Comprehending the Chaos

Leveraging Text to Improve Analysis

Omar Akbik

Cadence Doyle

Technomics, Inc.

February 2021

Data has become the world's most valuable industrial commodity. The defense industry is no stranger to this reality. The quantitative application of unconventional data sources, such as text, can fill gaps that exist in traditional analysis and reduce reliance on opinion based inputs that often require significant uncertainty adjustments. This paper will explore natural language processing (NLP) for classification and clustering in cost analysis and how the authors have applied it in practice.

Table of Contents

Table of Equations

Table of Figures

List of Tables

Introduction

"We only have a few data points." Whether the cost analyst is describing the physical number of points that they have available to build a regression model, or they are simply unable to figure out what data they have, this is one of the most common phrases used in weapon system cost analysis. It is also a phrase that is rooted in the analyst's ability to access and leverage unconventional data sources such as text. Even when data is collected and stored in efficient flat-files or relational databases, qualitative fields such as "Remarks" or task descriptions often contain the bulk of the usable information in a data set. As the size of the data set grows, the ability to tap into these qualitative fields efficiently becomes more important.

Natural Language Processing (NLP) has become a hot topic in commercial industry, with advances that have proven to be very effective, and profitable, for business operations. The tools and skills needed to access and use this valuable form of information are not completely foreign to some performing cost analysis today. This paper explores the history of NLP, advancements that have been made in NLP and machine learning, and how commercial and academic applications of both can be leveraged within the cost community. To address a common data cleansing and normalization problem, the authors describe a use case for classifying data items based on line-item descriptions. The paper concludes with an applied example of the use of open source documentation, and how the documentation can be used to discover latent information on a set of programs that would have previously required significant human study.

Literature Review

The Beginnings

NLP has gained momentum in the past two decades with the increase in data collected in both structured and unstructured forms, but the process of making text machine-readable began back in the 1940s with Father Roberto Busa. As an Italian Priest and librarian, Busa worked with IBM to craft a concordance of the works of Saint Thomas Aquinas. The creation of *Index Thomisticus: Sancti Thomae Aquinatis operum omnium indices et concordantiae* (*The Index and Concordance of All the Works of Saint Thomas Aquinas)* started as a manual process, recording each word used in the selected text. Father Busa realized there may be a more efficient way to accomplish his goal, and pitched the idea to IBM. Their collaboration began before IBM built

their first computer, so Busa and his team progressed from punch cards, to magnetic scripts, finally to typists on computers [1]. In 2005, they published the finished *Index Thomisticus* through a website, rounding out the first exploration into the digitizing of humanities [2].

Since this effort, there have been many advancements and applications in the world of text analysis. Those who seek to categorize their text based on common characteristics typically use classification or clustering. Both classification and clustering, terms often used interchangeably, aim to sort data elements into groups. However, the two methods have distinct applications. Classification uses pre-established categories to sort the text, whereas clustering establishes the categories during the learning process [3]. In other words, classification uses humans to select categories and clustering uses machines to select categories. Another important concept is feature selection, or the process of selecting categories during the clustering process [4]. Various industries have leveraged classification and clustering for both text and non-text data, including economics, healthcare, aviation and construction. Regardless of industry, analysts and decision makers require the ability to extract meaning from text, and the ability to do so efficiently. The examples below will illustrate how classification and clustering have enhanced understanding or decision-making abilities in a variety of industries.

Data Cleansing Concepts

Much like any analytical process, analysts must ensure they pre-process text in such a way that allows for statistical calculations before performing meaningful NLP or analysis. After ingesting the data, analysts and researchers employ standard cleaning processes which removes non-ASCII characters[1] and formatting. Many also remove stop-words, or common "filler" words that have meaning in the structure of a language, but do not provide contextual value for analysis, such as "a", "the", "this". This practice may vary; one study on spam filtering using NLP found that removing stop-words actually decreased model accuracy, since spam and genuine emails use stop-words differently [3]. Analysts also often remove numeric characters, but there may be benefits to leaving them in depending on the goal of the study.

The cleansing process may vary by the type of data being analyzed. For example, the Aviation Safety Report System (ASRS) began collecting incident reports of aviation events back

[1] American Standard Code for Information Interchange (ASCII) charters consist of the English alphabet, numbers, punctuation, and common symbols. Non-ASCII characters include ©, ™, €, →, Ŷ, etc.

in the 1970s. Reporters recorded these incidents not in Standard English, but rather a unique form of short-hand, employing the use of industry-specific abbreviations. In order to classify these incidents, subject matter experts needed to first decode acronyms and expand abbreviations [5]. The need to define acronyms to produce meaningful insights is not unique at ASRS, as cross industry analysis can often encounter specific acronyms and phrases that do not hold any meaning outside of the industry. This definitional problem is also an important consideration with regards to foreign languages. For example, when attempting to classify diagnoses using Georgian medical records, Manana Khachidze and her team struggled with the agglutinative[2] nature of the Georgian language. In fact, they had to use a new stemming algorithm created for the Georgian language [6].

In addition to content definition and stop-word removal, NLP employs a number of unique cleansing processes that aid in the quantitative understanding of the data. Tokenizing, lemmatizing, stemming, and annotating text are all important functions that transform the text into standard form that aids in the creation of machine-readable tables.

Tokenization separates text into character strings delimited by a white space, thereby breaking the text into words called tokens. One can turn these tokens into n-grams[3] for analysis by looking at adjacent tokens. For example, the word "earned" on a resume may not tell you much about a person's job, but the phrase "earned value management" (a *tri*-gram) may offer much more insight. Depending on the goal of the study, n-grams may be more accurate for prediction than words alone [5].

Lemmatization and stemming have the same goal: conflate related word forms to a common word [7]. Lemmatization leverages dictionaries in order to consider the morphological[4] analysis of the word and produce a meaningful stem for each word. Stemming shortens words to their roots by removing suffixes [8]. Stemming is used to save computation resources, or when lemmatization algorithms do not exist for a particular lexicon. The aforementioned study on Georgian medical records used a stemming algorithm since a lemmatization algorithm was not available [6]. NLP is still in its infancy in Greece, and one study had to translate all documents to English, lemmatize the documents, and translate the lemmatized words, or lemmas, back into

[2] The language forms complex words by stringing together morphemes, meaningful word elements, without changing them in spelling or phonetics
[3] A group of 1-n words
[4] Morphology is the study of word formation and the relationship between words in the same language

Modern Greek [9]. Where available, most practitioners prefer lemmatization as it is akin to synonym replacement, providing a more relevant dissection of each word [10].

In order to tag individual words with their corresponding part of speech (POS), analysts may use annotation algorithms. Different dictionaries within the same language may employ different techniques for POS tagging, though most use the universal POS tags, such as PRON for pronouns and DET for determiner. As stated previously, the difference between classification and clustering is whether the target groups are known or unknown, and feature selection is the process of choosing target groups through the clustering process. Annotation becomes important for clustering, as often nouns become the features [10]. Table 1 below demonstrates a sentence (or string), the related tokenized string broken into bigrams, the string with stop-words removed, the stemmed string (note the trailing "es" removed from the verb "ensnares"), the lemmatized string, and the annotated string.

Sentence Type	Result
Original	I am no bird; no net ensnares me
Bigrams	"I am", "am no", "no bird", "bird no", "no net", "net ensnares", "ensnares me"
Stop-words Removed	I bird net ensnares me
Stemmed	"I am no bird no net ensnar me"
Lemmatized	"I be no bird no net ensnare I"
Annotated	"I" (PRON), "am" (AUX), "no" (DET), "bird" (NOUN), "no" (DET), "net" (NOUN), "ensnares" (VERB), "me"(PRON)

Table 1: Sentence Variations by Methodology

Mathematical Constructs

Once the cleaning process is complete, the analyst will have their corpus[5] in an N-dimensional Vector Space Model (VSM), where N is the number of distinct terms used in each document and each axis represents one lemma or stem [7]. Most manipulate this space into a mathematical matrix using a weighting system to reflect a term's relative significance in the feature space. Analysts can then use the matrix for prediction purposes. In their paper, Yili Wang and Heeyong Youn provide a comprehensive review of the different weighting systems,

[5] Latin for "body", refers to the collection of text

including Category Discriminative Strength, Part of Speech Feature-Based weighting, and Intra-Distribution Measure [4]. These weighting systems are relatively new, so today a more simplistic algorithm called term frequency-inverse document frequency (TF-IDF) is more popular due to its effectiveness and the low computational complexity [7].

To understand the TF-IDF, one must understand the document-term matrix (DTM), also known as the term-document matrix when the axes are reversed. The DTM is a mathematical matrix wherein each row is a new document, each column represents one n-gram denoted by lemmas or stems, and the value represents how many times that term appears in the document [11]. Note that the usage of the word "document" does not necessarily mean a Microsoft Word file. This "document" can be a single post from a message board [10], or individual reports consisting of less than 500 words, such as a Tweet [5]. Table 2 below provides an example of two strings representing two separate documents, Table 3 illustrates the resulting term-document matrix.

Document	Content
String 1	I will attend the virtual symposium
String 2	I plan to go to the online symposium

Table 2: Example of Documents

	I	will	attend	the	virtual	symposium	plan	to	go	online
Document 1	1	1	1	1	1	1	0	0	0	0
Document 2	1	0	0	1	0	1	1	2	1	1

Table 3: Term-Document Matrix of Table 2

Armed with a term-document matrix, an analyst can begin the weighting process with the following TF-IDF formula:

$$w_{i,j} = tf_{i,j} \times \log\frac{N}{df_i}$$

Equation 1: TF-IDF

Here, $w_{i,j}$ is the weight of the i^{th} term in the j^{th} document. $tf_{i,j}$ represents the term frequency (tf) of the i^{th} term in the j^{th} document. N is the number of documents in the DTM, or the number of rows in the DTM. Finally, df_i is the number of documents containing the i^{th} term.

Classification problems and clustering problems leverage these term-weightings in different ways. Classification will use term-weightings as predictors for response variables, as target groups are known. With unknown target groups, clustering uses term-weightings for feature selection, and then categorizes the document by the selected features.

Clustering and classification processes can benefit from leveraging machine learning (ML) algorithms, many of which analysts have applied to NLP and categorization processes in general. Because of the breadth of available algorithms, most researchers test multiple algorithms and select the best methodology for their desired outcome based on hypothesis tests and goodness-of-fit statistics, much like traditional statistical analysis. However, this does not always mean choosing the technique that delivers the best F1 score[6]. For example, medical researchers using tumor size to predict the presence of a rare cancer may select an algorithm with the lowest recall[7], whereas social media engineers care much less about promoting content that may not match a user's interest and instead prioritize precision[8]. There is not a "one-size-fits-all" algorithm, so descriptions of each methodology follows.

Machine Learning in NLP

For classification problems, there are a variety of supervised learning[9] algorithms one can employ, such as Naïve Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Trees. For clustering problems, studies often incorporate unsupervised learning[10] algorithms such as Principal Component Analysis (PCA), leveraging variance inherent in the data, and K-Means Clustering, based on Pythagorean[11] distances between data points. There are methods to enhance each of these models, such as Gradient Boosting, Bootstrapping, or Cross-Validation [12]. A brief overview of these processes is outlined below.

---

[6] The F1 Score is a calculation based off Recall and Precision, defined in the footnotes below, given by the following formula: $F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$

[7] Recall is a calculation defining how many positives the model assigned out of the total number of positives present in the test set, given by the following formula: $recall = \frac{true\ positives}{true\ positives + false\ negatives}$

[8] Precision is a calculation defining how many positives were correctly determined out all the positives labeled as such in the test set, given by the following formula: $precisioin\ \frac{true\ positives}{true\ positives + false\ positives}$

[9] Supervised ML tasks predict outputs based on labeled data. Classification is a supervised ML algorithm because it requires known target groups, or labels

[10] Unsupervised ML tasks predict outputs based on unlabeled data. Clustering is a supervised ML algorithm because it has unknown target groups, or labels

[11] The Pythagorean Theorem uses the following equation to find the distance (d) between two points (x_1, y_1), (x_2, y_2): $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Naïve-Bayes Classifier (NB)

The NB classifier uses Bayesian probability theory and assumptions of independence in an attempt to identify the "most likely" category in which a document or data element belongs [3]. If attempting to classify produce for example, consider the classifier assigning a particular piece of fruit as an orange since it is round, has a peel, and is 15 cm in diameter. NB classification considers these characteristics as independent, and each characteristic contributes to the probability of that piece of fruit being classified as an orange independently. It ignores the possible collinearity between variables frequently present in the real world. For this reason, NB is not always the most accurate learning algorithm; however, because it is not computationally demanding nor difficult to manipulate, NB is common in classification [13].

Analysts have successfully applied NB in NLP classification efforts dating back to 1961, though more recent studies use NB in spam filtering and anonymization, which is the process of obscuring sensitive information in a document [3]. When discussing the sentiment[12] behind bodies of text, Sun's team found that NB underperforms SVM but is still the go-to method for most sentiment classification studies [10]. NB and KNN have considerable overlap due to their use of Bayesian statistics [12].

K-Nearest Neighbors

KNN is an instance-based learning algorithm, which is considered a lazy learning[13] algorithm. When an analyst assigns pre-determined classes to the training set of data, KNN determines the class of the test data by observing the class of its nearest neighbors, as defined by the Pythagorean distance between the two points [13]. This is where weighted document term matrices apply, such that documents with the most terms in common are classified as the same group. In statistical terms, given a positive integer K and a test document, the KNN classifier first identifies the K points in the training data that have the shortest triangular distance to the data point in question. In other words, KNN classifies a test observation to the class with highest estimated probability using Bayes' rule[14] [12]. KNN has seen success as a supervised learning

[12] Sentiment analysis attempts to uncover feelings behind words and phrases, i.e., whether an author had positive or negative attitude towards a particular topic

[13] Lazy learning is a machine learning method where the algorithm begins classification when it receives the data, and has no "training" step before the classification process begins

[14] A theorem determining probability of an event (A) based on prior knowledge that a different event (B) has already occurred, using the following formula: $P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$

algorithm, and was of particular use to the Ministry of Labor, Health and Social Affairs of Georgia. To create a centralized repository of health documents for the Georgian population, the government needed a way to classify years of patients' health documents, created as free text files, to store them neatly in a database. As mentioned in the Data Cleansing section, Khachidze's team used NLP to classify diagnoses in medical documents. The only features that gave less than adequate results were documents about the liver and biliary systems, which were often misidentified as the other due to their common characteristics. Otherwise, the team determined their classification effort was a success. The team tested both SVM and KNN, where SVM had slightly better precision and recall than its opponent [6].

Support Vector Machines

SVM is somewhat unique in that it can be applied to either supervised or unsupervised learning problems. Vladimir Vapnik developed SVM in 1996 as a classification algorithm before Asa Ben-Hur and his team modified it for clustering purposes [14]. For supervised learning situations, SVM combines support vector classifiers and non-linear kernels. Support vector classifiers are hyperplanes seeking to divide the data in two groups, classifying each test case according to the side of the hyperplane where the data point lands. SVM takes support vector classifiers into multi-dimensional space with more than two groups using non-linear kernels. Figure 1 demonstrates a support vector classifier with a polynomial kernel. Represented by thick, black lines, the support vector classifier divides the data into two classes, blue and pink. The black dashed lines represent other possible support vector classifiers. This algorithm would tag any test data which falls between the black lines as a red category, and would classify any outside the lines as the blue category. The SVM misidentifies some observations, as indicated by the blue dots within the pink background.

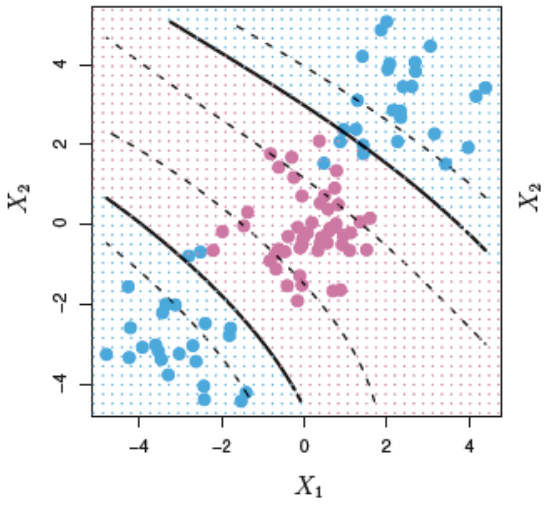

Figure 1: SVM Illustration (James, Witten, Hastie & Tibshirani, 2013)

Analysts use non-linear kernels to enlarge the feature space to accommodate a non-linear boundary between the classes, though Gareth James and his team have much more to say on the topic [12].

A study on aviation incident reports indicates the SVM technique gave "excellent results for document classification tasks". Using a SME-generated list of 37 classifiers, the authors obtained an optimal precision rate of 86% and an F1 score of 79%. They employed a robust resampling method via bootstrapping to enhance their model, though one must attribute a part of their success to the amount of data they have. A combination of the Aviation Safety Report System and the European Coordination Centre for Accident and Incident Reporting Systems databases gave them hundreds of thousands of documents with 15 million words to act as indicators or differentiators for classification [5].

Decision Trees

Arguably the most common ML process leveraged today, decision trees are a family of logically-based, non-probabilistic classification methods. They are directional graphs that start at the base with a single node and then extend to many leaf nodes. These leaf nodes represent the features an analyst attempts to classify [15].

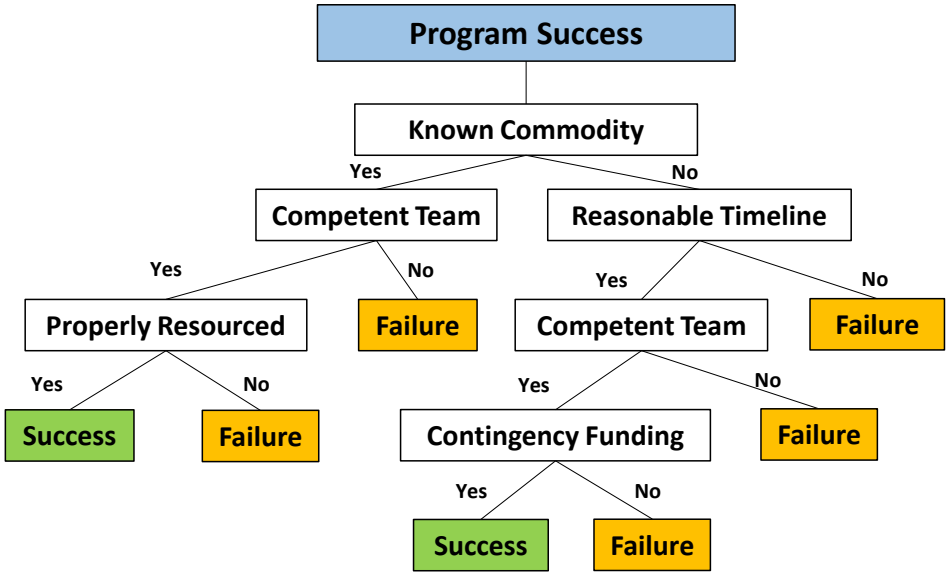

Figure 2: Decision Tree Diagram

Decision trees are easy to understand and explain; some analysts suggest they mimic the decision-making processes of humans. Though they also have an ability to handle qualitative factors without relying on dummy variables, they do not have the predictive performance of other algorithms mentioned previously [12]. A study on construction incident reports highlights this fact well, as the F score for the authors' decision tree model was not as impressive as their SVM or KNN models [15]. However, *An Introduction to Statistical Learning with Applications in R* details a way to enhance the predictive accuracy of decision trees through aggregation, such as bagging, random forests, and boosting [12]. Other studies report boosting decision trees as effective as other methods, e.g., SVM [3]. Boosting involves making copies of the original decision tree sequentially, where each tree grows based on information obtained from previous trees vice creation of independent samples [12]. This method can turn a fairly inaccurate decision tree into a robust classifier.

Gradient Boosting Machines

Friedman coined the name "Gradient Boosting Machine" in 2001, though it was based on the work of Freund and Schapire who introduced weighted, iterative classification in the 1990s [16]. GBM is a boosting algorithm applied to decision trees that uses previous information, gained sequentially from previous decision trees, to create a better model [17]. Each new model starts with information gathered in the previous model, delivering a more accurate prediction of

the class variable. The new model attempts to minimize the loss function[15] associated with the system, and becomes correlated with the negative gradient of the loss function[16]. [9]. Many analysts consider GBM to be state-of-the-art technology due to its success in classifying challenging datasets, though its computation time and resource requirements exceed most other machine learning techniques [16].

An application of GBM in the realm of NLP originates from a demand for sentiment analysis in newspaper articles written in Modern Greek. Despite the lack of tools for NLP in Greek, the authors' ability to translate the text into English and leverage extensive NLP tools contributed to a formidable piece of analysis. The authors' research in this area shows that between decision trees, SVM, NB, Deep Learning, and GBM algorithms, GBM out-performed the other methods in both precision and recall, as seen below in Figure 3 [9].

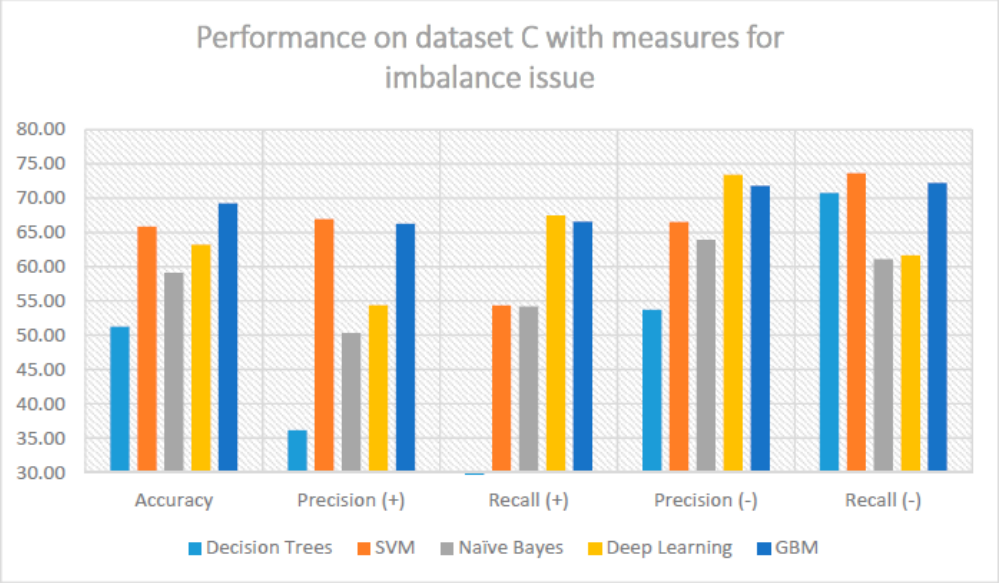

Figure 3: Model Results (Athanasiou & Maragoudakis, 2017)

Principle Component Analysis

All of the aforementioned processes and algorithms have been used extensively in NLP and, more broadly, data analysis. In contrast, PCA has not.

[15] In a model, loss is the penalty for a bad prediction. The loss function aggregates the individual losses in a model, resulting in a number used to compare models.
[16] A gradient is a slope, the negative gradient is the most negative slope. The negative gradient of the loss function is therefore the most negative slope of the loss function, meaning it reduces the loss function in order to create a better model.

PCA is a dimension reduction technique that derives a low-dimensional set of features from a large set of variables [12]. Analysts typically use this method to find orthogonal, or uncorrelated, linear combinations of a large set of correlated variables in order to capture the most amount of variance in the data. Analysts often use PCA to extract the features associated with the matrix eigenvectors[17] based on the descending value of the eigenvalues[18] in the vector space [18].

While not employed extensively in NLP, PCA is a common tool in statistical analysis. One study has successfully used PCA to determine which features indicate progression in Parkinson's disease. The authors took 14 measurements of Parkinson's' patients center of pressure using quantitative posturography[19], which measures the sway of the body while standing. Postural problems worsen as the disease progresses in Parkinson's patients, and doctors prescribe different medications, such as dopamine and levodopa, to combat these issues. Identifying which center of pressure measures indicate disease progression allows doctors to treat proactively rather than reactively. The team used PCA to discover that five out of fourteen collected features provided the most insight into the postural control, including mean velocity and principal sway direction [19].

PCA's ability to leverage eigenvalues inherent in a high-dimensional matrix is well suited to vectorized text document applications (e.g., a DTM) because of the large number of dimensions that correlate one document to the next [12]. Though PCA and NLP have existed for decades, very few studies regarding NLP have used PCA for feature selection. One study leveraged PCA for feature reduction, but used K-Means and Hierarchical clustering to choose the features. This enabled the authors to quickly translate patient summaries, medical prescriptions, and lab results into a machine-readable table with categories for diagnosis type [20].

The Future

This literature review is by no means exhaustive, but offers a fairly comprehensive view of how private industries have leveraged NLP and ML techniques. Academic and professional

[17] Eigenvectors are characteristic vectors of a linear system of equations, such as a matrix equation. They have many uses in engineering and physics, PCA uses eigenvectors to determine the directions of the vector space
[18] Eigenvalues are characteristic roots of a linear system of equations, such as a matrix equation. Eigenvalues correspond with Eigenvectors, and represent the magnitude of the direction of the eigenvector
[19] Posturography is the technique used to quantify postural control in upright stance in either static or dynamic conditions

studies have employed many different algorithms to uncover information from their text which can lead to better decision-making. Cost estimators and analysts have always sought to use the best data for their analyses, whether that be past performance data or build specifications, but have not always been successful in accessing or utilizing that data.

In the following sections this paper will discuss the mechanics behind classifying small text snippets using NLP and ML. It will then bridge the gap in the current literature by employing the standard PCA processes in the context of document clustering for NLP. Specifically, the authors demonstrate how analysts can use PCA, and other standard Eigen-processes, to find clusters within a library of documents in order to identify analogies for use in cost and economic analysis of weapons systems.

Applications of NLP in Defense

NLP has become a common, almost ubiquitous tool in the private sector. As demonstrated in the literature review, companies and researchers leverage NLP in order to accomplish simple business requirements such as text mapping and feature extraction, and for more advanced purposes like automated help desk ticket resolution. The literature surrounding NLP is robust, with applications spanning nearly every industry. However, one major gap in the academic application of NLP relates to project management, particularly applications pertaining to the project management of federal programs. This application space consists of programs that are oftentimes larger and more complex than their commercial counterparts, in part due to the stringent risk management protocols which require compliance with federal procurement law, as well as the need to meet strict military characteristics requirements [21]. Management of these programs typically includes applying standard project management techniques, such as cost and earned value analysis, but has not widely adopted more advanced analytical techniques such as NLP and ML to date.

That is not to say that the federal government, and the Department of Defense (DoD) in particular, has not started the process of exploring the application of NLP for other purposes. The Joint Artificial Intelligence Center (JAIC) has been actively exploring how policy analysis and implementation can use NLP, e.g., to ensure that enacted policy is not redundant with past implementations [22]. ML in general has also garnered the attention of congress, as the National

Defense Authorization Act (NDAA) for Fiscal Year 2019 called for the further development of ML and AI solutions to operational problems within the Department, both within the JAIC as well as the individual components [23]. Further, the 2018 National Defense Strategy specifically called for the application of commercial advancements in ML and AI to advance autonomous systems [24]. Despite these positive developments, NLP has failed to gain traction within the business of defense.

Of course, it is naïve to assume that the wide adoption of NLP would take hold overnight. Project management is an analytical discipline and the application of new techniques, regardless of rigor, should prove to have significant decision making value to stakeholders. Cost analysts in particular follow well-defined principles and processes for establishing technical baselines, identifying data, collecting and preparing the data, and ultimately producing models that provide insights to decision makers. The early stage activities, namely identifying and collecting data, are often the most arduous and time consuming in the analytical lifecycle. Data preparation is also a significant hurdle that analysts need to overcome in order to produce meaningful analysis.

NLP can provide the most value to cost analysts in these early stages of the lifecycle. Clustering algorithms may be employed to identify analogous programs of interest, thereby pointing the analyst towards data sources that may provide significant model inputs and reduce data source identification efforts. Classification algorithms can be leveraged to prepare said data for model development. Data mapping is a common problem that analysts face during the preparation phase, and ensuring that the mapping is of high enough quality to facilitate comparisons among data items is critically important to the success of the project. While not an exhaustive treatment of NLP applications, NLP has shown real potential in helping to solve these problems.

Classification: An Updated Methodology

One of the major deficiencies in program management for defense programs is the ability to tag and map line items within the data used for analysis. Because federal program offices oftentimes operate independently of one another, data collection and record keeping of financial and programmatic metrics is dependent on the systems employed within the program office and the skillsets of those responsible for recording and retaining the data. This can lead to wide disparities in the quality of financial and programmatic data. These disparities are compounded

by differences in tagging structures due to deviations from the Military Standard work-breakdown structure (that only defines WBS elements down to a certain level), and are complicated even further for weapons systems that lie outside of the purview of the DoD, such as those that fall under the umbrella of the National Nuclear Security Administration (NNSA).

Regardless of the state of the raw data, analysts require normalized and standardized data in order to produce meaningful analytical products, such as cost and schedule estimates. One key step in producing this analysis is aggregating "like" line items together, and comparing/contrasting those aggregations in order to gain a complete picture of a program. These activities require the data to be tagged, or categorized, correctly. The tedious tagging task requires an unlucky analyst to read each line item and determine, based on best judgement, how to categorize each line item. Or an analyst may use a simple analytical algorithm (think SEARCH or FIND in MS Excel) that searches for key words in qualitative data fields in order to make a judgement call. The subjectivity of these practices is not only error prone, but a wildly inefficient use of time that an analyst could spend producing meaningful analytical products that would have a positive impact on the management of the program. This inefficiency eats away at the scarce analytical resources upon which federal programs rely for sound data-driven advice.

ML classification algorithms have shown potential in filling this void. These algorithms compare model inputs against a pre-defined set of categories, seeking to use these inputs to decipher which category the data falls into. ML classification models range in their analytical construct, from tree-based bagging algorithms (e.g., Random Forest) to non-linear optimization (e.g., logistic regression). While various forms of ML classification deserve their own academic treatment, one such method, GBM, has proven powerful and efficient at solving classification problems for cost estimating data.

GBM for Cost Data Classification

As outlined in the literature review, GBM is a tree-based supervised ML technique. The most widely known of these tree-based ML techniques, Random Forest, seeks to average the results of a pre-defined ensemble of decision trees in order to turn what would be considered a "weak" learning algorithm, into a "strong" learner through the power of numbers. In contrast to Random Forest's method of treating each tree independently, GBM seeks to improve upon each of the subsequent bagged iterations by incorporating the error present in the prior iteration

through a learning rate. These learning rates are incorporated through the construction of new base-learners that are maximally correlated with the negative gradient of the loss function of the *entire* ensemble [25].

$$F_m(x) = F_{m-1}(x) + \alpha \sum_{j=1}^{J} \gamma_{jm}$$

Equation 2: GBM Estimation Process

The method of performing GBM is demonstrated in Equation 2, where α is the learning rate, and γ is the log-odds value that minimizes the differentiable loss function of the observed error. The ultimate goal of finding the most efficient path down the error-rate curve lends GBM particularly well to classification problems that may have a higher inherent standard deviation than simple averaging, such as Random Forest.

The inputs required to produce an effective GBM model parallel those required to construct a "traditional" statistical model (e.g., linear regression) and, not surprisingly, the data must be "tidy", i.e., cleansed and normalized. This constraint effectively opens the door to leveraging the basic structures of NLP, such as a DTM, to produce classifications. Further, because GBM does not make any assumptions about the distribution of the underlying data and collinearity constraints are unlikely to bias the model, the high-dimensional, sparse nature of the matrix further reduces the application constraints.

Consider a case where a program or oversight office has a large number of data submissions with various fields, but has a very limited capacity to tag the data. The data submissions recur monthly, and most of the relevant identifiable information is stored in text fields. Rather than spending countless hours manually tagging data on a monthly basis, an analyst could employ the basic NLP techniques described in the data cleansing section to standardize, normalize, and cleanse the data fields through stemming or lemmatization to produce a common set of data elements to be used to create a DTM.

This scenario is relatively common, and one in which the application of NLP has demonstrated real value. Assuming that the project team is able to identify discrete categories that data line-items should relate to, qualitative data fields are likely to provide the keys to how

each data element should be mapped. For example, the following subset of data entries (Table 4) represent the beginnings of the input variables to a GBM classification model. In order to leverage the "Raw Text" field for data mapping, text fields are cleansed and normalized and the "Clean Text" field is tokenized. The tokenized cleansed data fields are then transformed into a DTM, and leveraged as input to the GBM model.

Raw Text	Clean Text
big data infrastructure - equipment/software/support - buy #1 big data infrastructure - equipment/software/support - buy #1	big datum infrastructur equip softwar support bui 1 big datum infrastructur equip softwar support bui 1
Wire EDM NN Shell Wire EDM	wire edm nn shell wire edm
5 axis machines-M90 AM (5) #3	5 axi machin m90 be 5 3

Table 4: Raw vs Clean Classification data

In this scenario, the analysts know that the data falls into a three distinct categories, namely 'Metalwork, Machinery and Equipment' (WPU113), 'Special Industry Machinery and Equipment' (WPU116), and 'Machinery and Equipment: Miscellaneous Instruments' (WPU118). By manually mapping these three somewhat similar data elements once, future mapping error can theoretically be reduced by employing a classification algorithm for future submissions. The dataset used for this example contains 1777 rows for training, which were manually mapped into the three categories, and 314 data elements used as a test set to gauge the efficacy of the GBM algorithm for mapping these data elements.

Result	WPU113	WPU116	WPU118
Correct	654	334	650
Incorrect	18	40	81

Figure 4: Training Set GBM Results

Result	WPU113	WPU116	WPU118
Correct	112	53	103
Incorrect	5	14	27

Figure 5: Test Set GBM Results

While not a perfect exercise in data classification, the results of the GBM data mapping for this example proved to have significant benefits for the analysts responsible for analyzing the data in terms of time required to produce the mapping. If employed over time, the learning algorithms would likely experience improved efficiency and accuracy, further reducing the need for analyst intervention and decreasing the likelihood for human error.

Clustering: When Categories are Unknown

Data mapping is an important step in the analysis process. However, the underlying data categories are unfortunately not always known. In these instances analysts require another method for categorizing data in order to make it useful for analysis and model input. Clustering is an unsupervised method of utilizing the data elements themselves to discern how they should be grouped together. As opposed to classification, clustering data elements makes no pre-determined assumptions regarding the groups in which the data might fall into.

Practitioners in industry apply various forms of clustering algorithms, many of which are based on distance metrics (a number of these algorithms have been covered in the literature review- see the sections on SVM and PCA). These metrics aim to measure either the angular distance between vectors created by character strings, or the Euclidean distance[20] between data points in order to find groups of "closest match". A simple example of one such metric, the cosine similarity, is demonstrated in Figure 6.

Document	Content
String 1:	"Develop build 3 DEVELOP Systems Engineering & Program Management firing set"
String 2:	"Program Manage Fire Set Build 3 Development"

	build	develop	engin	fire	manag	program	set	system
String 1	1	2	1	1	1	1	1	1
String 2	1	1	0	1	1	1	1	0

[20] "Euclidean distance" is defined as the length of a line segment connecting two points on a plane, as calculated via the Pythagorean Theorem.

$$\cos(\theta) = \frac{\sum_{i=1}^{n} X * Y}{\sqrt{\sum_{i=1}^{n} X^2} * \sqrt{\sum_{i=1}^{n} Y^2}}$$

$$\cos(\theta) = 0.8616$$

Figure 6: Cosine Similarity Example

As the name implies, cosine similarity is used to measure the cosine of the angle between two documents, with a score falling between -1 and 1.[21] While these metrics can be very useful in determining similarities between both strings and documents, they are very dependent on the weight of each of the individual entries in the vector, limiting their effectiveness on shorter strings. For example, unrelated short strings (less than 10 words) may contain common words that do not provide valuable context to the true meaning of the line item, such as the word "Program." Leveraging weighting techniques, such as TF/IDF, is of limited use where strings contain such limited contextual information. Distance metrics cannot easily compare more than two documents at once without requiring additional assumptions, or beyond orthogonal entries. Data mapping through clustering of short strings is therefore limited without additional information or external data sources.

Fortunately, where program offices and managers lack in standardized data collection, they make up for in requirements documentation and other programmatic material. These documents contain a bevy of information describing the systems (from their technical characteristics to their strategic importance) much of which can produce meaningful similarity metrics as described above. Analysts have leveraged and employed these techniques extensively in literature and in practice [26].

Looking beyond simply mapping line items, clustering has the potential to influence another major aspect of cost analysis often overlooked during the modeling process. In the early stages of a program's acquisition lifecycle, cost analysts rely heavily on information for analogous systems, including cost data to understand potential cost drivers and technical data (e.g., weight, square footage, etc.) for parameters (i.e., independent variables) that have demonstrated a certain level of statistical significance in predicting cost over time. However,

[21] A zero angle is equivalent to a cosine similarity score of 1. Completely divergent vectors, with a 180 degree angle is equivalent to a cosine score of -1.

these data rarely provide insight into exogenous variables that may have impacted the program, such as inter-program dependencies, schedule delays or even program cancellation. Furthermore identification of analogous systems is an inherently qualitative determination, oftentimes dependent on SMEs having the awareness and technical understanding of analogies. Clustering algorithms may aid in systematically and objectively identifying analogies for consideration in regression and other analyses.

Eigen-Processes for Clustering

The clustering processes described have proven effective in comparing the similarity between documents and sub-documents on aggregate, and would enable the analysts to discover "like" programs from within a library of requirements. There are, however, mathematical techniques that are commonly employed in econometric and statistical analysis that are notably absent from current literature. Two related techniques, Principle Component Analysis (PCA) and Factor Analysis demonstrate great potential in the text clustering realm. PCA, a dimension reduction technique typically used in regression analysis to calculate the linear combination of a large set of correlated variables is well suited to the large dimensionality of the DTM, and identifying the correlation between texts. Factor Analysis seeks to find the correlations between DTM vectors and the orthogonal vectors that make up the variance in the data, as captured through a variance-covariance matrix of the DTM. This makes Factor Analysis well suited to identifying how various features inherent in the data form groups (or clusters) based on a pre-defined number of "factors" that act as a proxy for exogenous variables of interest [27]. Mathematically, the Factor Analysis process can be defined as an estimate of the true covariance matrix, as in Equation 3:

$$\hat{\theta} = \hat{L}\widehat{L^T} + \hat{\psi}$$

Equation 3: Factor Analysis Process

Where θ is the estimated covariance matrix, L is the loading factor (or estimated eigenvalue, eigenvector pairs for n-factors), and ψ is the diagonal of the covariance-matrix containing the unique variance in the dataset. All of the variables are estimated via maximum likelihood. These Eigen-processes can find the corresponding correlations, and groupings, between the unobserved content in a list of documents, and the relevant eigenvectors that make

up the dimensional space. This clustering effect can compare the descriptions of various weapon systems of various asset classes. To demonstrate via an applied example, the chosen list of systems used in this analysis are listed in Table 5:

Weapon	Asset Class
B61, B83, B28	Nuclear Bomb
W88, W80, W87, W76	Nuclear Warhead
Peacekeeper Missile	Inter-Continental Ballistic Missile (ICBM)
GAM-87-Skybolt	Air-Launched Ballistic Missile (ALBM)
Standard Missile 6, Rolling Airframe Missile	Surface-to-Air Anti-Air Warfare (AAW)
AGM-114 Hellfire, AGM-179 JAGM	Air-to-Surface Missile (ASM)
Trident	Submarine-Launched Ballistic Missile (SLBM)
Tomahawk	Long-Range Cruise Missile (Tactical Strike)
AMRAAM, Sparrow, Sidewinder	Air-to-Air Missile (AAM)

Table 5: Select Weapon Systems

In order to keep this demonstration in the public domain and consistent with documented success of source data training NLP applications, articles related to each weapon system in Table 5 have been scraped from Wikipedia for use in this example [28].[22] By scraping the text from the corresponding Wiki articles (or any text document or webpage) the analyst can create a repository of information for a set of systems or components of interest, and leverage the repository to compare systems for use in analogous program identification.

Document repositories can easily be translated into a DTM, enabling the analysis of document contents using familiar quantitative tools. However, while simple statistical calculations are possible and may yield some insights (such as generating a correlation matrix), the DTM produced from a document repository will undoubtedly be a large, unwieldy matrix. As stated, PCA is an efficient means to reduce the variable count[23] of the DTM due to high dimensionality while retaining the variability of the data. By identifying the eigenvectors that account for the most amount of variance in the data set, the analyst can easily compare the

[22] Open source documentation was used in order to keep this example in the public realm. Selected Acquisition Reports (SARs) or other DoD specific data sources can be substituted in practice.
[23] Variable count in the DTM are the individual tokens that make up the corpus (or document). PCA will find the optimal linear combination of the variable set that captures the most variance in the data.

content of the document library against a finite set of uncorrelated linear functions in order to isolate correlated groups. The output of the PCA on the DTM constructed from the document library is featured in Table 6.

	PC1	PC2	PC3	PC4	PC5
Standard deviation	9.8168	5.1387	4.22994	3.93319	3.60072
Proportion of Variance	0.4227	0.1158	0.07848	0.06785	0.05686
Cumulative Proportion	0.4227	0.5385	0.61697	0.68482	0.74168

Table 6: Document-Term Matrix (DTM) Principle Component Analysis (PCA) Output

The PCA output indicates that between four and five principle components of the DTM account for over 70% of the variation contained in the matrix via the "Cumulative Proportion." It should be noted that the number of principle components that account for a sufficient amount of the variance in the system is less than the number of pre-defined asset classes available in the library, indicating some correlation between the descriptions of the assets themselves.

The number of principle components identified during PCA will be used as input for the Factor Analysis, as they provide a general idea of how many eigenvectors should be used to capture as much correlation as possible from the covariance matrix. Factor Analysis will then seek to identify correlation between the data elements (in this case the documents in the library) against a set of unobserved factors. The true meaning of the unobserved factors are undefined and are open to subjective interpretation, but the number of, and variance captured in, said factors should mimic the eigenvectors produced by the data. Table 7 displays the output of the Factor Analysis on the DTM.

```
Uniquenesses:
                B61_nuclear_bomb                  B83_nuclear_bomb                  B28_nuclear_bomb
                           0.252                             0.257                             0.446
                             W88               W80_(nuclear_warhead)                               W87
                           0.285                             0.251                             0.242
                             W76                 LGM-118_Peacekeeper                  GAM-87_Skybolt
                           0.373                             0.314                             0.302
RIM-116_Rolling_Airframe_Missile            RIM-174_Standard_ERAM                 AGM-114_Hellfire
                           0.148                             0.284                             0.368
                     AGM-179_JAGM                 Trident_(missile)             Tomahawk_(missile)
                           0.419                             0.078                             0.185
                   AIM-120_AMRAAM                     AIM-7_Sparrow                  AIM-9_Sidewinder
                           0.194                             0.284                             0.147

Loadings:
                               Factor1 Factor2 Factor3 Factor4
B61_nuclear_bomb                         0.259   0.825
B83_nuclear_bomb                         0.253   0.810  -0.153
B28_nuclear_bomb                0.193    0.233   0.643   0.219
W88                                      0.802   0.264
W80_(nuclear_warhead)           0.249    0.739   0.360   0.105
W87                             0.220    0.811   0.188  -0.128
W76                                      0.779   0.116
LGM-118_Peacekeeper             0.767    0.271           0.156
GAM-87_Skybolt                  0.822    0.104
RIM-116_Rolling_Airframe_Missile 0.872   0.104          -0.280
RIM-174_Standard_ERAM           0.838
AGM-114_Hellfire                0.736    0.212           0.202
AGM-179_JAGM                    0.705    0.103          -0.264
Trident_(missile)               0.889    0.152          -0.326
Tomahawk_(missile)              0.891
AIM-120_AMRAAM                  0.801            0.150   0.374
AIM-7_Sparrow                   0.817            0.119   0.183
AIM-9_Sidewinder                0.892    0.100   0.118   0.180

                Factor1 Factor2 Factor3 Factor4
SS loadings       7.604   2.843   2.082   0.643
Proportion Var    0.422   0.158   0.116   0.036
Cumulative Var    0.422   0.580   0.696   0.732

Test of the hypothesis that 4 factors are sufficient.
The chi square statistic is 12.93 on 87 degrees of freedom.
The p-value is 1
```

Table 7: Factor Analysis Output

The results of the Factor Analysis confirm that four factors are sufficient to capture over 70% of the variance present in the data via the SS loadings (sum-of-squared loadings)[24], which is consistent with the results of the PCA. Additional factors would have captured additional variance, and likely identified additional correlations between the documents that were previously unaccounted for. The model also validates that each of the documents in question provides some value to the overall system via the uniqueness scores. The uniqueness score, which measures how much of the variance in the system is "unique" to each of the elements, should ideally be low (close to zero). The fact that not one of the elements demonstrates a uniqueness score above 0.5 validates that each of the project documents provides some level of

[24] The sum-of-square (SS) loadings are exactly as they sound. For each factor, the SS loading is the sum of the squared loading factors under each of the factors.

information to be interpreted by the defined set of factors. Finally, the p-value confirms the null hypothesis that the four factors are sufficient to describe the set of documents in the library.[25]

The loadings themselves represent the clusters. Each element has a loading coefficient associated with at least one of the unobserved factors, representing the correlation that the data element (document) has with the unobserved factor.[26] The results of this particular Factor Analysis can be interpreted at a high level as follows:[27]

- **Factor 1:** Documents with high loadings under factor 1 are generally associated with missiles. Because most of the documents in the library are associated with missile systems of various forms, it is not surprising that the majority of the elements demonstrate a high correlation with this factor.
- **Factor 2:** Documents with high loadings under factor 2 are most generally associated with nuclear warheads. Three of the documents that are not associated with nuclear warheads explicitly were designed to be nuclear delivery vehicles (Peacekeeper ICBM, Skybolt, Trident), and rightfully demonstrate significant loadings under factor 2.
- **Factor 3:** The highest loadings under factor 3 are associated with nuclear bombs. However, there may be characteristics associated with other weapons systems that caused certain non-nuclear weapons systems to correlate with the factor. The three air-to-air missiles with loading factors under three, while low, indicate that some similar component characteristics may exist, such as a guiding system (radar or thermal) with an association to the tail-kit.
- **Factor 4:** High loadings under factor 4 are the most difficult to interpret. These loadings are most generally associated with documents with heavy emphasis on words and themes associated with "air", "strike", and "ground", indicating air-launched systems. Higher factor 4 loadings associated with the B28 nuclear bomb, the W80 warhead (and potentially associated the future delivery vehicle, the Long-Range-Stand-Off weapon), and the Peacekeeper missile indicate the factor may just be associated with those systems

[25] Null-hypothesis assumes the number of factors is sufficient to estimate the covariance matrix.
[26] The lower-bound of the loading coefficients were not truncated for this example. Best-practices for sufficient factor correlation varies, and liberal loading interpretation was deemed valuable for this example.
[27] Negative loadings not considered in this analysis.

that simply “strike from above,” to include gravity systems. This is supported by the absence of a loading factor under the Tomahawk and the Standard-Missile 6.

While many of these system examples demonstrate drastic operational differences in their designs, operational uses, and system interfaces, it is undeniable that the text used to describe them can be analyzed to find hidden correlations, in much the same way that correlations can be found between the historical returns on stock prices. Though the Peacekeeper and Hellfire missiles are incredibly different in physical size and mission scope, the open source articles describing them contain text with a cosine similarity of 0.72 and a 62 percent correlation when transposed into a vector space. This similarity is captured in the Factor Analysis output, as both systems have nearly identical factor loadings. This similarity does not seem to be due to random chance, as the more obviously dissimilar two systems are, the lower these two metrics become.

Table 8 displays a sample of these metrics. Highly related systems like Sparrow and its intended replacement, AMRAAM, demonstrate high similarity scores. As expected, diametrically different systems such as Apache Helicopter and Trident missile demonstrate virtually no correlation.

Document 1	Document 2	Cosine	Correlation
AMRAAM	Sparrow	0.79	0.71
W87	W88	0.75	0.67
Apache Helicopter	Hellfire Missile	0.58	0.40
Apache Helicopter	Trident Missile	0.49	0.34

Table 8: Select Weapon System Text Comparisons

While these programs have been analyzed at the top level for this example, it is feasible that comparing requirements at the sub-component level, as well as design and production descriptions of various components, may yield more accurate analogies. New weapons systems are unlikely to have the same technical specifications as those that they are intended to replace, and some components may have more in common with assets of a different type. For example, the Joint-Air-to-Ground (JAGM) missile experienced a nearly two year delay in operational testing due to a software communication issue with the Apache helicopter [29]. This same issue was not present in the legacy Hellfire missile that the JAGM is intended to replace, implying that new or modernized software requirements were required for both JAGM and Apache. By

analyzing the system requirements of the JAGM against a library of other asset types that have already gone through operational testing (or at least more recent operational testing), the analysts may have been able to identify a potential schedule issue due to the technical descriptions and corresponding program outcomes, or may have been able to utilize more accurate inputs in the parametric models.

Why this matters?

Data remains the most important commodity for cost and economic analysis regardless of industry or sector, and practitioners should strive to apply the latest state-of-the-art techniques available to gain insights from it. Advances in analytical tools has made virtually any type of data accessible to the cost analyst, and as such, valuable data sources cannot be regarded as purely quantitative or financial. As demonstrated in this paper, qualitative information such as text can provide more efficient means of preparing and analyzing data, as well as different ways of looking at data all together. Further, the poor track record of analysts responsible for producing reliable cost and schedule estimates, well documented by oversight offices and congressional reports, necessitates advances in how traditional data sources are used. From leveraging free text fields to ensure accurate data mapping, to utilizing often set-aside program documents to identify hidden program similarities, NLP can enhance the analytical process and enable a deeper level of analysis.

Overall, it is clear NLP and ML can be a useful, and powerful, tool in the cost analyst's toolkit. The potential applications for NLP in the cost community are far-reaching, and this paper has only begun to scratch the surface of potential applications. As industry continues to blaze a path forward in the language-processing and learning fields, the public sector must continue to grow and fill the ever-growing knowledge gap. It is imperative to keep momentum in both NLP and ML to improve the credibility of estimates, offer better insights, and enhance the decision-making ability for programs and agencies throughout the federal government.

Bibliography

[1] T. N. Winter, "Roberto Busa, S.J., and the Invention of the Machine-Generated Concordance," University of Nebraska, Lincoln, 1999.

[2] The Economist, "How Data Analysis Can Enrich the Liberal Arts," London, 2020.

[3] B. W. Medlock, "Investigating Classification for Natural Language Processing Tasks," Univeristy of Cambridge, Cambridge, 2008.

[4] Y. Wang and H. Youn, "Feature Weighting Based on Inter-Category and Intra-Category Strength for Twitter Sentiment Analysis," *Applied Sciences,* 2018.

[5] L. Tanguy, N. Tulechki, A. Urieli, E. Hermann and C. Raynal, "Natural Language Processing for aviation safety reports: from classification to interactive analysis," *Computers in Industry,* pp. 80-95, 2016.

[6] M. Khachidze, M. Tsintsadze and M. Archuadze, "Natural Language Processing Based Instrument for Classification of Free Text Medical Records," *BioMed Research International,* pp. 1-10, 2016.

[7] E. M. Voorhees, "Natural Language Processing and Information Retrieval," *International Summer School on Information Extraction,* pp. 32-48, 1999.

[8] J. Chavan, "NLP: Tokenization, Stemming, Lemmatization, Bag of Words,TF-IDF, POS," 8 May 2020. [Online]. Available: https://medium.com/@jeevanchavan143/nlp-tokenization-stemming-lemmatization-bag-of-words-tf-idf-pos-7650f83c60be.

[9] V. Athanasiou and M. Maragoudakis, "A Novel, Gradient Boosting Framework for Sentiment Analysis in Languages where NLP Resources Are Not Plentiful: A Case Study for Modern Greek," *Algorithms,* p. 34, 2017.

[10] F. Sun, A. Belatreche, S. Coleman, T. M. McGinnity and Y. Li, "Pre-processing online financial text for sentiment classification: A natural language processing approach," in *Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, London, 2014.

[11] J. Silge and D. Robinson, Text Mining with R, Sebastopol: O'Reilly Media, Inc, 2017.

[12] G. James, D. Witten, T. Hastie and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, New York: Springer Science+Business Media, 2013.

[13] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica,* pp. 249-268, 2007.

[14] A. Ben-Hur, D. Horn, H. T. Siegelmann and V. Vapnik, "Support Vector Clustering," *Journal of Machine Learning Research,* pp. 125-137, 2001.

[15] F. Zhanga, H. Fleyeha, X. Wangb and M. Luc, "Construction site accident analysis using text mining and natural language processing techniques," *Automation in Cosntruction,* pp. 238-248, 2019.

[16] G. Biau, B. Cadre and L. Rouvière, "Accelerated gradient boosting," *Machine Learning,* p. 971–992, 2019.

[17] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics ,* p. 1189–1232, 2001.

[18] H. Lian, "On feature selection with principal component analysis for one-class SVM," *Pattern Recognition Letters,* pp. 1027-1031, 2012.

[19] L. Rocchi, C. Lorenzo, A. Cappello and F. B. Horak, "Identification of distinct characteristics of postural sway in Parkinson's disease: A feature selection procedure based on principal component analysis," *Neuroscience Letters,* pp. 140-145, 2006.

[20] F. Gargiulo, S. Silvestri and M. Ciampi, "A clustering based methodology to support the translation of medical specifications to software models," *Applied Soft Computing,* p. 199–212, 2018 .

[21] The White House, "The Office of Federal Procurement Policy," The White House, 2021. [Online]. Available: https://www.whitehouse.gov/omb/management/office-federal-procurement-policy/. [Accessed 02 2021].

[22] J. Barnet, "AI for Everybody: JAIC unveils more initatives to DoD Bolster business systems," FedScoop, Washington, 2020.

[23] *National Defense Authorization Act for Fiscal Year 2019,* 2018.

[24] United States Department of Defense, *Summary of the National Defense Strategy of The United States of America,* Washington, 2018.

[25] A. Natekin, "Gradient Boosting Machines, a tutorial," Technical University of Munich- Department of Informatics, 04 12 2013. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full#h3. [Accessed 12 2020].

[26] B. Li, "Distance Weighted Cosine Similarity for Text Classification," Henan University of Technology, Henan, 2013.

[27] J. Ramsey, Functional Data Analysis with R and MATLAB, Springer, 2009.

[28] Economist International, "Wikipedia is 20, and its reputation has never been higher," The Economist, London, 2021.

[29] Government Accountability Office, "Weapon Systems Annual Assessment: Knowledge Gaps Pose Risks to Sustaining Recent Positive Trends," United States Government Accountability Ooffice, Washington, 2018.

[30] G. Strang, Linear Algebra and Its Applications, Thomas, 2006.

[31] J. Ramsey, Functional Data Analysis with R and MATLAB, Springer, 2009.