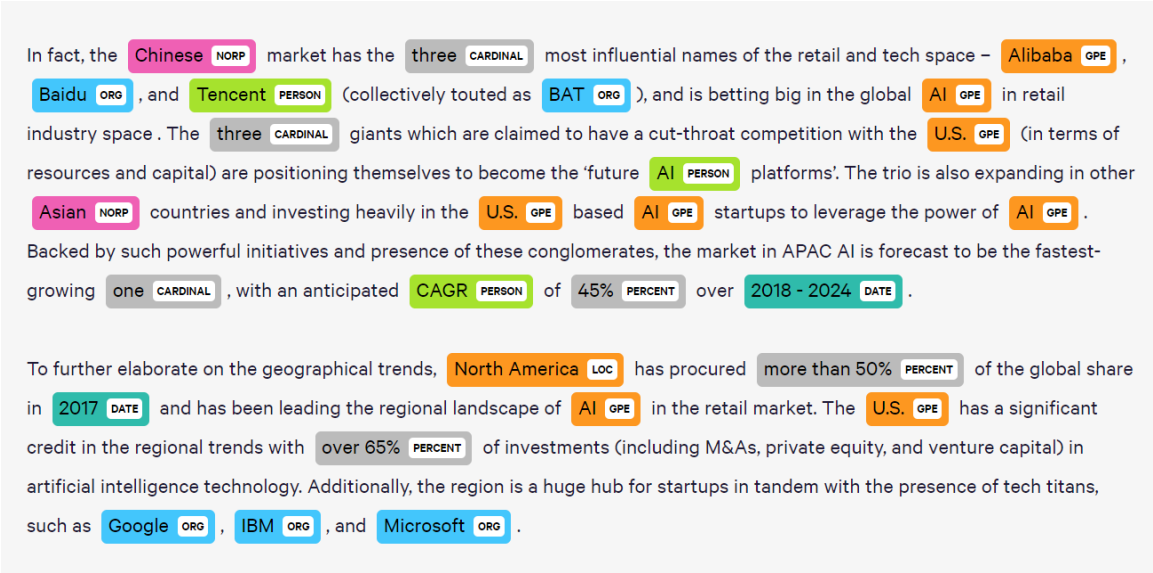

Figure 5. NER Demonstrated (Terry-Jack, 2019)

Labels are the backbone of NER. There are many pretrained NER models which can identify generic entities like person and place. However, for domain specific entities such as companies, diseases, and laws, a corpus with these specifically labeled is needed (Zhang, Lin, Gao, & Chen, 2019). Labeling a corpus is completed mostly by hand. Labeling millions of words takes a considerable amount of time. As such, acquiring the required corpus is not a trivial matter. There are several available for free. However, for more niche applications, appropriate corpus are usually purchased or generated.

To generate an NER model, each word in the corpus can be reduced to its lemma to greatly shrink the feature space. This can increase learning speed at the cost of accuracy; it is sometimes used but not necessary (Kutuzov & Kuzmenko, 2019). After the

corpus is prepared, embeddings can then be trained or pretrained embeddings downloaded. The obtained embedding allows new words to be compared to the corpus words. If a new word is found to be similar, the label from the corpus word can be applied to it. Ultimately this process allows entities to be classified/recognized.

Evaluation Metrics

ML does not provide a perfect model. As such, the model must be evaluated in some way to convey its worth. Standard metrics have been developed to evaluate models. For classification, the F1 score is the preferred metric (Zhang, Wang, Zhao, & Wang, 2015). The F1 score evaluates how well the predictions match the truth. Related terms are true positives, true negatives, false positives, and false negatives (Farhadloo & Rolland, 2013). True positive means the classifier was positive (true) and the prediction was positive, thus they were in alignment. Compared to false positive where the classifier was negative (false) but the prediction was positive. A confusion matrix helps identify this concept further. Though shown in Figure 6 as a 2x2 matrix, a confusion matrix can take on a more robust visage (Ariza-Lopez, Rodriguez-Avi, & Alba-Fernandez, 2018).

	Predicted Positive	**Predicted Negative**
Actually Positive	*True Positive*	*False Negative*
Actually Negative	*False Positive*	*True Negative*

Figure 6. General Confusion Matrix

Precision.

Precision measures how many predictions were correct out of all the positive predictions made (Avola, et al., 2019). It is useful when the costs of false positives are high. For example, a test should have high precision if those who test positive will receive a risky surgery. Only those that need the surgery should have it, any false-positives would be taking unnecessary risks.

$$\frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

Recall.

Recall measures how many were predicted positive out of the ones that were truly positive. It is valuable when the cost of false negatives is high. Recall is sometimes referred to as the detection rate (Wang, Li, Wan, & Wang, 2019). For example, a test should have high recall if it is used to identify infected patients. Failing to detect an infected patient could allow them to infect the rest of the hospital.

$$\frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

F1 Score.

F1 Score is the harmonic mean of precision and recall which means it gives much more weight to low values (Geron, 2017). As a result, it takes both high precision and high recall to obtain a high F1 score.

$$2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

Cross Validation

Cross validation is a technique applied during model generation, when an algorithm is applied to a data set (James, Witten, Hastie, & Tibshirani, 2015). Cross validation varies the samples, from a single data set, used to train and validate a model. This variation allows the same algorithm to develop several similar models instead of only one. The models are then aggregated to determine the expected outcome of the algorithm on the dataset.

A specific cross validation strategy is k-fold. In this approach the dataset is split into k equal parts. A model is then trained using all of the parts except one, reserving the excluded part for validation. This process is repeated until each part has served as the validation data, resulting in k models. Since each model trained and validated on different data, each produced a unique F1 score. The F1 score from each model can then be averaged together, producing an expected F1 score for the proposed algorithm on the dataset. This method is displayed in Figure 7.

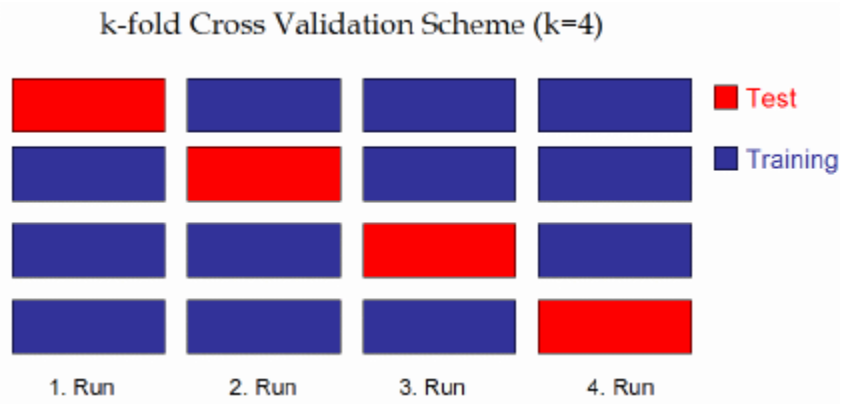

Figure 7. K-Fold Cross Validation (Bisgin, Kilinc, Ugur, Xu, & Tuzcu, 2011)

In addition to providing an expected F1 score, aggregating the models helps to mitigate the effects of a single model over or under fitting the data. Extremely sensitive/accurate models are said to overfit the training data. They learn the sample too well and lose generality the population exhibits. Alternatively, some models favor excessive generality and do not appropriately capture important traits in the training data. Averaging these notable models with others causes their extreme characteristics to become subdued.

Classification Models

Classification models utilize a set of features to predict which class the sample should belong to. There are many different classification modeling methods.

Logistic Regression.

One classification technique is logistic regression. This method uses the logistic function to split the decision space into a binary ruling (James, Witten, Hastie, & Tibshirani, 2015). This technique can be applied to multiple classes by using the concept of one against many; if the sample does not belong to the class of interest, it must be in one of the others. One significant advantage of this technique is it provides the probability that the sample belongs in the predicted class. It has one primary hyperparameter which is the solver it uses. The concept is displayed well in Figure 8.

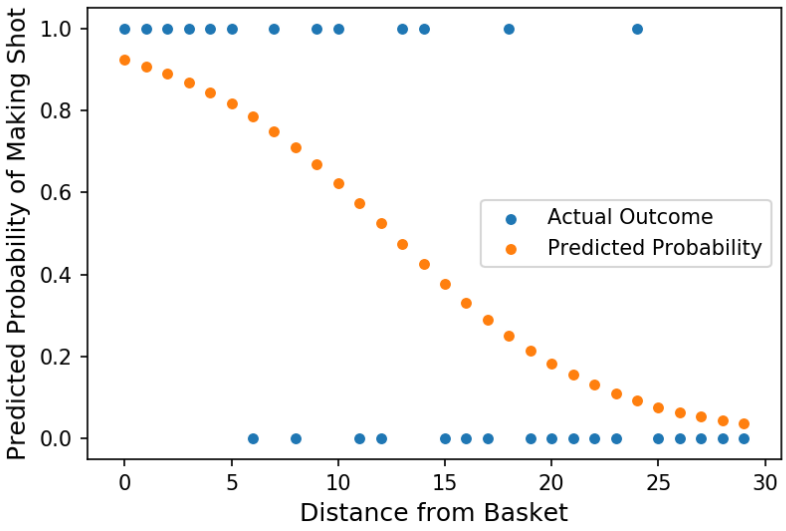

Figure 8. Logistic Regression Example (Yiu, 2019)

Support Vector Machine.

Support Vector Machine (SVM) is a classification method that utilizes an algorithmic approach. It develops models which try to increase the space between classes (James, Witten, Hastie, & Tibshirani, 2015). While a very robust modeling technique, it does not provide a probability of class inclusion. The two primary hyperparameters of SVM are C and kernel. The kernel function allows data to be projected into a higher dimension. This allows the arrangement of the data to be modified to better apply a hyperplane which separates the classes. C, also known as the regularization parameter, functions as the cost of a datapoint breaking that hyperplane, as the cost increases fewer and fewer misclassified datapoints are allowed. While a high cost sounds appealing, it makes the model unyielding and can cause lower success. SVM is demonstrated in Figure 9.

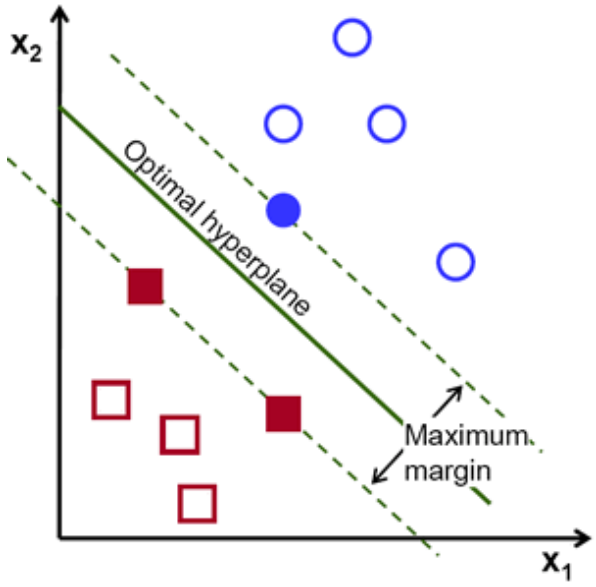

Figure 9. Support Vector Machine Example (Gandhi, 2018)

Random Forest.

Random forests are a collection of decision trees (James, Witten, Hastie, & Tibshirani, 2015). A decision tree is a series of decisions based upon feature values that branch and bound to a conclusion. A random forest generates many decision trees and randomly limits the features the decision nodes can use. In addition, each decision tree uses a unique training set created by bootstrapping. Bootstrapping randomly draws, with replacement, a specified number of samples from the original training set. These design characteristics cause different, independent decision trees to be generated. The ensemble of the many different trees is then leveraged to determine which outcome is the most common for a set of inputs. A primary hyperparameter of the random forest algorithm is the max tree depth. That is the maximum number of decisions it can make for a single outcome. Both decision trees and random forests as displayed in Figure 10, highlighting how a random forest is a collection of decision trees.

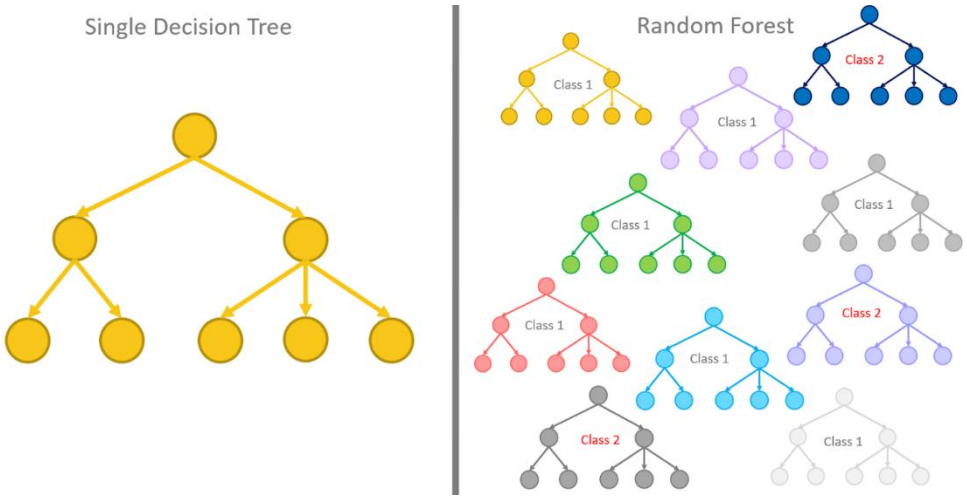

Figure 10. Random Forest Example (Silipo, 2019)

Relevant Research

There are off-the-shelf NER options available, the most common in Python being SpaCy and Stanford NER. SpaCy provides a pretrained model which is built on a generic entity corpus (Partalidou, Spyromitros-Xioufis, Doropoulos, Vologiannidis, & Diamantaras, 2019). The model can be adjusted by providing raw text to train a new embedding as well as provide additional labeled data to add domain entities (Honnibal, 2020). Stanford NER functions very similar to SpaCy except it is based in java and there is less of a distinction between the embedding and the entity recognition processes (Luthfi, Distiawan, & Manurung, 2014).

Research was conducted to determine if using the Stanford NER algorithm was as effective as training a custom model for generic entity recognition. The Stanford NER produced an F1 score of 66.97% while the custom NER scored 68.69% (Sotomayor & Veloz, 2017). Since this study there have been advancements in embeddings. Training a

custom NER using Embeddings from Language Models (ELMo) would likely perform better (Peters, et al., 2018).

As domain specific corpus can be very expensive to produce; efforts have been made to bypass this requirement. One such effort was to combine many free corpus from the internet (Menezes, Milidiú, & Savarese, 2019). The idea being with a vast training set, a model could become robust enough to handle niche data. Unfortunately, on its own, it did not perform as well as traditional efforts. A similar effort was applied to detecting legal entities (Ex. Laws, judgements, etc.) where the goal was to use legislative documents posted in the news, journal articles, and contracts (Badji, 2018).

It was found that a small corpus can be successfully paired with unsupervised learning to identify domain specific entities (Zhang, Lin, Gao, & Chen, 2019). The study examined words near the target word to determine the context of the word. If the context was similar to the training set, it would apply domain specific entity recognition. When compared to the classic domain-relevance-based entity recognition algorithm Concept-Relation-Concept Tuple-based Ontology Learning (CRCTOL), this method found 65% more entities. CRCTOL is used to determine ontologies from documents (Jiang & Tan, 2010). Ontology is a representation of a subject by determining its concepts and how they are related (Chandrasekaran, Josephson, & Benjamins, 1999).

Other improvements to NER have been focused on the embedding portion of the model. In 2015 a clinical corpus was used to develop the embedding and compared to the default embedding (Wu, Xu, Jiang, Zhang, & Xu, 2015). The result was an over 2% increase in the F1 score for the NER model. In another effort, Word2Vec was used as the embedding generator, and while increasing the size of the unlabeled corpus did improve

the embeddings, the improved embeddings failed to improve the NER performance (Siencnik, 2015).

Assuming an annotated corpus is obtained, there is still the decision on the best method to use. NER models can be built with various ANN architectures. A study applying NER to legal entities found that bidirectional LSTMs (bi-LSTM) out performed conditional random field (CRF) models in terms of F1 score (Leitner, Rehm, & Moreno-Schneider, 2019).

Applying NLP to contracts is not a new endeavor. In fact, at the *Conference on Data Science and Machine Learning Applications* on May 4th, 2020, the use of NLP on contracts was discussed (Kim, Lee, Lee, & Lee, 2020). The specific application discussed was predicting the costs associated with engineering design. In another study, insurance policies/contracts content was automatically analyzed and tagged to automate the process of finding relevant contracts (Zhang, Sun, & Ji, 2019).

Application of NER on contracts has become a commercial venture. A company called Skyl.AI helps organizations do many kinds of ML, including NER (Named Entity Recognition, 2020). Specifically, it has helped insurance companies and real estate companies do contract content analysis. Google is also involved; their cloud environments offer NLP, to include NER (Natural Language, 2020). They offer generic entity extraction as well as domain specific entity extraction. Unfortunately, both options require a labeled corpus for domain specific entity extraction. They do provide tools to assist with labeling, but the labor burden is still on the customer.

Preparation Steps

Source.

Approximately four million contracts were loaded onto one of the Department of Defense's (DoD) high power computers (HPCs). The HPC provided a secure environment to host the protected data. It also possesses the resources to effectively work with large data sets. In addition, the HPCs are partially funded, allowing any DoD organization to use them for free to a certain extent.

The original contracts consisted of PDFs, scanned images, and word documents. In addition to the type of file, there were inconsistent layouts, tags, and other aspects of the documents. The Data Analytics Resource Team (DART) converted them all to plain text files.

Access.

There are two methods primarily used to access the data. The first utilizes a program optimized by the HPC resource team called iLauncher (iLauncher v1.10 Downloads, 2020). This method provides access to Python, a virtual desktop, and a few other useful tools, all within the user's workspace. It is the simplest way to begin an HPC session. The second method can be completed on its own but is usually done proceeding the first method to provide additional capabilities. It utilizes Putty, a secure shell (SSH) client, to develop a secure connection directly to the server (Download Putty, 2020). This connection allows access to any directory the user is authorized and gives the ability to execute Linux commands. The combination of these methods was used for this project. Putty was used to copy documents to the user's workspace where they could be accessed by iLauncher's Python instance.

Environment.

To utilize Python programming on the HPC several unique packages are needed. Python utilizes too many packages to have all of them available initially. Instead, they are downloaded and installed as needed. To host the installed packages a custom environment must be created, this prevents the standard environment becoming overwhelmed with mostly unused packages. The server installation of Jupyter Notebook has a conda (as in Anaconda) module which facilitates simple environment management. From this tool environments can be created, duplicated, and deleted. This tool also allows a list of known packages to be added to an environment. For packages not known to the environment manager, a script can be run within a Jupyter Notebook to download and install the necessary packages.

Instance.

The instance on which the development environment is built is one of the HPC standard user instances. It accesses 24 cores and 126 gigabytes of virtual memory. Processes are executed with standard priority, a mid-level provision.

III. Methodology

Chapter Overview

This chapter details the processes employed to achieve the goal of determining part availability. Each process is explored in depth in the same order in which it is utilized, displayed in Figure 11. The methodology is conducted on a sample of contracts for the study, however, the tools developed could be applied to the entire data set.

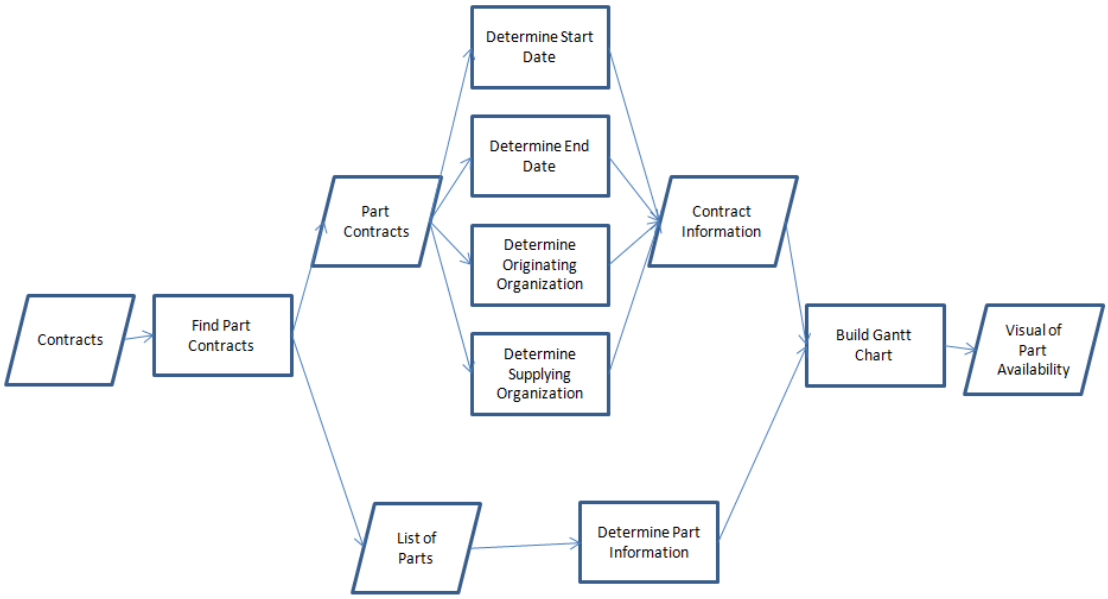

Figure 11. Flowchart of Methodology

First, part contracts are found using pattern matching. Then NER models are applied to the contract text to identify date and organization entities. Those entities are then fed to classification models to determine which entities are relevant for the study. Any information not acquired through this method is located with text mining. While contract information is collected, part information is also gathered. Once both portions

are complete their products are combined. This useful combination is ultimately displayed in a Gantt chart for ease of understanding.

A Gannt chart represents time on the horizontal axis and different items on the vertical axis. The period of time relevant to each item is represented on the graph by a horizontal bar. This allows quick determination of when an item begins and ends, as well as any time segments during which items overlap. An example Gantt chart is shown in Figure 12.

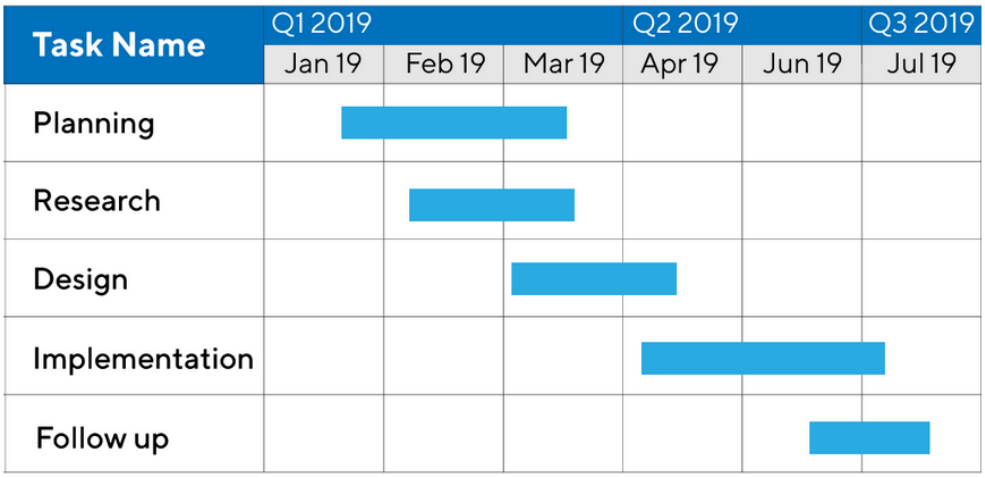

Figure 12. Example Gantt Chart (Gantt Chart, 2021)

Datasets

To develop and validate the methodology a sample from the contract database is used. This sample contains approximately 1000 of the 4 million contracts. Two additional samples are used to examine the success of applying the approach to new data. These samples are provided by the DART team from extracts of their on-going work. They are referred to as the initial data set, first test set, and second test set.

The initial data set contains 999 contracts. The contracts range from 2 to 149 pages, averaging 8.6 pages. A histogram of the number of pages is in Figure 13. The first test set contains 1166 contracts, and the second text set contains another 999 contracts. The data sets are compared in Table 1 below.

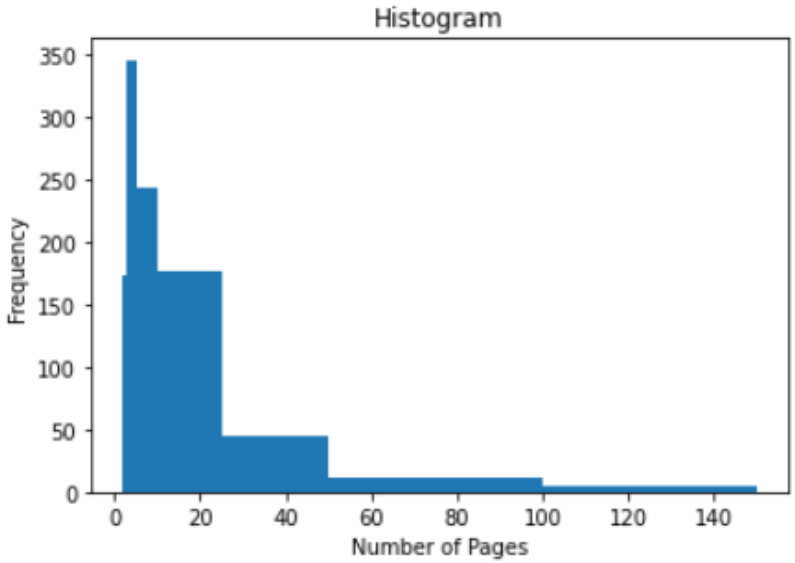

Figure 13. Histogram of Contract Page Counts

Table 1. Datasets Summarized

	Initial Data Set	First Test Set	Second Test Set
Number of Contracts	999	1166	999
Min # Pages	2	1	1
Average # Pages	8.6	9.7	7.6
Max # Pages	149	205	134

Part Contract Identification

To locate part-buy contracts within the sample, an approach known as Regular Expression is used. Each part has a national stock number (NSN). This NSN is always in

the format ####-##-###-####. Each document's text is searched for any characters that match this pattern. An example of a found NSN is in Figure 14.

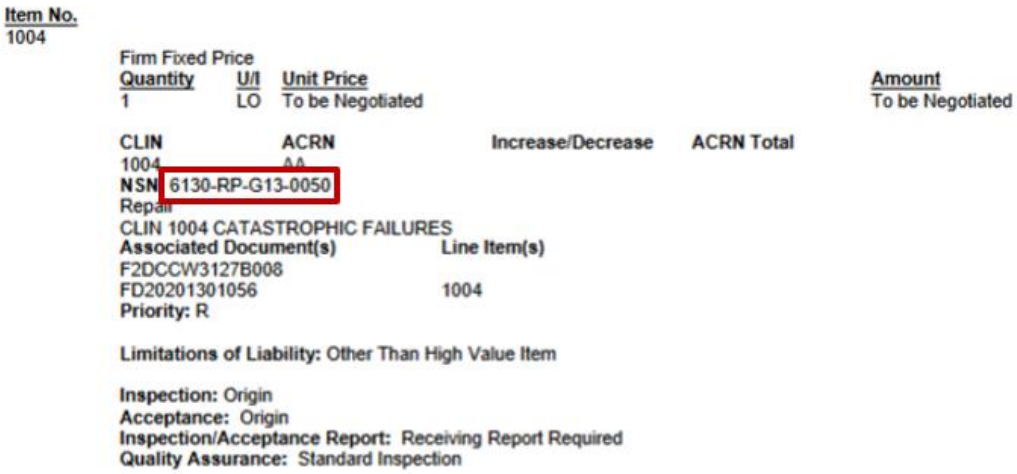

Figure 14. Example of a Found NSN

Contract Information Extraction

With a set of part-buy contracts identified, named entity recognition is used to determine each contract's originating organization, supplying organization, origination date, and expiration date.

To determine the entities in each document, SpaCy's large English model is used. This umbrella model contains an NER model, an embedding model, and several others. The NER model identifies several entity types including organization, dates, and money. To utilize any of the SpaCy models, a document must first be converted into a SpaCy object. Once completed, the NER model is applied to return entities of a desired type. The result is a long list of suspected entities. However, since this is a predictive model, not all entities are valid. To identify the useful entities within this list, another model is applied.

Evaluating the relevancy of an entity is accomplished with a binary classification model. To construct a classification model a training set containing features and responses must be provided. While the words that represent the entity could provide a feature set, they are unlikely to provide enough information to determine relevancy. Instead, the entity plus several words around it forms the feature set. This sentence fragment provides enough information for decision making.

Each word in the sentence fragment is converted to a numeric representation, here forth referred to as an embedding vector, by an embedding model. For this study, the embedding model delivered with the large, English SpaCy model is used. Once each word is converted to an embedding vector, the vectors are averaged together to produce a single vector. This single vector represents the sentence fragment. Each element of the vector serves as a feature.

Currently there is not an automatic method to classify, nor an existing classified data set. As such, a subset of sentence fragments and their corresponding entities is manually evaluated to be “relevant” or “not relevant.” This provides the data set necessary to create the classification model.

Before any modeling efforts begin, a portion of the classified data set is set aside for testing. This sequestered test data is later used for an impartial evaluation of the developed models. A 20% test/train split is used. The practice of using 20% for the split is based on the Pareto Principle and is widely used as a starting point (Detective, 2020).

The resulting training set is not balanced since most entities are irrelevant. Imbalanced classes may create inappropriate, biased classification models. To mitigate this a corrective sampling technique called SMOTE is applied. SMOTE stands for

Synthetic Minority Oversampling Technique and belongs to the Imbalanced-learn python package (Lemaître, Nogueira, & Aridas, 2017). It over-samples small classes and can under samples large classes to create a balanced ensemble. The result is a much larger, balanced training set which maintains the characteristics of the original data.

With a balanced training set, classification methods are examined. There are many binary classification methods. As such, instead of examining only a few, 30 of the most popular can be evaluated using a python package called lazypredict (Pandala, 2020). Lazypredict applies models with default settings on a sample set. The results identify which modeling techniques are likely capable of producing the best results. From these results, a final modeling technique can be selected, tuned, and trained.

Named entity recognition in combination with classification modeling produces meaningful extracts from the documents. This approach returns a single entity type. To return each entity type, the entire approach must be applied over and over until all instances are accounted for. This requires manual data classification and model evaluation for each repetition.

For information of interest in which the previous method fails, text mining techniques are used instead. These methods are typically based on a series of rules that look for patterns or exact matches within a document. Using these methods is not ideal which is why NER is applied first. The rules employed work well for observed cases but almost never apply well to others. This results in a rigid model, unsuited to different style documents.

Examining the list of organizations (“ORG” entities) found, it rarely contains the originating organization. This is likely due to the unique names of the originating

organizations, such as 88th ABW, which the NER model has not seen before. As such, using NER for this goal is not advised, instead text mining is used.

Part Information Extraction

Beyond the information determined by NER methods, other aspects of the contracts are desired such as part name. Part information is obtained from https://www.iso-group.com/. A specific part is located by adjusting the web address to include the NSN, such as https://www.iso-group.com/NSN/3830-01-352-6260. Each NSN page utilizes an identical layout, which can be seen in Figure 15. With the use of a popular web scraping tool called Beautiful Soup, this strict layout is leveraged to successfully extract information from the webpage (Richardson, 2020). Beautiful Soup is a powerful python library which handles webpage encoding and parsing. It also provides several functions that assist users in quickly and efficiently manipulating webpages.

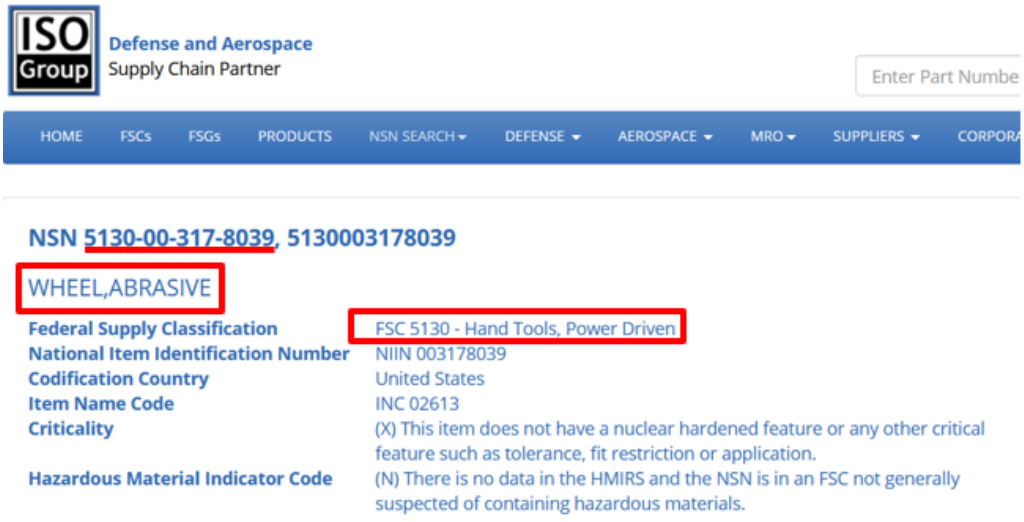

Figure 15. Part Webpage Example

Part Availability

With relevant information extracted from each contract, analysis can be performed. One of the primary items of interest is part availability.

The first concern of part availability is if a part is available in current supplies. There are databases that manage supplies where this can be determined. If a supply is available, it can be sourced. If a supply is not available, the next concern becomes are there any contracts that can be used to produce more supplies.

The contracts can be searched for a specific part however there could be many contracts for a single part. Once contracts are identified they must still be examined to determine if they are currently valid. To alleviate this issue, a Gantt Chart can be created to track the periods in which a part has an active contract. With this tool, a user can quickly determine if a part is covered, how long until a part contract expires, or which contract covers the part. To make the most use of this tool, it could be used to determine which parts are currently covered by an active contract but soon will not be. This creates a focus list for future contract efforts to ensure part availability.

Summary

Accessing contract data is not a straightforward affair. The data is sensitive which requires it be housed in a secure environment. The HPC provides such an environment, but the server architecture of the HPC evokes additional effort. Specific access must be granted, unique tools utilized, and additional coding implemented to work on a server. Once access to the contracts is established in a useful coding environment, the main effort can begin.

Using multiple techniques, important information can be extracted from a contract. NER models in combination with classification models can be used to determine important characteristics such as the organization supplying the contract as well as the date the contract expires. Some important features require more rudimentary techniques, such as text extraction following a set of rules. Methods such as this can be used to determine the organization that generated the contract and other features that do not perform well with other methods. In addition to extraction, some features can be used to retrieve additional information. One such example is using the part number (extracted using a simple pattern locator) with robust webpage scraping tools to search the internet for additional information about the part. This new information can then be added to the contract's summary.

Information is only part of the process, it must be leveraged to provide utility. Using elements of the extraction such as part number and contract end date, a Gantt Chart can be created. This simple tool allows quick discovery of parts with active or soon to be ending contracts.

IV. Results and Analysis

Chapter Overview

This chapter details the results of implementing the methodology. Application to three sample sets is presented and analyzed.

Initial Results

This section details the results of applying the methodology on the initial data set. Using Regular Expression, 301 of the 999 contracts from the initial data set are identified as part contracts. These 301 contracts are used for the remainder of this section.

To facilitate NER, each contract is converted to a SpaCy object. This process is nearly instantaneous, taking less than 0.01 seconds per contract. The entities of interest for NER are the organization fulfilling the contract, the date the contract became effective, and the date the contract expired. The supplying organizations is found by using the "organization" SpaCy NER model. Likewise, the effective date and end date are identified by the "date" model. The immediate benefit of this is that only two models are run on the documents. Each model has a unique application time, summarized in Table 2. Table 2 also shows the number of entities found.

Table 2. NER Model Results

	Organization Entities	Date Entities
Average Run Time Per Document	0.27 seconds	0.57 seconds
Average Number of Entities Found Per Document	118.8	47.6

The output from the "organization" NER model contains many of the supplying organizations. With this success, a subset of 10 documents is collected. Each document is manually examined to determine the supplying organization. Once identified, the

supplying organization is located in the list of entities returned for the document and classified as “relevant”; all other returned entities are classified as “irrelevant”. This produces a classified data set with 301 irrelevant entities and 17 relevant entities. Such a manual process takes time and skill to be reliable; for this application it took roughly an hour to complete.

Utilizing SMOTE, a final data set of 927 irrelevant entities and 927 relevant entities is created. This step corrects the imbalanced classes. To create the test set, 20% of the data (371 samples) is set aside. The other 80% (1483 samples) is passed to LazyPredict for classification model determination. The LaztPredict functions takes approximately 12 seconds to execute. The results of the LazyPredict application are below in Table 3.

Table 3. LazyPredict Output

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
AdaBoostClassifier	1.00	1.00	1.00	1.00	3.10
RandomForestClassifier	1.00	1.00	1.00	1.00	1.25
ExtraTreesClassifier	0.99	0.99	0.99	0.99	0.35
XGBClassifier	0.99	0.99	0.99	0.99	0.22
SVC	0.99	0.99	0.99	0.99	0.19
SGDClassifier	0.99	0.99	0.99	0.99	0.05
BaggingClassifier	0.99	0.99	0.99	0.99	1.41
LGBMClassifier	0.99	0.99	0.99	0.99	0.48
ExtraTreeClassifier	0.99	0.99	0.99	0.99	0.04
DecisionTreeClassifier	0.99	0.99	0.99	0.99	0.31
LogisticRegression	0.99	0.99	0.99	0.99	0.09
CalibratedClassifierCV	0.99	0.99	0.99	0.99	0.82
Perceptron	0.99	0.99	0.99	0.99	0.05
NuSVC	0.99	0.99	0.99	0.99	0.92
RidgeClassifierCV	0.99	0.99	0.99	0.99	0.08
RidgeClassifier	0.99	0.99	0.99	0.99	0.17
LinearSVC	0.98	0.98	0.98	0.98	0.20
LinearDiscriminantAnalysis	0.98	0.98	0.98	0.98	0.64
PassiveAggressiveClassifier	0.98	0.98	0.98	0.98	0.06
LabelSpreading	0.97	0.97	0.97	0.97	0.26
LabelPropagation	0.97	0.97	0.97	0.97	0.24
QuadraticDiscriminantAnalysis	0.96	0.96	0.96	0.96	0.14
GaussianNB	0.96	0.96	0.96	0.96	0.04
NearestCentroid	0.95	0.95	0.95	0.95	0.04
KNeighborsClassifier	0.95	0.95	0.95	0.95	0.35
BernoulliNB	0.92	0.93	0.93	0.92	0.28
CheckingClassifier	0.49	0.50	0.50	0.32	0.03
DummyClassifier	0.49	0.49	0.49	0.49	0.04

Table 3 shows the algorithm used, the time taken to generate a model, and the success of that model in the form of its F1 score. The higher the F1 score the better the model performs as a classifier, which is why the results are arranged in descending order by this statistic. The highest F1 score is achieved with the adaptive boost algorithm, however, it took over 3 seconds to run, more than double the next longest application and far longer than almost every other algorithm. For this reason, it is not considered a good

option. The next highest F1 score is the random forest algorithm. There are no apparent issues with this option, so it is selected as the best option to proceed with.

A random forest model is tuned by adjusting one of its primary hyperparameters, "maximum tree depth." In addition, 10-fold cross validation is used to ensure a more biased, lower variance model. Seven values for the hyperparameter are evaluated: 1, 2, 3, 5, 10, 25, and 50. Figure 16 shows the results. Ten produced the highest F1 Score of 0.998 on the validation data. Applying the model with a max depth of 10 to the test data yields an F1 Score of 0.997. This high rate of success is considered appropriate to apply the model further.

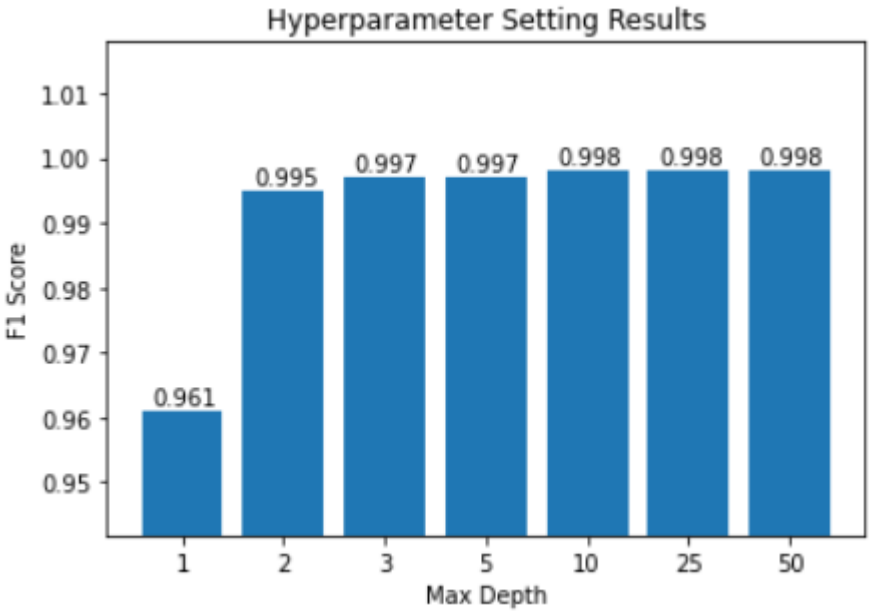

Figure 16. Random Forest Max Depths' F1 Scores

This classification model is applied to the list of organizations returned by the NER model for each document. Of the 301 documents, 122 successfully have their

supplying organization found by applying the two models in sequence. The entire process to determine supplying organization is summarized in Figure 17.

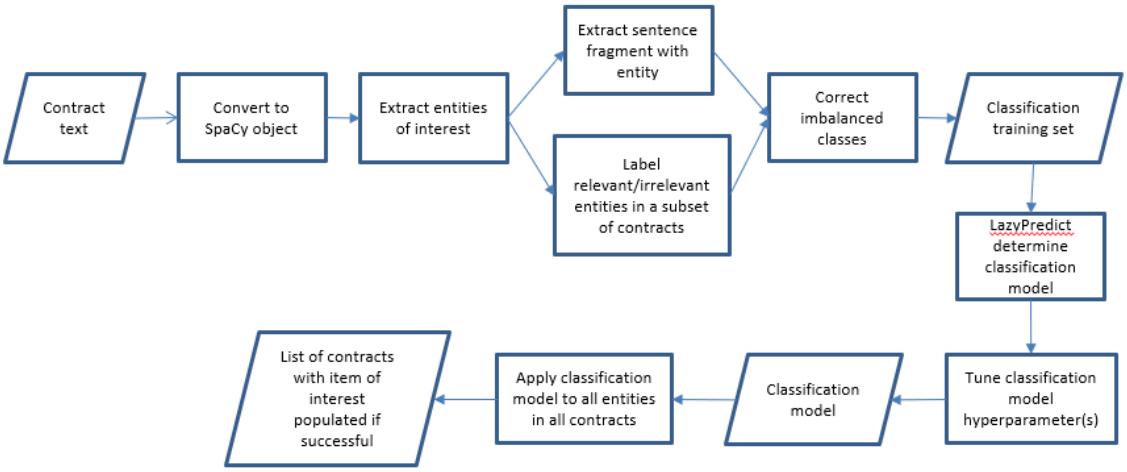

Figure 17. Process to Extract Information with NER and Classification Modeling

The process in Figure 17 is also used to determine the effective date and end date for each contract. Table 4 and shows the LazyPredict outputs for each process. For the contract effective date, logistic regression is one of the most promising classification models. The only hyperparameter to tune in the logistic regression algorithm is the solver. Three of the most common solvers are explored: liblinear, Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (LBFGS), and newton-conjugate gradient (newton-cg) (Hale, 2019). Liblinear improves one variable at a time, seeking an optimal solution (minimum cost/loss) by looping through the next most promising variable. LBFGS uses gradient evaluation to estimate the update needed to minimize the cost. Newton-cg computes the second derivative of the cost function to determine the update

needed to minimize the cost. Figure 18 shows the F1 score for a model created with each solver, indicating liblinear as the best solver.

For the contract end date, SVM is likely the most effective classification method. SVM has two hyperparameters to tune, the kernel and C. The two kernels explored are linear and radial basis function (RBF). It is best practice to check if data is linear, which leads to using the linear kernel (Zoltan, 2018). RBF is a general kernel that is commonly used when nothing is known about the data (SVM Kernel Functions, 2021). The values for C explored are based on 8 log scale steps from 0.01 to 100, resulting in 8 values to explore. It is difficult to predict what level of regularization will be needed, thus very small and very large values of C must be explored. To efficiently move through such a vast range, log steps are ideal. For this instance, 8 steps break up the test space well. Figure 19 shows the F1 score for each hyperparameter combination. The linear kernel with a C of 7.2 performed the best.

Table 4. LazyPredict Output for Contract Effective Date

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
ExtraTreeClassifier	1.00	1.00	1.00	1.00	0.02
RidgeClassifierCV	1.00	1.00	1.00	1.00	0.05
LogisticRegression	1.00	1.00	1.00	1.00	0.06
Perceptron	0.99	0.99	0.99	0.99	0.03
PassiveAggressiveClassifier	0.99	0.99	0.99	0.99	0.03
LGBMClassifier	0.99	0.99	0.99	0.99	0.15
XGBClassifier	0.99	0.99	0.99	0.99	0.10
SVC	0.99	0.99	0.99	0.99	0.04
DecisionTreeClassifier	0.99	0.99	0.99	0.99	0.05
ExtraTreesClassifier	0.99	0.99	0.99	0.99	0.18
RandomForestClassifier	0.99	0.99	0.99	0.99	0.32
BaggingClassifier	0.99	0.99	0.99	0.99	0.21
NuSVC	0.98	0.97	0.97	0.98	0.07
CalibratedClassifierCV	0.98	0.97	0.97	0.98	0.23
SGDClassifier	0.98	0.97	0.97	0.98	0.03
GaussianNB	0.98	0.97	0.97	0.98	0.03
LinearSVC	0.96	0.97	0.97	0.96	0.07
RidgeClassifier	0.95	0.96	0.96	0.95	0.03
AdaBoostClassifier	0.95	0.95	0.95	0.95	0.67
BernoulliNB	0.95	0.95	0.95	0.95	0.04
LabelSpreading	0.93	0.93	0.93	0.93	0.04
LabelPropagation	0.93	0.93	0.93	0.93	0.04
LinearDiscriminantAnalysis	0.91	0.91	0.91	0.91	0.07
NearestCentroid	0.91	0.91	0.91	0.91	0.02
KNeighborsClassifier	0.91	0.91	0.91	0.91	0.05
QuadraticDiscriminantAnalysis	0.47	0.50	0.50	0.30	0.05
CheckingClassifier	0.53	0.50	0.50	0.37	0.02
DummyClassifier	0.36	0.37	0.37	0.36	0.02

Table 5. LazyPredict Output for Contract End Date

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
AdaBoostClassifier	1.00	1.00	1.00	1.00	0.50
XGBClassifier	1.00	1.00	1.00	1.00	0.07
SVC	1.00	1.00	1.00	1.00	0.04
QuadraticDiscriminantAnalysis	1.00	1.00	1.00	1.00	0.04
ExtraTreesClassifier	1.00	1.00	1.00	1.00	0.18
BaggingClassifier	0.99	0.99	0.99	0.99	0.16
SGDClassifier	0.99	0.99	0.99	0.99	0.03
RandomForestClassifier	0.99	0.99	0.99	0.99	0.27
PassiveAggressiveClassifier	0.99	0.99	0.99	0.99	0.03
LogisticRegression	0.99	0.99	0.99	0.99	0.04
LinearSVC	0.99	0.99	0.99	0.99	0.09
CalibratedClassifierCV	0.99	0.99	0.99	0.99	0.45
DecisionTreeClassifier	0.99	0.99	0.99	0.99	0.05
Perceptron	0.98	0.98	0.98	0.98	0.03
ExtraTreeClassifier	0.98	0.98	0.98	0.98	0.02
RidgeClassifierCV	0.98	0.98	0.98	0.98	0.04
RidgeClassifier	0.98	0.98	0.98	0.98	0.03
LabelSpreading	0.98	0.98	0.98	0.98	0.04
LabelPropagation	0.98	0.98	0.98	0.98	0.04
LGBMClassifier	0.97	0.97	0.97	0.97	0.11
KNeighborsClassifier	0.95	0.96	0.96	0.95	0.05
GaussianNB	0.95	0.95	0.95	0.95	0.02
NearestCentroid	0.93	0.93	0.93	0.93	0.02
NuSVC	0.93	0.93	0.93	0.93	0.07
BernoulliNB	0.92	0.92	0.92	0.92	0.03
LinearDiscriminantAnalysis	0.90	0.90	0.90	0.89	0.05
CheckingClassifier	0.52	0.50	0.50	0.36	0.02
DummyClassifier	0.47	0.47	0.47	0.46	0.02

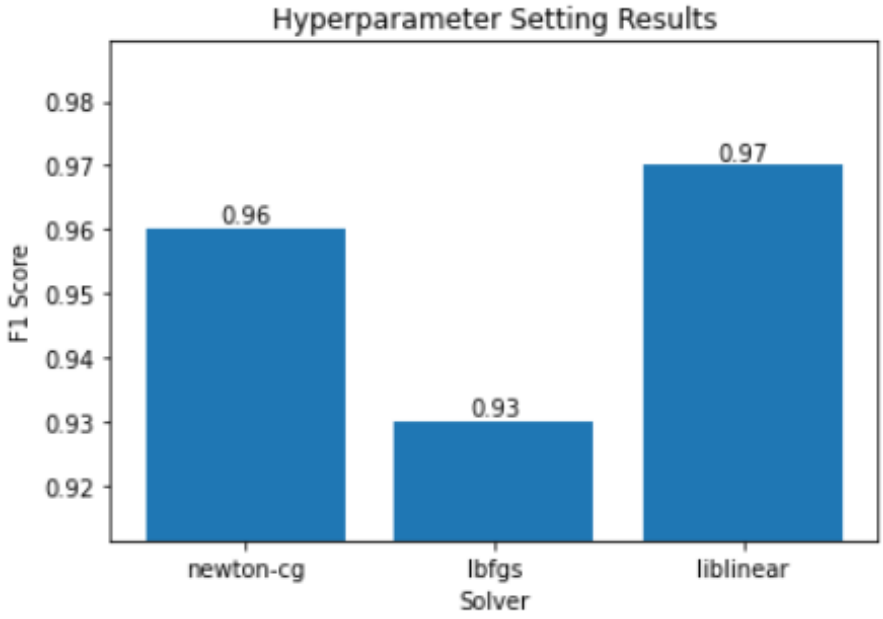

Figure 18. F1 Scores for Logistic Regression Hyperparameter Settings

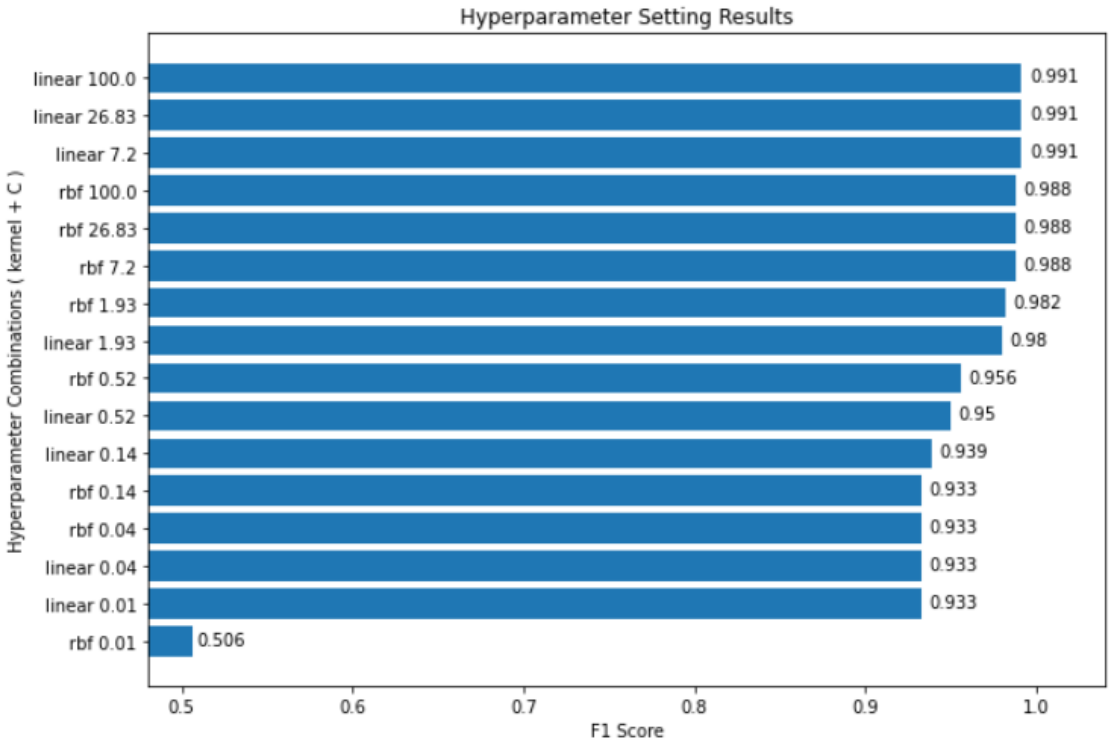

Figure 19. F1 Scores for Support Vector Machine Hyperparameter Settings

With those classification models successfully developed, three items of interest are covered. A summary of the three entity recognition efforts is in Table 6. Now focus can shift to resolving the other items of interest.

Table 6. Results of NER with Classification

	Supplying Organization	Effective Date	End Date
Number of Documents in Manually Classified Set	15	10	10
Number of Relevant Entities Found	17	9	22
Number of Irrelevant Entities	927	211	214
Number of Samples after SMOTE	1854	422	428
LazyPredict Best Classifier	Random Forest	Logistic Regression	Support Vector Machine
LazyPredict F1 Score	1.00	1.00	1.00
Tuned Hyperparameter	Max depth = 10	Solver = liblinear	Kernel = linear C = 7.2
Tuned Validation F1 Score	0.998	0.97	0.991
Tuned Test F1 Score	0.997	0.98	0.988
Number of Documents with Relevant Entities Found	122	181	96
Percent of Documents Successfully Extracted	40.5%	60.1%	31.9%

In addition to information about the contract, information about the part(s) in each contract is also desired. While many contracts contain only one NSN, several have more; in some cases, up to 312 in a single contract. In instances where there are multiple NSNs, usually some NSNs are repeated. To eliminate repetitive, useless information retrievals, repeated NSNs in a contract are eliminated. Once refined, the list of NSNs in a contract is used to determine part information.

A website, https://www.iso-group.com, provides part information. Each part's webpage contains the NSN in the URL. As such, modifying the URL to contain the NSN

of interest successfully loads that NSN's webpage. Once loaded, information is scraped using Python's Beautiful Soup package. The name of the part identified by the NSN as well as the category to which the part belongs is obtained. Repeating this process for each of the 459 parts successfully gathers information on 457 of them. As a result, every one of the 301 parts contracts has some information added.

To obtain the contract originating organization, text mining is implemented. However, before applying this approach, an issue must be resolved. Many contracts appear to be similar but have different raw text layouts due to how they were extracted. To correct this, the original PDF of the contract is converted into raw text using the PyPDF2 package (Phaseit Inc., 2016). This new raw text serves as the medium for the current approach.

To employ text mining, specific words which precede the originating organization are identified. From these, if an end word is located within 100 characters, a nearly perfect extract occurs by isolating the string between the start and end word and splicing based on spaces in the string. However, in the absence of an end word, the 100-character string is stripped where excessive spaces are found, also producing a useful extract. This approach provides consistent results. Of the 301 contracts, 275 have their originating organization successfully identified.

All the information collected is maintained in separate files. This is primarily due to the need to keep processes separate for ease of implementation and to protect the outputs. To make a useful, final product, the files must be combined. The resulting product contains a list of contracts, their server location, the parts they contain (including

NSN, name, and category), originating organization, supplying organization, effective date, and end date. Of the 301 contracts, 30 have all fields successfully populated.

Contracts with effective date and end date populated are used to determine part availability. For each part, all contracts containing the part with both dates populated are collected. The dates are then combined to determine periods of contract coverage. The periods of coverage for each part are then plotted on a timeline, creating a Gantt chart. An example Gantt chart covering five parts is in Figure 20. Part 7540-01-152-8070 is shown to be covered by multiple contracts; each change in color represents a new contract. While different contracts provide access to the part, there is no break in access; where one contract ends, another picks up. From this example it is easy to determine when a contract is open for a particular part.

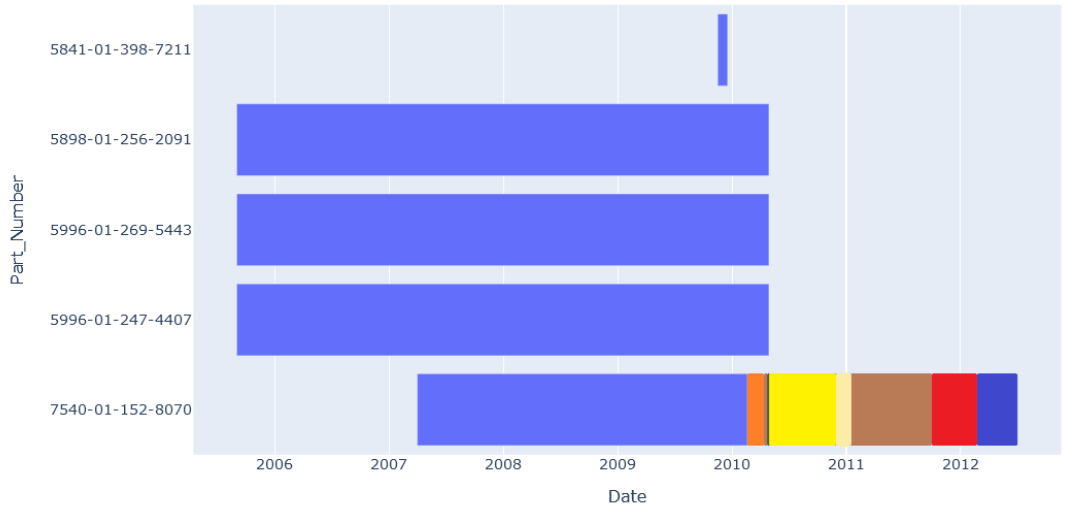

Figure 20. Example Chart of Parts Covered by Contracts

Additional Results

With the successful implementation of the methodology on a single data set, efforts are taken to apply the product to additional data sets. Each coded portion and its resulting product are combined into a single batch file. This allows the technique to be easily applied to any similar data set. Two additional data sets are acquired for testing. The results of all three applications are summarized in Table 7.

Table 7. Results from Multiple Runs

	Initial Data Set	First Test Set	Second Test Set
Number of Contracts	999	1166	999
Found Part Contracts	301	592	80
(Time to run)	*27 sec*	*16 sec*	*16 sec*
Found Originating Organization	275	559	55
(Time to run)	*6 sec*	*13 sec*	*6 sec*
Found Supplying Organization	122	153	4
(Time to run)	*9 min, 34 sec*	*26 min, 36 sec*	*2 min, 18 sec*
Found Effective Date	181	157	9
(Time to run)	*5 min, 32 sec*	*14 min, 40 sec*	*1 min, 3 sec*
Found End Date	96	155	5
(Time to run)	*5 min, 35 sec*	*14 min, 31 sec*	*1 min, 3 sec*
Parts (Information Found)	457	381	8
(Time to run)	*6 min, 49 sec*	*3 min, 41 sec*	*6 sec*
Contracts with All Features Found	30	19	0
Contracts Sufficient for Gantt Chart	62	57	2
Parts in Gantt Chart	323	57	1
(Time to run)	*9 sec*	*5 sec*	*8 sec*
Total Time	28 min, 28 sec	49 min, 12 sec	*5 min, 2 sec*

Analysis

The application of the methodology is successful; however, the desired outcome is not achieved. Though the process did provide several important insights.

Application success is based on the overall effectiveness of each portion of the process. The NSN search discovers almost a third of the contracts are part contracts, enough to create a meaningful set. To determine specific details of those contracts a combination of NER and classification are used. SpaCy NER models are able to locate many various entities within each contract and classification models determine which of those entities are relevant. About a third of the desired details are found. For other details, text mining techniques are applied. This approach finds nearly all the desired details. In addition to contract details, part details are also needed. Nearly all desired details are successfully obtained using web scraping methods. From these efforts a Gantt chart is successfully created.

While each individual portion is successful, the project as a whole is not. Due to the lack of overlapping success, only 10% of the contracts have every detail of interest successfully extracted. The lack of overlapping success also means only 21% of the contracts have both an effective date and end date identified. With such few contracts viable for the Gantt chart, only 71% of the parts are represented at all.

Using pattern recognition results in a fast, effective method to identify part contracts. It takes less than 0.03 seconds to determine if a contract contains an NSN. While some contracts are likely missed, most are probably found.

Each contract detail acquired by an NER model combined with a classification model takes about 1.37 seconds per document to acquire. Since the NER models only

take 0.42 seconds to identify entities for a document, most of the run-time is due to applying the classification model. Also, the NER models generate a large number of entities, creating a sparse data space for the classification models to perform in. While the tuned classification models are highly effective on the training data, the many irrelevant entities in the true data makes locating relevant entities difficult, resulting in poor performance.

Alternatively, each detail acquired by text mining only takes 0.02 seconds per contract. In addition, text mining is also more successful at acquiring each detail. Due to the higher success rate and quicker execution time, for this dataset, text mining techniques are likely the best choice. However, this method is highly unlikely to apply well to a different dataset.

Retrieving part information from www.iso-group.com is fast, taking only 0.89 seconds per part. However, since there are 1.63 parts per contract on average, it takes about 1.5 seconds per contract to acquire part information this way.

The Gantt chart executes quickly and provides useful information in an easy-to-understand format. Even when only one part is plotted, as in the case of the third data set, there is still value. Figure 21 shows how the single part is covered by two contracts and each contract's covered date range.

Figure 21. Notional Gantt Chart Product

Unfortunately, the overall execution time is long. The entire process averages 5.7 seconds per contract. If this method were applied unaltered to all 4 million contracts, it would take approximately 263 days to run. However, the process was developed to execute sequentially on a single core. Taking advantage of the other 23 cores would reduce runtime to 11 days.

In addition to the contract evaluation time, there is the process development time. The manual classification of entities takes about an hour, applying LazyPredict and tuning the results takes an additional 5-10 minutes. Thankfully, this process is only completed once per item of interest. That said, if this process was applied to each dataset in an effort to improve accuracy, it would add considerable time.

The NER models run very quickly, averaging only 0.42 seconds per document. A custom NER could obtain many of the contract details more quickly, bringing down the overall execution time to only 3.8 seconds. This would reduce execution time by a third and likely be more accurate.

Summary

The individual steps of the process are successfully executed, and results compiled. However, while each standalone portion performs well enough, there is very little overlap in successes. In addition, some steps take considerable time to execute. As a result of these findings, this method presented is not practical. An alternative method such as a custom built NER is likely more appropriate.

V. Conclusions and Recommendations

Chapter Overview

The partial success and failure of the approach presented in this paper leads to several conclusions. These conclusions are discussed in detail within this chapter.

Conclusions of Research

Supercomputing and machine learning provide access to information at unparalleled levels. Evaluating hundreds of documents for specific pieces of information would take a person weeks to accomplish. Automating the task allows it to be accomplished in half an hour. Furthermore, once a strategy is developed, it can be applied to millions of documents in less than a year, a feat unimaginable for its human counterpart. Though the process is not without its complications.

Machine learning in the form of NER identifies information within a text based on its use, allowing things such as meaningful dates and organizations to be found quickly and accurately. However, existing NER models return all dates and all organizations. If these models are to be useful, additional refinement of the results is needed. Classification models can be used to provide this refinement. This combination of NER models with classification models is also very adaptable to new texts. This is important since all contracts do not follow the same format.

Unfortunately, the performance of this technique was not sufficient to consider it a viable option. The process took a considerable amount time to run and the results were less than satisfactory.

Where NER failed, text mining techniques were used. They performed very well. However, text mining typically only works in similar applications and fails if formats change. As such, despite its success, it is not a recommended approach either.

While the NER models used were not successful, the speed at which they can be applied and return significant information was noteworthy. If custom NER models were built that only returned the desired entities, the classification step, which contributed largely to the failure of the method, could be eliminated. These custom NER models would require a significant investment of time or money to develop.

One part of the process that was successful was extracting part information. With the methods outlined in this paper, a useful repository of parts could be built, identifying when parts are available under contract, who supplies them, and who owns the contract. This information could be used in many beneficial ways, but possibly most important, it could be used to determine if a new contract is needed to maintain or establish access to parts.

Recommendations

To improve upon the approach taken within this paper, a custom NER should be built. This could be used in place of the existing NER models combined with classification models to produce more accurate result, faster. To develop a custom NER, a labeled corpus must be produced. While there are tools to assist this process, it is still manual and requires thousands of contracts be labeled. Once produced, a company such as Google or Skyl.AI could assist in building and applying the custom NER. With the expertise gained from this paper, an individual would also likely be able to create a reasonable model and apply it.

If there was not enough time to produce an entire corpus, the approach taken in this paper could likely be improved in less time. That time would be spent increasing the number of labeled entities on which the classification models are built. Increasing the number of contracts labeled should improve the applicability of the prebuilt NER and improve the overall success rate.

A unique classification algorithm was used for each classification model developed. However, the initial results for each model indicated extra trees might be a good approach. It might be possible to develop a single classification model using the extra trees algorithm. Using a single model could have several advantages, the most immediate benefit being triple the number of training samples.

Furthermore, while not the most difficult part of the process, the collection of part info does take time. Since this information does not change regularly, if a list of parts was obtained, a repository could be built. The benefit of this is a lengthy internet search would not be required each time a part is identified in a contract. The repository could be searched, and the part information quickly acquired.

An additional advantage to using an HPC is the ability to increase computing power. The process utilized for this paper utilized a single core with a standard priority, however the HPC has 592 cores (Talon User Guide, 2020). To use multiple cores, parallel processing would need to be implemented. If all 384 cores in the project node were utilized, the project could run in under 17 hours.

Besides parallel processing, multi-threading would allow multiple, simultaneous internet searches for part information. Implementing this could greatly reduce the time taken to gather part information. Even without consuming all the cores, employing both

parallel processing and multi-threading could result in a process that could be run on all the contracts over a weekend.

Significance of Research

The team, DART, working to provide significance from the effort to collect AFSC contracts received pertinent guidance with regards to the important goal of determining contract expirations. This goal can be achieved to a certain degree using the process in this paper. However, to meaningfully determine expirations for all contracts, time is needed to develop a corpus or improve the classification data sets. No matter the approach, the desired goal of determining part availability is attainable. More importantly, from this determination, access to parts can be assured.

Summary

Contract information can be successfully extracted using machine learning and text mining techniques. However, to provide meaningful outputs a well-developed corpus is needed. In lieu of this corpus, a combination of several methods can be applied to generate a moderately successful model. Once contract information is extracted, it can be used to develop useful tools such as a listing of part availability.

Appendix A

Package Name	Version		Package Name	Version
aiohttp	3.6.3		nltk	3.5
argon2-cffi	20.1.0		notebook	6.1.4
async-generator	1.1		numpy	1.19.1
async-timeout	3.0.1		packaging	20.4
attrs	20.2.0		pandas	1.1.3
backcall	0.2.0		pandocfilters	1.4.3
beautifulsoup4	4.9.3		parso	0.7.0
bleach	3.2.1		pexpect	4.8.0
blis	0.4.1		pickleshare	0.7.5
boto3	1.9.66		pillow	8.1.0
boto	2.49.0		pip	20.2.3
botocore	1.12.67		plac	0.9.6
cachetools	4.1.1		plotly	4.12.0
catalogue	1.0.0		pluggy	0.13.1
certifi	2020.6.20		preshed	3.0.2
cffi	1.14.3		prometheus-client	0.8.0
chardet	3.0.4		prompt-toolkit	3.0.8
click	7.1.2		protobuf	3.13.0
cryptography	3.1.1		psutil	5.8.0
cycler	0.10.0		ptyprocess	0.6.0
cymem	2.0.3		py	1.9.0
decorator	4.4.2		pyasn1-modules	0.2.8
defusedxml	0.6.0		pyasn1	0.4.8
docutils	0.16		pycparser	2.2
en-core-web-lg	2.3.1		pygments	2.7.1
en-core-web-sm	2.3.1		pyopenssl	19.1.0
entrypoints	0.3		pyparsing	2.4.7
gensim	3.8.0		pypdf2	1.26.0
google-api-core	1.22.2		pyrsistent	0.17.3
google-auth	1.22.1		pysocks	1.7.1
google-cloud-core	1.4.3		pytest	6.1.1
google-cloud-storage	1.31.0		python-dateutil	2.8.1
google-crc32c	1.0.0		pytz	2020.1
google-resumable-media	1.1.0		pyzmq	19.0.2
googleapis-common-protos	1.52.0		regex	2020.10.11
idna	2.1		requests	2.24.0
imbalanced-learn	0.7.0		retrying	1.3.3
importlib-metadata	2.0.0		rsa	4.6

iniconfig	1.1.1		s3transfer	0.1.13
ipykernel	5.3.4		scikit-learn	0.23.2
ipython-genutils	0.2.0		scipy	1.5.2
ipython	7.18.1		selenium	3.141.0
jedi	0.17.2		send2trash	1.5.0
jinja2	2.11.2		setuptools	50.3.0.post20201006
jmespath	0.10.0		six	1.15.0
joblib	0.17.0		smart-open	3.0.0
jsonschema	3.0.2		soupsieve	2.0.1
jupyter-client	6.1.7		spacy	2.3.2
jupyter-core	4.6.3		srsly	1.0.2
jupyterlab-pygments	0.1.2		terminado	0.9.1
kiwisolver	1.3.1		testpath	0.4.4
lazypredict	0.2.7		thinc	7.4.1
lightgbm	3.0.0		threadpoolctl	2.1.0
lxml	4.5.2		toml	0.10.1
markupsafe	1.1.1		tornado	6.0.4
matplotlib	3.3.3		tqdm	4.50.2
mistune	0.8.4		traitlets	5.0.4
mkl-fft	1.2.0		urllib3	1.24.3
mkl-random	1.1.1		wasabi	0.8.0
mkl-service	2.3.0		wcwidth	0.2.5
multidict	4.7.6		webencodings	0.5.1
murmurhash	1.0.2		wheel	0.35.1
nbclient	0.5.1		xgboost	1.2.1
nbconvert	6.0.7		xlrd	1.2.0
nbformat	5.0.8		yarl	1.6.2
nest-asyncio	1.4.2		zipp	3.3.1

Bibliography

Agrawal, S. (2019, february 10). *what the heck is word embedding.* Retrieved from towards data science: https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81

Ali, Z. (2019, November 20). *Simple Tutorial on Word Embedding and Word2Vec*. Retrieved from Medium: https://medium.com/@zafaralibagh6/simple-tutorial-on-word-embedding-and-word2vec-43d477624b6d#id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjAzYjJkMjJjMmZlY2Y4NzNlZDE5ZTViOGNmNzA0YWZiN2UyZWQ0YmUiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iL

Ariza-Lopez, F. J., Rodriguez-Avi, J., & Alba-Fernandez, M. V. (2018). Complete Control of an Observed Confusion Matrix. *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, (pp. 1222-1225).

Avola, D., Cinque, L., Foresti, G. L., Lamacchia, F., Marini, M. R., Perini, L., . . . Telesca, G. (2019). A Shape Comparison Reinforcement Method Based on Feature Extractors and F1-Score. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, (pp. 2155-2159).

Badji, I. (2018). Legal entity extraction with NERSystems.

Bessmertny, I. A., Platonov, A. V., Poleschuk, E. A., & Pengyu, M. (2016). Syntactic text analysis without a dictionary. *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, (pp. 1-3).

Bisgin, H., Kilinc, O., Ugur, A., Xu, X., & Tuzcu, V. (2011, 1). Diagnosis of long QT syndrome via support vector machines classification. *J. Biomedical Science and Engineering, 444036*, 264-271. doi:10.4236/jbise.2011.44036

Bulbul, H. I., & Unsal, Ö. (2011). Comparison of Classification Techniques used in Machine Learning as Applied on Vocational Guidance Data. *2011 10th International Conference on Machine Learning and Applications and Workshops*, *2*, pp. 298-301.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems and their Applications, 14*, 20-26. doi:10.1109/5254.747902

Detective, D. (2020, Jan 31). *Why We Use an 80-20 Split*. Retrieved from Toward Data Science: https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d

Download Putty. (2020). Retrieved from Putty: https://www.putty.org/

Farhadloo, M., & Rolland, E. (2013). Multi-Class Sentiment Analysis with Clustering and Score Representation. *2013 IEEE 13th International Conference on Data Mining Workshops*, (pp. 904-912).

Gandhi, R. (2018, June 7). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Retrieved from Towards Data Science: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Gantt Chart. (2021). Retrieved from ProductPlan: https://www.productplan.com/glossary/gantt-chart/

Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* Boston: O'Reilly.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Hale, J. (2019, December 20). *Don't Sweat the Solver Stuff*. Retrieved from Towards Data Science: https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451

Halevy, A., Norvig, P., & Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems, 24*, 8-12.

Han, P., Shen, S., Wang, D., & Liu, Y. (2012). The influence of word normalization in English document clustering. *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2*, pp. 116-120.

Honnibal, M. (2020). *Training spaCy's Statistical Models*. Retrieved from spaCy: https://spacy.io/usage/training

iLauncher v1.10 Downloads. (2020, August 10). Retrieved from HPC Training: https://training.hpc.mil/mod/page/view.php?id=3651

Isahara, H. (2007). Resource-based Natural Language Processing. *2007 International Conference on Natural Language Processing and Knowledge Engineering*, (pp. 11-12).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2015). *An Introduction to Statistical Learning.* New York: Springer.

Jiang, X., & Tan, A.-H. (2010, January). CRCTOL: A semantic-based domain ontology learning system. *JASIST, 61*(1), 150-168.

Kim, Y., Lee, J., Lee, E.-B., & Lee, J.-H. (2020). Application of Natural Language Processing (NLP) and Text-Mining of Big-Data to Engineering-Procurement-Construction (EPC) Bid and Contract Documents. *Conference on Data Science and Machine Learning Applications.* Riyadh. Retrieved from http://eds.b.ebscohost.com.afit.idm.oclc.org/eds/detail/detail?vid=5&sid=08311b79-350a-482d-a0f1-70b60a8e92d2%40pdc-v-sessmgr04&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=edseee.9044209&db=edseee

Kutuzov, A., & Kuzmenko, E. (2019). To lemmatize or not to lemmatize: how word normalisation affects ELMo performance in word sense disambiguation. *To lemmatize or not to lemmatize: how word normalisation affects ELMo performance in word sense disambiguation*.

Leitner, E., Rehm, G., & Moreno-Schneider, J. (2019). Fine-Grained Named Entity Recognition in Legal Documents. In M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, & Y. Sure-Vetter (Ed.), *Semantic Systems. The Power of AI and Knowledge Graphs* (pp. 272–287). Cham: Springer International Publishing.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research, 18*, 1-5. Retrieved from http://jmlr.org/papers/v18/16-365

Li, H., Ma, B., & Lee, K. A. (2013). Spoken Language Recognition: From Fundamentals to Practice. *Proceedings of the IEEE, 101*, 1136-1159.

Luthfi, A., Distiawan, B., & Manurung, R. (2014). Building an Indonesian named entity recognizer using Wikipedia and DBPedia. *2014 International Conference on Asian Language Processing (IALP)*, (pp. 19-22).

Marinov, M., & Efremov, A. (2019). Representing Character Sequences as Sets : A simple and intuitive string encoding algorithm for NLP data cleaning. *2019 IEEE International Conference on Advanced Scientific Computing (ICASC)*, (pp. 1-6).

Menezes, D., Milidiú, R., & Savarese, P. (2019). Building a Massive Corpus for Named Entity Recognition Using Free Open Data Sources. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, (pp. 6-11).

Muthukadan, B. (2020). Retrieved from Selenium with Python: https://selenium-python.readthedocs.io/

Named Entity Recognition. (2020). Retrieved from Skyl.AI: https://skyl.ai/solutions/named-entity-recognition

Natural Language. (2020). Retrieved from Google Cloud: https://cloud.google.com/natural-language

Pandala, S. R. (2020). *Lazypredict*. Retrieved from PyPI: https://pypi.org/project/lazypredict/

Park, Y., & Kang, S. (2019). Natural Language Generation Using Dependency Tree Decoding for Spoken Dialog Systems. *IEEE Access, 7*, 7250-7258.

Partalidou, E., Spyromitros-Xioufis, E., Doropoulos, S., Vologiannidis, S., & Diamantaras, K. I. (2019). Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy. *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, (pp. 337-341).

Patel, B., & Shah, D. (2013). Significance of stop word elimination in meta search engine. *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, (pp. 52-55).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Deep contextualized word representations*.

Phaseit Inc. (2016, May 2016). *PyPDF2*. Retrieved from PyPI: https://pypi.org/project/PyPDF2/

Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, (pp. 35-39).

Richardson, L. (2020). *Beautiful Soup 4*. Retrieved from PyPi: https://pypi.org/project/beautifulsoup4/

Salim, S. S., Ghanshyam, A. N., Ashok, D. M., Mazahir, D. B., & Thakare, B. S. (2020). Deep LSTM-RNN with Word Embedding for Sarcasm Detection on Twitter. *2020 International Conference for Emerging Technology (INCET)*, (pp. 1-4).

Shokripour, R., Anvik, J., Kasirun, Z. M., & Zamani, S. (2013). Why so complicated? Simple term filtering and weighting for location-based bug report assignment recommendation. *2013 10th Working Conference on Mining Software Repositories (MSR)*, (pp. 2-11).

Siencnik, S. K. (2015). Adapting word2vec to Named Entity Recognition. *NODALIDA*.

Silipo, R. (2019, October 1). *From a Single Decision Tree to a Random Forest*. Retrieved from Towards Data Science: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

Sotomayor, M., & Veloz, F. (2017). Thesaurus-based named entity recognition system for detecting spatio-temporal crime events in Spanish language from Twitter. *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, (pp. 1-5).

SVM Kernel Functions. (2021). Retrieved from Data Flair: https://data-flair.training/blogs/svm-kernel-functions/

Talon User Guide. (2020). Retrieved from AFRL HPC: https://afrl.hpc.mil/docs/talonUserGuide.html#sysConfig

Terry-Jack, M. (2019, May 3). *NLP: Pretrained Named Entity*. Retrieved from Medium: https://medium.com/@b.terryjack/nlp-pretrained-named-entity-recognition-7caa5cd28d7b

Wang, Y., Li, L., Wan, X., & Wang, J. (2019). Woven Fabric Defect Detection Based on the Cascade Classifier. *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, (pp. 1-5).

Williams III, C. O. (2020). *Meta Learning Recommendation System for Classification.* Dayton: AFIT.

Wu, Y., Xu, J., Jiang, M., Zhang, Y., & Xu, H. (2015, Nov 5). A Study of Neural Word Embeddings for Named Entity Recognition in Clinical Text. *AMIA Annual*

Symposium proceedings. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765694/

Yiu, T. (2019, May 5). *Understanding Logistic Regression*. Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-logistic-regression-using-a-simple-example-163de52ea900

Zhang, D., Wang, J., Zhao, X., & Wang, X. (2015). A Bayesian Hierarchical Model for Comparing Average F1 Scores. *2015 IEEE International Conference on Data Mining*, (pp. 589-598).

Zhang, K., Sun, L., & Ji, F. (2019). A TOI based CNN with Location Regression for Insurance Contract Analysis. *2019 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1-8).

Zhang, S., Lin, S., Gao, J. F., & Chen, J. (2019). Recognizing Small-Sample Biomedical Named Entity Based on Contextual Domain Relevance. *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, (pp. 1509-1516).

Zoltan, C. (2018, November 13). *SVM and Kernel SVM*. Retrieved from Towards Data Science: https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
25-03-2021	Master's Thesis	SEP 2019 – March 2021

TITLE AND SUBTITLE

Contract Information Extraction Using Machine Learning

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Butcher, Zachary E., Captain, U.S. Air Force

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-21-M-145

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Life Cycle Management Center
1865 4th St
Wright-Patterson Air Force Base
OH 45433
POC: Mr. Philip Ball (Philip.ball@us.af.mil)

10. SPONSOR/MONITOR'S ACRONYM(S)

AFMC AFLCMC/LZIA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The Air Force Sustainment Center assisted by the Data Analytics Resource Team and the Defense Logistics Agency collected four million contracts onto one of the Air Force Research Laboratory's high power computers. This thesis focuses on the effort to determine if parts are available through those contracts. Some information is extracted using machine learning in combination with natural language processing. Where machine learning methods are unsuccessful or inappropriate, text mining techniques, such as pattern recognition and rules, are used. Upon completion, the information is combined into a Gantt chart for quick evaluation. Only 21% of the contracts have their information correctly extracted with this process. To provide an accurate depiction of availability for every part, an improvement is needed. The likely most effective improvement is to develop a custom NER model.

15. SUBJECT TERMS

Machine Learning, Named Entity Recognition, Natural Language Processing, High Power Computing, Word Embedding

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Jeffery Weir, AFIT/ENS
U	**U**	**U**	**UU**	70	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 Ext 4523 (DSN 785) (Jeffery.Weir@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18