3-23-2018

Text Classification of installation Support Contract Topic Models for Category Management

William C. Sevier

TEXT CLASSIFICATION OF INSTALLATION SUPPORT CONTRACT TOPIC MODELS FOR CATEGORY MANAGEMENT

THESIS

William C Sevier, First Lieutenant

AFIT-ENS-MS-18-M-161

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-18-M-161

TEXT CLASSIFICATION OF INSTALLATION SUPPORT CONTRACT TOPIC MODELS FOR CATEGORY MANAGEMENT

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

William C Sevier, BS

First Lieutenant, USAF

23 March 2018

AFIT-ENS-MS-18-M-161

TEXT CLASSIFICATION OF INSTALLATION SUPPORT CONTRACT TOPIC MODELS FOR CATEGORY MANAGEMENT

THESIS

William C Sevier, BS
First Lieutenant, USAF

Committee Membership:

Civilian Bradley R. Boehmke,
Chair

Civilian Carl R. Parson, PhD
Member

AFIT-ENS-MS-18-M-161

Abstract

Air Force Installation Contracting Agency manages nearly 18 percent of total Air Force spend, equating to approximately 57 billion dollars. To improve strategic sourcing, the organization is beginning to categorize installation-support spend and assign accountable portfolio managers to respective spend categories. A critical task in this new strategic environment includes the appropriate categorization of Air Force contracts into newly created, manageable spend categories. It has been recognized that current composite categories have the opportunity to be further distinguished into sub-categories leveraging text analytics on the contract descriptions. Furthermore, upon establishing newly constructed categories, future contracts must be classified into these newly constructed categories in order to be strategically managed. This research proposes a methodological framework for using Latent Dirichlet Allocation to sculpt categories from the natural distribution of contract topics, and assesses the appropriateness of supervised learning classification algorithms such as Support Vector Machines, Random Forests, and Weighted K-Nearest Neighbors models to classify future unseen contracts. The results suggest a significant improvement in modeled spend categories over the existing categories, facilitating more accurate classification of unseen contracts into their respective sub-categories.

for my family, friends, and brothers.

Acknowledgements

I would like to thank my research advisor, Dr. Bradley Boehmke, for providing me with the tools I needed to to produce this work, and for the mentorship in all things data science. I would also like to thank AFIT professor, Dr. Jason Freels for creating an maintaining an invaluable template for which to generate a thesis in R.

Table of Contents

		Page
	Abstract	iv
	List of Tables	ix
	List of Figures	x
I	**Introduction**	1
1.1	Motivation	2
1.2	Research Objectives	2
1.3	Assumptions and Limitations	3
1.4	Thesis Outline	4
II	**Literature Review**	5
2.1	Current Methods	5
2.2	Knowledge Discovery in Databases	7
2.3	Machine Learning	7
2.4	Text Mining	8
2.5	Text Summarization	9
2.6	Unsupervised Learning Methods	10
	Clustering Methods	10
	Topic Modeling	11
2.7	Supervised Learning Methods	12
	Text Classification	12
2.8	Probabilistic Methods	13
III	**Methodology**	15
3.1	Overview	15
3.2	Statistical Tools	16
3.3	Data Source	17
3.4	Exploratory Analysis	19
3.5	Data Preparation	22
3.6	Pre-Processing	24
	Tokenization	24
	Filtering	24
	Lemmatization and Stemming	25

Vector Space Model . . . 25
Target Values . . . 26
Document-Term Matrix . . . 28
3.7 Classification . . . 30
Sampling . . . 31
Weighted K Nearest Neighbors . . . 33
Random Forest Classifier . . . 35
Support Vector Classifier . . . 38
3.8 Confusion Matrix . . . 41
3.9 Latent Dirichlet Allocation . . . 43
Optimal Topic Number . . . 44
IV Analysis And Results . . . 48
4.1 Chapter Overview . . . 48
4.2 Initial Accuracy Results . . . 48
4.3 Oversampling and SMOTE . . . 49
4.4 Optimal Topics Analysis for Subcategories of PSC Category 70 . . . 51
Accuracy of Contructed Category Classification . . . 53
Evaluating Misclassification . . . 55
4.5 Optimal Topics Analysis for IT Categories . . . 57
V Conclusion and Future Research . . . 61
VI Appendix . . . 63
6.1 Packages . . . 63
6.2 Functions . . . 63
6.3 Methodology Code . . . 72
Bibliography . . . 80

List of Tables

Table		Page
1	PSC Category Description and Count	19
2	Top Term by PSC Category	21
3	Contract Data Considered for Text Analysis	23
4	Sample of Assigned Target Values	27
5	Number of Classes per PSC Category	28
6	Document-Term Matrix Sample	29
7	Proportion of Document Classes, unbalanced	32
8	Proportion of Document Classes, Oversampling-balanced	32
9	Proportion of Document Classes, SMOTE-balanced	33
10	Confusion Table Metrics	42
11	Optimal Topics Analysis Results With Topic Ranking	47
12	Classification with Crossvalidation Models, tf and tf-idf	49
13	Classification with Crossvalidation Models, Oversampling and SMOTE	50
14	Classification with Crossvalidation Models, tf and tf-idf, 8 Topics	53
15	Accuracy Statistics of Classification Models	53
16	Classification Computation Time	54
17	Misclassified Contracts	56
18	Most Frequently Misclassified Words	57
19	Classification with Crossvalidation Models, tf, 10/6/3 Topics	58

List of Figures

Figure		Page
1	OSD Portfolio Group Taxonomy	6
2	Suffix Array for Term Frequency [47]	9
3	Text Classification Process [17]	12
4	Proposed Methodology Flow for Contract Document Classification	15
5	Distribution of Product Service Code Categories	18
6	Distribution of Number of Terms vs. Number of Unique Terms per PSC Category	20
7	Term Distribution per PSC Category	22
8	Document-Term Matrix Sparsity as Feature Reduction	30
9	KNN Classifier Accuracy vs. K	35
10	OOB Error by Number Trees	36
11	Random Forest CV Tuning for Mtry	37
12	Feature Importance Plot	38
13	SVM Cross-Validation Plot	40
14	Confusion Matrix for Level 2 Classification of Category 70	41
15	Plate Notation of LDA	43
16	Topic Number Analysis	46
17	Per-Topic Per-Word Probabilities	51
18	Pairwise Log2 Ratio of Beta Comparison	52
19	Confusion Matrix of PSC 70 Classifications with 8 Topics	55
20	Topics Optimality Analysis on PSC Categories	58
21	6-topic Model on PSC Categories	59

TEXT CLASSIFICATION OF INSTALLATION SUPPORT CONTRACT TOPIC MODELS FOR CATEGORY MANAGEMENT

I. Introduction

Contracting practices in the Department of Defense (DoD) strive to appropriately allocate management of contract categories. Currently, installation support contracts constitute a significant portion of obligations, with broad categories comprised of varying goods or services being managed by the same organizational entities. In 2014, a concept of operations (CONOPS) proposed a category management and strategic sourcing approach to spend analysis and supply chain management [1]. This delegation of responsibility over spend categories that include a wide spectrum of different contract types is inhibiting efficient oversight at the cost of increased DoD spending. Installation Support spend represents 17.82 percent of total Air Force contract spend, summing to approximately 58 billion dollars over the last 5 years [2]. These general categories could be further distinguished into sub-categories using text analysis techniques, and future contracts be classified to these categories through implementation of machine learning techniques, providing improved efficiency in contract management and could show potential savings due to active consumption shaping by portfolio managers.

[1] Muir, Keller, Knight. "Category Management: A Concept of Operations For Improving Costs At The Air Force Installation". United States Air Force.2014

[2] http://www.afimsc.af.mil/Units/Air-Force-Installation-Contracting-Agency/

1.1 Motivation

In 2014, the office of management and budget issued a memorandum outlining the importance of category management, or the management of procurement over entire categories of common spend rather than individual units, and the plans to roll out Category Management and Strategic Sourcing best practices government-wide [32]. In this way, government procurement practices would more closely resemble those of the private sector. Specifically, category management of Information Technology (IT) contracts were highlighted as an area for which category management could prove beneficial in reducing costs and limiting duplicate contracts. Further, one of the objectives of this implementation is to leverage innovations in technology to facilitate the government's, and by extension the Department of DoD's, adoption of the category management solution. This research aims to evaluate in which ways technology may be leveraged, in the form of text analysis and machine learning, to best aid analysts in the implementation of category management.

1.2 Research Objectives

For Category Management to be effective, there needs to be clear categories for which to group common spend. Text analysis provides a tool for which to systematically compare the relation of contracts by commonalities in their contract description. Similarities between the frequency of words in a contract description should allow analysts to group contracts of similar spend into a common category. To this end,it is proposed that through use of Latent Dirichlet Allocation (LDA), topic models may be constructed for which to categorize IT contracts. Using the probability of association of certain words with their respective topics, contracts may be categorized by the words they contain, and the frequency in which they are used. In this endeavor, the following are to be explored: how can supervised classification techniques, specifically Support Vector Machines (SVM), Random Forests, and Weighted

K-Nearest Neighbors (K-NN) be used to classify future contract obligations into identified contract categories? Concurrently, which method would be most accurate and practical? Although this research proposes a methodology for text analysis and classification, this research will be the foundation to develop an analysis tool in the form of a R-Package for text data analysis and classification. This tool will accomplish pre-processing of the text, convert to document term matrix, partition over train and test splits, fit to the data an ensemble of classification models, validate with the test data, and return metrics and graphics for model comparison and results. The process of this analytical tool will flow naturally from topic modeling analysis on the same data set, providing target categories on which the classification models may be trained. This will provide the analyst or user not only with an insightful model representing the content of contract categories, but also a system for building classification algorithms for future contract classification.

1.3 Assumptions and Limitations

The research is constrained by several limitations and bound to several assumptions. First and foremost, there is available a limited number of contracts for use in this study, each with brief text fields of varying length. Using the description of the contract, text analysis will on short-text rather than the lengthy narratives for which these tools were intended. As the intent of the research is to explore a feasible and reproducible methodology for government analysts, only the most practical statistical learning algorithms were considered, exempting more involved model-building, for example neural networks. The research is bound by the assumption that the data provided is representative of the current state of installation support contract processing, and also that the subset of the data used in this research is parallels the format and structure of the omitted data. Most importantly, however, it is assumed that the description provided for each contract accurately represents what the contract requires, or relavant contract characteristics, rather than mirrors the category to

which it was submitted.

1.4 Thesis Outline

Chapter 2 serves as a literary review of published literature relevant to the research. Topics reviewed will include current methods of categorizing contracts within the DoD and solution approaches to the problem of categorizing and classifying new contracts into their respective categories. Approaches to analyzing and categorizing contracts based on their inputs can be distinguished in topics regarding wrangling text data, preprocessing of text data for use in machine learning processes, and classification and clustering methods with categorical and continuous features. Chapter 3 explains the data source used in this study and the preparation that went into cleaning the data before the analysis could begin. Chapter 4 describes the analysis of the classification algorithms, topic modeling, and misclassification exploration. Chapter 5 concludes the study and gives recommendations for future research.

II. Literature Review

2.1 Current Methods

Category Management is the strategic management of spend across an organization by category [27]. These spend categories are comprised of contract obligations of similar type. Management of many of these spend categories is delegated to various entities, and although they are responsible for this spend, may not have the resources or experience to appropriately manage such a diverse portfolio. This is the case with the majority of installation support contracts, as they are comprised of a spectrum of goods and services. Air Force Installation Contract Agency is responsible for the management and execution of installation support contracts for the service. They outlined four primary actions in order better implement category management in the Air Force:

> First, the Air Force must adopt centristic strategic supply management practices for installation support and assign portfolios of spend categories to champions. Second, champions must be responsible and accountable for controlling and improving category costs and consumption across the enterprise and must be empowered to shape consumption and drive purchasing behavior. Third, champions must assign managers for categories who possess or can obtain expert domain knowledge in their categories. Finally, the Air Force must improve its capabilities for business intelligence to support managers, the definition and analysis of their categories, and the long-term development of category improvement initiatives; a centralized Business Intelligence Competency Center is recommended for this purpose [27].

In order to achieve this goal, it is important to shift away from the current taxonomy of contract management in Air Force installation support contracts to a system more conducive to the strategic management of spend categories, thus reducing cost through *active* consumption shaping. A proposal of an improved system is outlined below.

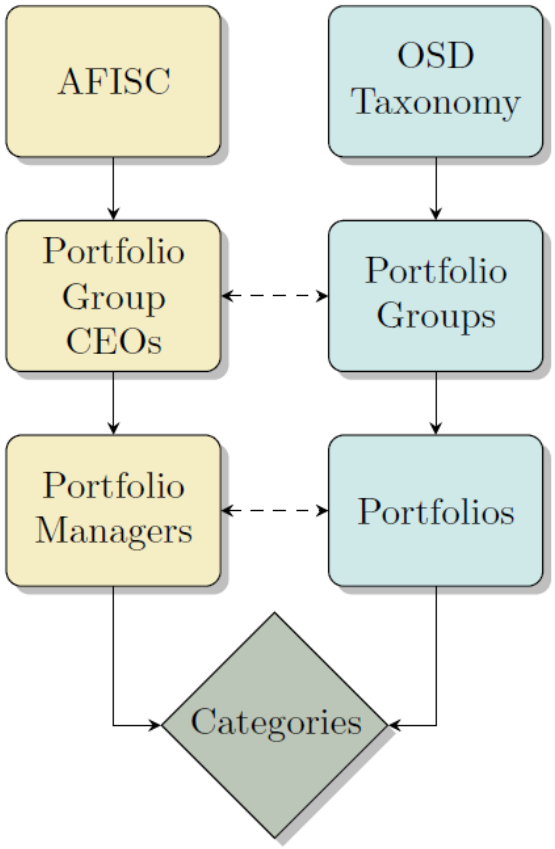

Figure 1. OSD Portfolio Group Taxonomy

This portfolio group taxonomy is currently implemented at the Office of the Secretary of Defense (OSD), and provides each portfolio with the expertise needed to manage and shape the spend category. Management is broken down into "Portfolio Managers" who are responsible for an individual portfolio of which they are a subject matter expert, and "Portfolio Group CEOs" who would manage a group of spend categories [27]. This system is dependent on the accurate categorization of incoming installation support contracts, and allowance of portfolio managers the flexibility to define their categories and to later refine

them, if needed [26]. However, the proposed CONOPS prescribes terms frequency as the analytic approach for accurately categorizing spend categories. This would allow for analysts to portray the spend categories in an insightful but relatively superficial way using the text fields of the contracts. There exists a more extensive approach to shaping spend categories, namely with inclusion of inverse document frequency (providing also the frequency in which the term appears in each document) and topic modeling.

2.2 Knowledge Discovery in Databases

Knowledge discovery in databases (KDD) is defined as "extracting extracting implicit valid, new, and potentially useful information from data, which is non-trivial" [9, p. 82]. The tool for which to do this is referred to as *data mining*, in which algorithms are used to extract patters from data. KDD is the overall process for information extraction, whereas data mining is only one specific step in the KDD process. Although databases insinuate structure to data, the KDD process can also be implemented on more unstructured data, such as text.

2.3 Machine Learning

Machine learning is a branch of artificial intelligence and is commonly used for data mining. It aims to leverage statistical methods and algorithms to learn patterns in the data and use the learned patterns to provide predictions on attributes in unseen data. In this way new data can utilize trained models in order to automatically extract information from similar data for which certain attributes are not necessarily known [25]. Text classification is a sub-domain of machine learning, used to classify unknown text documents on learned features from training documents.

2.4 Text Mining

Text mining, initially dubbed knowledge discovery from text (KDT) [10] is a type of knowledge discover process tailored for unstructured text data. This process uses algorithms for analyzing large amount of text data, for which it is unfeasible to manually extract information. The necessity for text mining came about as databases were no longer limited to storing structure data, expanding to store text documents. The appropriate use of text-based data, especially that of free-form data entry, remains a challenge to many researchers. The lack of restriction on free-form allows for significant amounts of information to be drawn from each entry. However, challenges remain in capitalizing on this potential, as unpredictable text-based data is difficult to impose data analytics upon. Several methods exist in order to better organize and analyze this data. Natural Language Processing provides researchers tools for deriving statistics and quantifying characteristics of data comprised of natural language terms [39]. Many Natural Language Processing techniques use some common elements. Term Frequency (tf) is the process of counting the number of times a text object (word, term, or n-gram) appears in a *document* (the individual text field of focus). An n-gram is a collection of n number of letters from text data. These can take the form of syllables or roots of a word, or merely pairs of letters. One approach to indexing the corpus of a data set of text is through suffix arrays. A suffix array is "a data structure... for on-line string searches" [23], and facilitates the querying of strings in a body of text with computational efficiency. This structure can be used to calculate the term frequency and document frequency (df) (number of times term is used in each document).

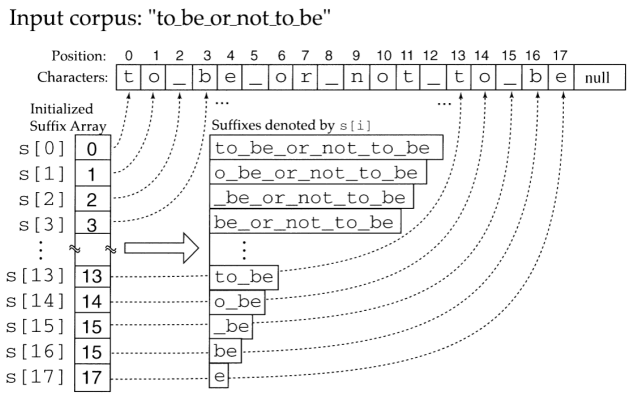

Figure 2. Suffix Array for Term Frequency [47]

Term frequency and document frequency allows for automatic text analysis on large text data sets. Term Frequency-Inverse Document Frequency is a statistic used to quantify the weighted value of each term in respect to its frequency within each document in the corpora. This comes in the form of a matrix with columns containing *tf-idf* values for each document, indexed by the respective term of that weighted frequency count [2]. Several weighting schemes have been tested, finding that "text indexing systems based on the assignment of appropriately weighted single terms produce retrieval results that are superior to those obtainable with other more elaborate text representations" [34, p. 10]. In implementing the *tf-idf* method, one can derive a summary of themes for each document, or for the set of documents. In doing so, this method would be superior to the unweighted term frequency approach to categorizing contracts initially proposed by AFICA.

2.5 Text Summarization

Text documents are generally comprised of one or more topics throughout, with many documents potentially being of similar topics. The ability to summarize text documents based on the words in their corpus is known as extractive summarization. In this way, the words used in each document provides insight as to the topic of the document of the whole, and may be used to provide a summary of what the document could be about. This study

is fundamental for grouping documents together based on similarities in summary [30].

2.6 Unsupervised Learning Methods

Unsupervised learning is the process of extracting hidden structures in unlabeled data. Unlabeled data are any data that do not have associated labels that associate the data to a specific class, category, or target value of interest. They are unsupervised in the sense that they do not require training, and assess statistical properties of the data using structures that are inherent to the data. The most common unsupervised learning methods are clustering and topic modeling.

2.6.1 Clustering Methods

Although Natural Language Processing is effective for extracting summary analysis and tagging documents with topics, unsupervised learning clustering algorithms are able to classify documents based on their similarity to other documents. Clustering is "a very broad set of techniques for finding subgroups, or clusters, in a data set" [18]. The most popular methods of clustering for use in machine learning are K-means clustering and Hierarchical Clustering. K-means clustering requires that the user define the number of clusters that the algorithm should implement, while hierarchical uses distance between entries to determine where cluster separations lie based on similarity. Document clustering using hierarchical methods is regarded a better quality approach, however, its time complexity makes it less practical for large data sets [41]. An alternative approach defined as *soft* clustering allows for mixture models as described earlier by Latent Dirichlet Analysis and can better accommodate for outliers and prevent small subclusters from inaccurately being absorbed into larger clusters nearby [14].

Several comparisons of clustering techniques are available, and specifically those involving clustering based on both categorical and continuous features. A study by Steinbach

provided an alternative to these approaches in the form of a K-means clustering variant, bisecting K-means. In this method, clusters are iteratively split based on similarity until the number of clusters required is achieved. This differs from K-means, which initially begins defining n clusters randomly until convergence. Bisecting K-means proved to be as good as or better than both K-means and Hierarchical approaches and has the added appeal of computational runtime on the order of n, rather than quadratic time complexity [41]. Two other algorithms have been presented for use specifically with mixed data types. These algorithms are K-prototypes, and fuzzy SV-k-modes. K-prototypes allows for the capturing of mixed data characteristics through use of prototypes, which store information about distributional characteristics rather than rely on the mean values of the clusters [15]. K-prototypes allow for the use of non-numerical features while retaining the efficiency of the K-means algorithm [19]. Further improving on the K-modes and K-prototypes algorithms proposed by Huang, fuzzy SV-k-modes algorithms provides a more efficient approach at clustering with mixed data and set-valued attributes. This specifically can be implemented to text queries that have brief document corpora much like the contract inputs for Installation Support. SV-k-modes allows again for the efficiency of k-means clustering without the data type restriction [3].

2.6.2 Topic Modeling

Topic modeling is very similar to clustering. Documents can be assessed for similarities based on the terms they contain. Similar documents can thus be grouped together based on their containing vocabulary. As each topic is a probabilistic distribution over words, and documents are a probabilistic distribution over topics, the topics of each of the clustered documents can be modeled based on each terms probabilistic association with each of the topics, providing context to the topics. In addition, documents can be evaluated based on their probabilistic association with each of the topics [42].

2.7 Supervised Learning Methods

Supervised Learning Methods differ from unsupervised learning in the sense that they require labeled data. Labeled data are data that have an attributed target value to each of the observations. Using features of the data, and the class labels, algorithms can create a classification function or use an instance-based approach to create a decision boundary. Text Classification is an example of supervised machine learning, as each document has a corresponding class or topic, which can be used to train a classifier to then predict the class or topic of unlabeled or unseen documents [25].

2.7.1 Text Classification

Text classification, automated using machine learning processes, has grown in popularity as computational power has become faster and more accessible. The realized potential of seamlessly indexing and sorting documents based on their predicted themes or topics has made exploration into classification techniques extensive and ever-improving. The process for preprocessing text data, training, and testing a model for text classification is as follows:

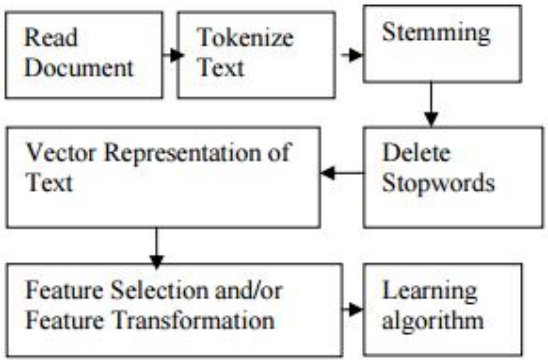

Figure 3. Text Classification Process [17]

Research published by Kahn [21] compares the numerous classification algorithms available for document representation and classification. This includes Support Vector Machines (SVM), Naive Bayes (NB), and k-Nearest Neighbors (k-NN). SVM was recognizes as the

leader in classification algorithms, in both computation time and accuracy. However, SVM fails to scale up with larger data sets, K-Nearest Neighbors is able to scale well, but relies heavily on training sets for classification accuracy. The variability of classification accuracy based on training set selection remains a challenge in document classification, as corroborated by Sebastiani [37]. This makes it difficult to train a general model that would work well with unseen data. Inference is achievable, but accurate prediction remains an obstacle. Sebstiani also finds SVM classification for text data remains superior to both neural network models and k-NN, and recommends boosted models and decision trees for further research. Although decision trees can yield promising classification results and are more robust and less prone to over-fitting, they are also computationally strenuous. The one common challenge to the advances in automated text classification is the failure of models to classify when a new corpus is introduced. There is inherent bias in document classification methods, as they train to a specific corpus, and thus have trouble classifying new documents of a different corpus. These three studies did not include cross-validation measures for model tuning in their process. This would allow for the optimal model parameters for each classification method to be tested in every case. However, exhaustive cross validation would also exacerbate the computational complexity of the models, causing computational time to be an issue [17].

2.8 Probabilistic Methods

Probabilistic text mining allows researchers to find hidden groups in data by creating probability distributions of similar statistical structures. Latent Dirichlet Allocation (LDA) and topic modeling are both examples of probabilistic models. Latent Dirichlet Allocation (LDA) provides the tools that are lacking in the classical scheme of *tf-idf*. LDA is "a generative probabilistic model for the collections of discrete data such as text corpora" [2, p. 993]. LDA uses a three-level hierarchical Bayesian mode, where each item is modeled as a mixture of the underlying set of topic probabilities. This makes LDA useful for text classification,

as each term can be predicted based on their relationship to a set of topics. The advantage of LDA over other latent variable models, is that it is able to more accurately model for inference on unseen documents.

III. Methodology

3.1 Overview

This research explores a methodological approach to text analysis implemented upon Air Force installation support contracts. This methodology is the foundation for an analysis tool for text data that will accomplish pre-processing of the text, convert to document term matrix, correct for class imbalances, partition over train and test splits, fit to the data an ensemble of classification models, validate with the test data, and return metrics and graphics for model comparison and results. The methodology of this tool will flow naturally from topic modeling analysis on the same data set, providing target categories on which the classification models may be trained.

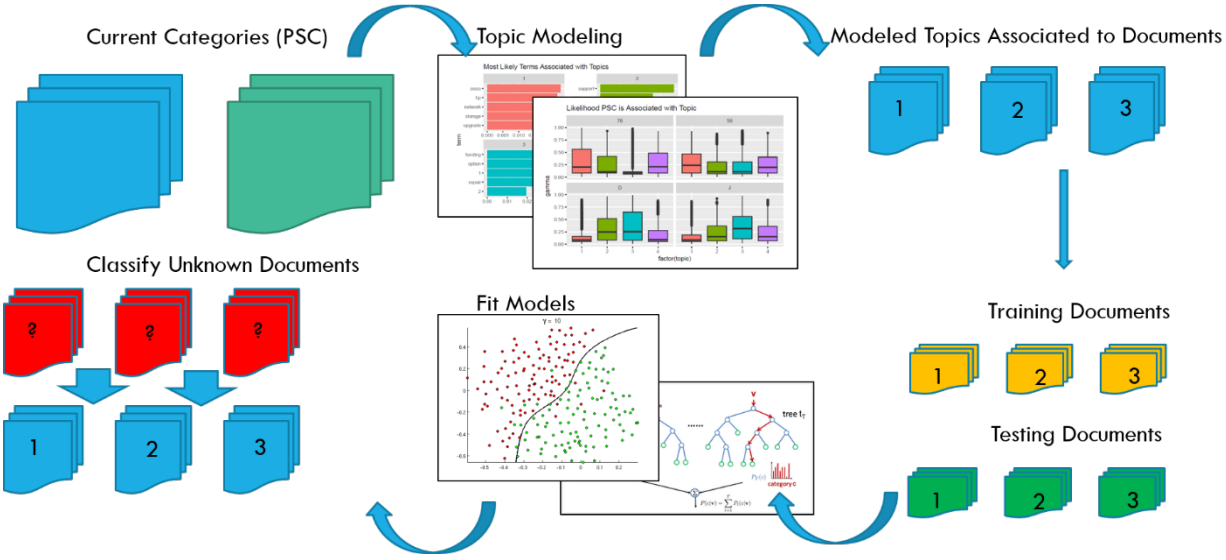

Figure 4. Proposed Methodology Flow for Contract Document Classification

In this paper, the data is explored from a structural point of view, and then also from a text analytic approach, investigating term frequencies and term distributions for each of the

PSC Categories. Each step of the methodology will be accomplished within the scope of the most data-abundant PSC Category (Category 70: ADP Equipment Support, and Software), before being expanded to the entirety of the categories. The former will model subcategories (level 2 categories) from PSC category 70 , while the latter will re-model all current PSC categories, assessing classification potential for both. For comparison of legacy category constructs to the proposed method, initial classification utilizes the legacy level 2 categories for which to train classification models. Topic modeling allows for themes of categories to be subsequently extracted. The extracted topics are then to be associated with their respective contract documents based on likelihood of association between document description and topic terms. The associated topics are then used as target values for classification training of the newly constructed categories. A test partition of these documents are used for validation of the model, or cross-validation techniques are implemented, and the models are compared for accuracy and computation complexity and compared to the legacy category viability for text classification. Further, Misclassifications can be examined to determine potential causes, and best practices can be recommended. Implementing text classification will not only allow for the construction of classification models for the categorization of future unseen contracts, but also provide insights as to current potential misclassifications with shaped topics.

3.2 Statistical Tools

The raw data was imported into R: a statistical computing environment [29] using the R-Studio IDE for ease of data management and visualization. Several statistical and data-wrangling packages were used throughout the research:

- **data.table** importing and conversion to structured dataframe, as well as for data manipulation [8]
- **tidyverse** data-wrangling, manipulation, and visualization [46]

- **psych** statiscal summaries in table form [31]
- **tidytext** topic modeling and text analysis and visualization [38]
- **scales** scaling data for visualization [45]
- **class** wrapper for classification evaluation in R [43]
- **randomForest** Random Forest cross validation, training, and analysis for R [22]
- **e1071** Support Vector Machines cross validation, training, and analysis for R [24]
- **kknn** Weighted K Nearest Neighbors cross validation, training, and analysis for R [36]

3.3 Data Source

The source of the data used for this research are the installation contract obligations for Information Technology (IT) spending. The contract data are pulled from the Federal Procurement Data System - Next Generation (FPDS-NG). The provided sample of (IT) contracts include description fields that contain manually-inputted short-text information about the contracts, some of which are only vaguely descriptive and loosely related to the categories under which they are managed. The data set is sourced from Air Force Installation Contract Agency. The raw data is in comma-separated values table format. Included in the contract information are many variables including cost, contract source information, and specialty codes. However, only the contract descriptions will be isolated and used for text classification, with the product service codes, product service code categories, and contract IDs used to organize and distinguish contracts. Each contract description is intended to describe for what purchase the contract serves. The product service codes and produce service code categories are used by the DoD to define categories for which contracting agents use to sort submitted contract transactions. The following tables summarize the number of contracts for their respective product service codes.

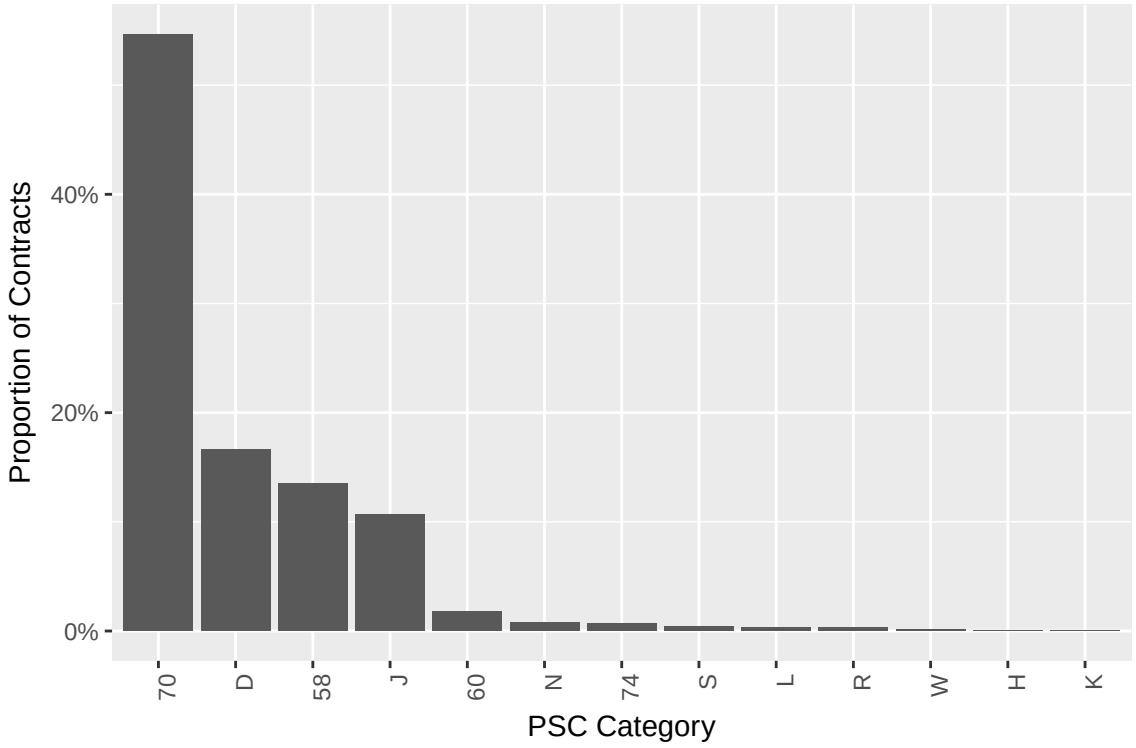

Figure 5. Distribution of Product Service Code Categories

These Product Service Codes (PSC) depicted in Figure 5 correspond to different categories of DoD spending. Each of these categories are in the realm of information technology, but their individual categories differ greatly. The majority of the contracts represent Automated Data Processing (ADP) Software, Supplies and Equipment. More than half of the IT contracts in the data come from this category, with over 90 percent of the total contracts associated with categories `70`, `D`, `58`, and `J`.

Table 1. PSC Category Description and Count

PSC Category	PSC Description	Number Contracts	Percent
70	ADP EQPT/SOFTWARE/SUPPLIES AND EQPT	18622	0.5464362
D	ADP AND TELECOMMUNICATIONS	5678	0.1666129
58	COMM/DETECT/COHERENT RADIATION	4598	0.1349218
J	MAINT, REPAIR, REBUILD EQUIPMENT	3623	0.1063118
60	FIBER OPTIC	624	0.0183104
N	INSTALLATION OF EQUIPMENT	274	0.0080401
74	OFFICE MACH/TEXT PROCESS/VISIB REC	234	0.0068664
S	UTILITIES AND HOUSEKEEPING	131	0.0038440
L	TECHNICAL REPRESENTATIVE SVCS.	104	0.0030517
R	SUPPORT SVCS (PROF, ADMIN, MGMT)	101	0.0029637
W	LEASE/RENT EQUIPMENT	59	0.0017313
H	QUALITY CONTROL, TEST, INSPECTION	21	0.0006162
K	MODIFICATION OF EQUIPMENT	10	0.0002934

In Table 1, the proportion of the PSC categories that make up the total data sample is shown. In addition, the associated PSC description for the each of the PSC categories refers to the title of each of the PSC categories, representing the expected contents of the legacy category divisions. These descriptions indicate to which category the contracts should submitted. As category 70 contracts provide the most observations with which to work, this category will be used to demonstrate the approach of text classification on modeled topics at the sub-category level.

3.4 Exploratory Analysis

Exploratory analysis establishes the characteristics and structure of the text data considered. This provides fundamental insight as to what the text data is comprised of, and the

distributions in which these terms are found in each of the categories. This characteristic is significant as statistical distinctions between contracts, and categories of contracts using text analytics depend on a frequency of common used terms between similar contracts, and unique terms used exclusively in contracts of similar content.

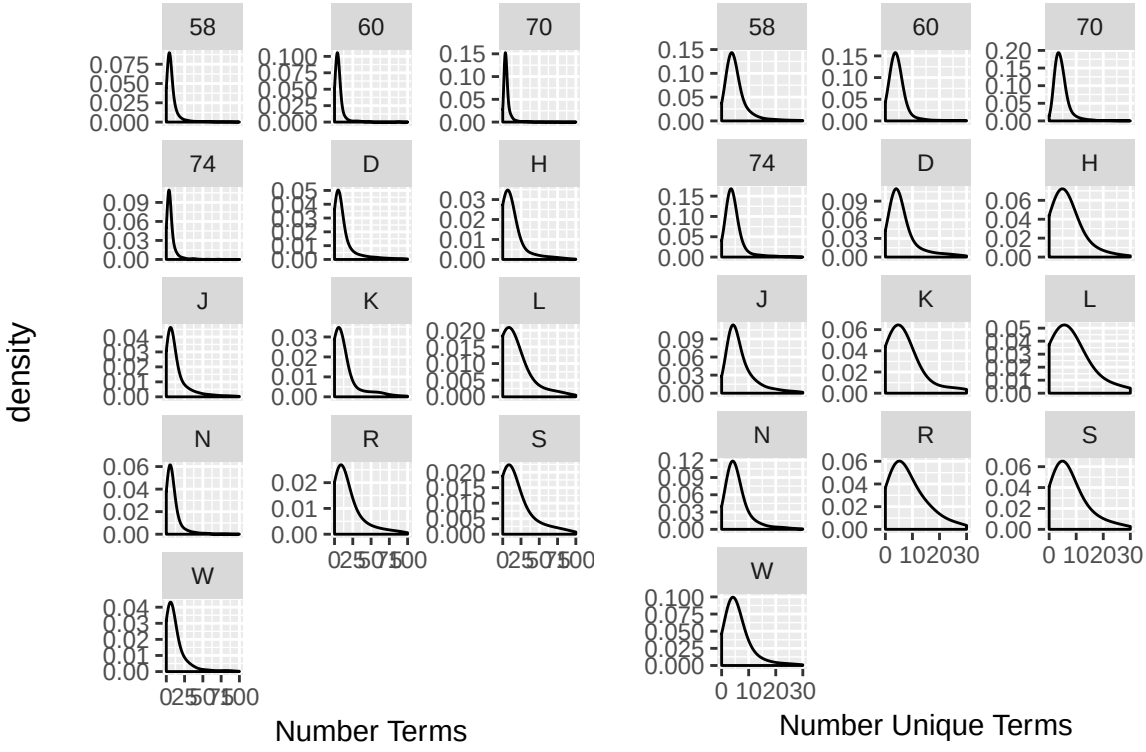

Figure 6. Distribution of Number of Terms vs. Number of Unique Terms per PSC Category

The distributions of number of terms of each of the PSC categories were first determined. Figure 6 indicates that all categories contain contracts with between 0 and 25 terms in each description field, with category 70 presenting a mean of 10.0938175 total terms in it's contract descriptions. Text analysis depends on the differentiation of documents based on distinction of terms and the frequency in which they are present in each descriptions. Therefore, as some of the contracts contain the aggregated descriptions of multiple delivery orders or transactions, it is necessary compare the number of distinct terms in the contracts of each category. There are expectedly less unique terms on average, as all categories having a mean of between 0 and 10 unique terms, and with category 70 having an average of 5.7857711

unique terms.

Table 2. Top Term by PSC Category

psc_cat	word	n
R	support	25038
J	repair	17118
S	services	9336
70	software	6489
D	support	5972
W	lease	2301
58	equipment	1856
N	installation	916
L	support	867
74	office	600
H	inspection	515
60	fiber	405
K	modification	276

Further summarization of the overall contents of the contracts associated with each PSC Category is depicted, investigating the most frequent term for each category. In Table 2, it can be determined that the following PSC categories may be explained in part by the their most frequent terms. In addition, prevalence of overlapping frequent terms provide support that these categories could be more effectively modeled, and that misclassifications could arise due to frequent terms being grossly attributed to more than one class.

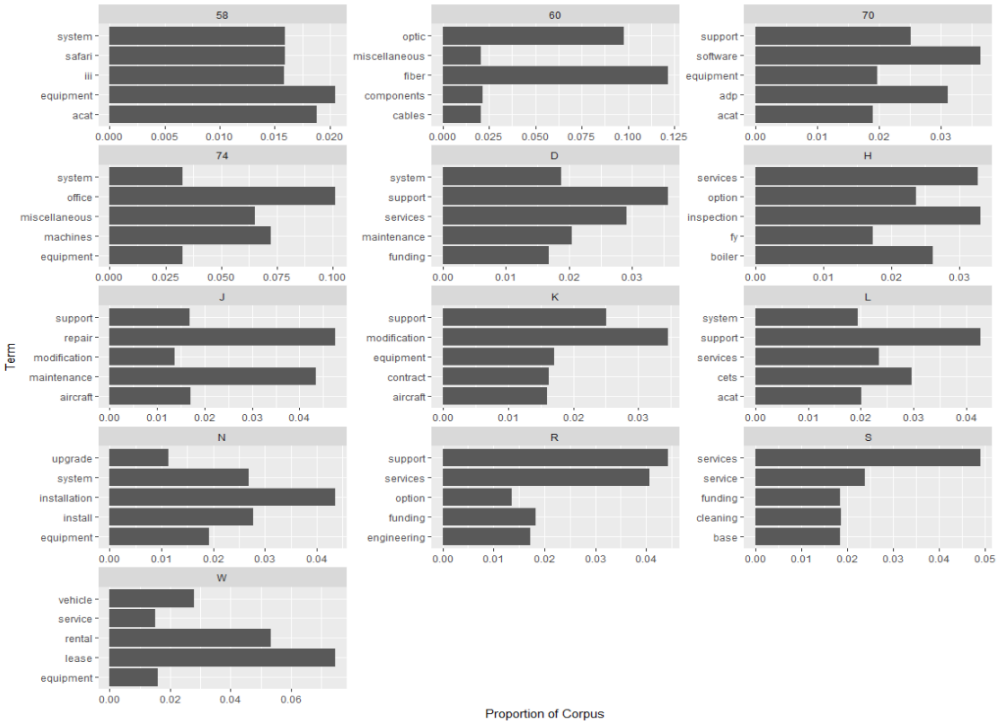

Figure 7. Term Distribution per PSC Category

Investigating the top 5 most frequent words in Figure 7 throughout all IT contract description allows for the visualization of which terms are found most often in documents of each of the categories, as well as the degree of which they are present by proportion. This asserts that many of the most frequent terms represent the most common contract actions found among all the PSC categories. However, due to the unbalanced distribution of number of PSC category contracts, these terms most likely are those most frequently found in the largest proportion of contracts, in this case those belonging to the ADP Software Supplies and Equipment.

3.5 Data Preparation

Once imported, initial data cleaning is required for further analysis. This includes removing any characters that were not alphanumeric from the description field, as well as deletion of automatically generated tags of “igf”(inherently governmental functions), “ct”" (critical

functions), “ot”" (other functions), and “cl”(closely associated). These tags are present in all contract descriptions as mandated by the FPDS-NG database, thus adding no value to the text descriptions used for analysis[1]. Further cleaning involves removing contract data where there exists no information submitted in the description field (`description`), PSC category field (`psc_cat`), or PSC code field (`psc`). Finally, the contract ID was separated into two separate columns, `contract_id`, and `transaction_id`. The contract ID is a unique 13-character code assigned to a specific contract, whereas all characters after the initial 13 identify which transaction is being referenced on the same contract. This ensured that analysis is conducted specifically on individual contracts, and not duplicate submittals of the same contracts. Further, the text field of these contracts were aggregated *per*-contract. So that each contract's description includes the text from all encompassing transactions of the same type.

Table 3. Contract Data Considered for Text Analysis

	document	psc_cat	psc	Description
63634	FA810112M0005	S	S209	PROVIDE TOWEL COVERALL CLEANIN
18726	FA873014F0030	70	7030	NETAPP CISCO EQUIPMENT
60204	FA930216ML029	R	R799	IGF OT IGF INVENTORY ORGANIZ
44787	FA449715P0086	J	J080	IGF OT IGF PREPARE AND PAINT
35094	FA850514C0002	J	J017	IGF OT IGF REPAIR REFURBISH
19213	FA877015F0506	70	7030	ACAT VIRTUAL MACHINE PROCESS
2754	FA852316F0026	58	5826	MINIATURIZED AIRBORNE GPS RECE
44096	FA812514M0006	J	J070	IGF OT IGF PREVENTIVE AND RE
67343	FA820112P0137	W	W099	CROWD CONTROL EQUIPMENT RENTAL
9853	FA330015F0092	70	7022	OFFICE DESKTOPS QEB A SEE FU

After initial cleaning, there remained 67365 observations from a total of 729659 original

[1]"FPDS-NG Data Validation Document". IBM. 2017

observations. Many of these contracts were duplicate contracts from several transactions. These contracts' descriptions were condensed and associated to one contract number. In addition, some of the omissions were warranted based on missing information or blank input fields. The structure of the data frame was compiled of the following variables: `document`, `psc_cat`, `psc`, and `description`. The document variable served as a unique ID for the contract via association with the contract number, the PSC Category variable represented level 1 PSC designator, and PSC (entire code) represented the level 2 sub-category within each respective PSC Category, shown in Table 3.

3.6 Pre-Processing

3.6.1 Tokenization

Tokenization refers to the initial preprocessing step of separating a character sequence into its individual terms or *tokens*. In this process, numerics, symbols, and punctuation were expunged from the character sequence [44]. This serves as the process for transforming a collection of terms into variables associated with each contract, rather than regarding the text as a whole.

3.6.2 Filtering

In the next step of preprocessing, filtering was accomplished. Filtering was conducted on documents in order to remove specific words from the collection of document terms [40]. The most popular method is the removal of *stop words*. Stop words are words that might appear in colloquial for syntax, or provide some other purpose other than providing information or context to the text corpus. For example, prepositions and conjunctions, words that appear so frequently that they are no longer distinguishing factors between documents (eg. *a*, *the*, etc.), and words that appear so rarely that they are unique to only one document would be

removed [33].

3.6.3 Lemmatization and Stemming

Lemmatization and *Stemming* are two text preprocessing methods that involve manipulating the remaining terms in order to extract only the valuable information from each. Lemmatization is the grouping of the various inflected form of a word so that plurals and other forms are grouped to their singular generic form. Stemming, on the other hand, obtains only the stem or root of each derived word. In this way, all words are reduced to their root form, so that verb and noun derivatives of each term are grouped (eg. concept, concepts, conceptualize, conceptualized, conception reduce to concept) [16].

3.6.4 Vector Space Model

With the corpus trimmed to only informative words or roots, the documents must be represented in numeric vector form. The converted form for numeric representation of words and documents is the *Vector Space Model* (VSM). This allows for documents containing text data to be analyzed efficiently, given large data collections [35, p. 613-620]. The concept behind this structure is that each word in the vocabulary $\mathcal{V} = \{w_1, w_2, .., w_v\}$ in document $\mathcal{D} = \{d_1, d_2, ..., d_D\}$ is represented by a numeric variable weighting the importance of the term in the document. This weighting is generally accomplished through term frequency $f_d(w)$ within each document, and the frequency of documents containing the word $f_D(w)$. Therefore, the term vector for document d can be represented by $\vec{t_d} = (f_d(w_1), f_d(w_2), ..., f_d w_v))$.

For any term $w_i \in d_j$ for each document d, frequency weighting will be assigned $\omega_i j = 0$ if the term does not appear in the document, and $\omega_i j > 0$ corresponding to the number of times it occurs in the document. Alternatively, *Term Frequency-Inverse Document Frequency* (TF-IDF) may be used to normalize these frequencies respective to how many documents in

which the term occurs. The weighting function then becomes:

$$q(w) = f_d(w) * log\frac{|\mathcal{D}|}{f_{\mathcal{D}}(w)}$$

in which the individual term weight is the term frequency within the document multiplied by the log ratio of the number of documents and the frequency in which the word appears in the all documents. The resulting term vector of weights is $\omega(d) = (\omega(d, w_1), \omega(d, w_2), ..., \omega(d, w_v)$. A set of vectors for each document creates a *document-term matrix* (DTM) [34, p. 513-523].

3.6.5 Target Values

Text analysis requires that the data first be organized according to a format by which machine learning algorithms can associate feature values to specific classification or target value. Target values provided algorithms with numerical class associations for which to train the models. To this end, the level 2 PSC codes are enumerated within their respective PSC categories. In this way, each of the PSC codes represent a single class that could be used for classification training. The final preprocessed data frame includes the document name as an unique ID, the PSC category and PSC level 2 code for the contract, the respective class or target value, and finally the corpus, or text data, for the corresponding contract.

Table 4. Sample of Assigned Target Values

document	psc_cat	psc	class	Description
FA703712P8555	70	7035	6	SEAL PRO D LAMINATOR
FA282312F3023	70	7035	6	MULTIPLE CISCO TECH REFRESH EQ
FA875112P0086	70	7010	1	DELL PRECISION T MT W
FA449715P0094	70	7050	8	GRAPHICS PRINTER CUTTER
FA440713FA068	70	7035	6	VMWARE VCENTER SERVER
FA670315FG001	70	7030	5	ADP SOFTWARE ADP SOFTWARE ADP
FA873014F0034	70	7030	5	CMMA SPT RENEWAL
FA282312C0074	70	7030	5	BASE LEVEL SOFTWARE SUPPORT BA
FA860116FG134	70	7030	5	ADP SOFTWARE
FA875116FG003	70	7025	4	ADP INPUT OUTPUT AND STORAGE D

The `class` variable represents the target values of each of the contract documents. These target values correspond to each of the level 2 Product Service Codes. Level 2 codes are user-determined categorization of the documents. As such, these classifications may be used to train classification methods for future inputted documents, based on the contents of the contract description, without the requirement for the user to determine if the category is appropriate.

Table 5. Number of Classes per PSC Category

PSC Category	N_Classes
70	8
60	12
58	20
D	25
J	78

The number of differentiable classes in each of the PSC categories were determined. Several categories appear to only consist of one level 2 PSC categorization. These categories were not included in preliminary classification as the classification algorithms require more than one target value in order to train to differentiable classes. Alternatively, some PSC categories only contain a small sample of contracts. Categories `W`, `H`, and `K`, have a sample size of less than 100. This could potentially prove problematic, as there may not be enough text data to properly differentiate between the contracts. Classification on categories that contain many classes relative to their number of observations would be futile, as there would not be enough data representing each class in the training sample to properly train to that class, and the minimal likelihood that the classes are all represented in an even samller test sample would prove the model uninformative.

3.6.6 Document-Term Matrix

The machine learning algorithms constructed required that the frequency of the terms in each of the contracts were in a format in which term frequencies represented variable values for each unique term, and each of the contracts (documents) acted as respective observations. For this the data frame was transformed into a *document-term-matrix*, a form of vector space model. In this numerical matrix structure, the corpus was scraped for words

that don't represent any relevant meaning and are used frequently in everyday colloquial. These *stop words* are words such as pronouns and words such as "the" and "as". Each variable in the matrix represents the frequency of the specific term.

Table 6. Document-Term Matrix Sample

	hp	afway	qeb	office	equipment	software	acat	adp	support	maintenance	storage	network	dell	license	system	change	modification	purchase	input	nasic
FA877110A0601	774	556	351	304	32	3	136	112	39	0	4	30	4	0	55	52	46	42	4	7
W91QUZ13A0002	0	0	0	18	2	159	43	52	6	0	0	0	0	59	5	9	5	7	0	5
FA873213D0012	3	1	0	3	165	25	61	30	37	13	16	61	5	7	16	15	19	8	3	24
FA877110A0604	2	171	68	49	28	2	39	96	24	1	16	0	2	0	8	11	21	19	16	2
FA873214D0004	15	3	1	5	151	143	66	30	102	86	46	39	42	43	37	14	17	17	3	37
FA877110A0603	0	148	48	23	22	0	43	77	22	4	13	19	5	0	12	8	10	14	13	0
W91QUZ09A0003	1	0	0	4	3	69	21	15	43	19	0	0	0	15	4	7	7	3	0	13
FA873213D0013	24	1	1	17	119	66	54	28	75	27	67	37	10	15	24	15	9	6	3	5
FA873213D0017	8	2	0	2	100	45	15	33	39	19	24	18	10	5	16	7	11	7	5	9
FA860114FG001	0	0	0	0	37	24	0	91	37	0	7	0	0	0	5	0	0	0	7	0

Sparsity represents the amount of the matrix that contains a term frequency of zero. The sparsity of the document term matrix can be manipulated by removing terms that only appear in an insignificant number of documents. These terms would likely be unique only to the document in which they are present and would add little value to modeling a topic for that class of document, or training a model to associate that term with a class. Therefore, removing sparse terms provides a form of feature reduction. Feature reduction can alleviate model bias and computational strain. The number of sparse terms chosen to remove from the document-term matrix was based on a target sparsity that was to be achieved. For this research, the document-term matrix was to be constructed with at most 95% sparsity. This sparsity was reached by removing the sparse terms that only appear in less than 2% of the documents, setting `sparse` parameter to 0.98. This allowed for the construction of a document-term matrix that has 35 terms for which to train and classify the models; a reduction of variables down from 850. Reducing the number of terms allows for a more generalized model, with the added benefit of a more practical number of variables for quicker model training.

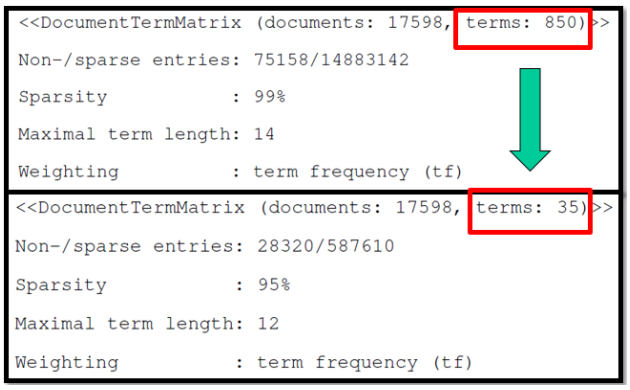

Figure 8. Document-Term Matrix Sparsity as Feature Reduction

In Figure 8, the characteristics of the document term matrix before and after removal of sparse terms is presented. It is possible to reduce the amount of features used futher, doing so in turn decreases the amount of non-sparse entries (or informative terms) as a consequence. However, removing sparse terms allows for a more efficient use of data, as the sparse terms are unlikely to aid in classification of documents, and may only increase the models' bias and overfitting potential. Although the maximal term length also decreases by two characters, this will not affect topic modeling or classification endeavors, as the use of the words, their meaning, or their length are not being leveraged, only their presence in their respective douments.

3.7 Classification

Classification as a machine learning practice has applications in many disciplines, only relatively recently finding its way into the field of text mining. Text classification specifically allows for the assignment of classes to text documents [25]. Conducted in this research was *hard* classification, in which documents have explicit category assignments for which to train model classifiers, rather than probabilistic values. Taking a subset of $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ training documents, correspondingly labeled with $\mathcal{L} = \{l_1, l_2, ..., l_n\}$ classes of the training subset [1], models may be trained to these document-label associations so that:

$$f : \mathcal{D} \rightarrow \mathcal{L} \qquad f(d) = l$$

where the documents to build the classification model f, is built using these associations. Documents unseen to the model (test set) d are then evaluated using the model, providing class labels l to each document.

3.7.1 Sampling

A data split for training and test partition was determined. The training partition was used to train each of the models, while the test partition was used to validate each model. This was accomplished by randomly sorting the observations and storing row ids of observations allocated to their respective partitions.

The training partition was used for cross-validation model tuning and subsequent training of the best model, while the test partition was used for evaluating how well the model performs with unseen data. The purpose of this technique is to limit the amount of bias in the model, mitigate over-fitting effects, and evaluate the model on unseen data, so that the model is not validated based on it's ability to merely search learned associated for the appropriate label. This allowed for built model that is generalized, and evaluated accurately according to its real-world application. This comes with a cost of variance in model accuracy, as the test partition may contain observations that are not represented accordingly in the training data. To this end, a 70/10/20 train-test split was chosen, where 70% of the data was used for training, 10% was used for cross-validation and 20% for model test runs. Investigating the data for class imbalances, it could be determined whether further sampling techniques are required.

Table 7. Proportion of Document Classes, unbalanced

1	2	3	4	5	6	7	8
0.0949505	0.0173683	0.0492869	0.1073729	0.4518058	0.160743	0.0576835	0.060789

There appeared to be a gross imbalance in the classification categories within this PSC. Class 2 represented only 0.0173683 of the classes, equating to 211 observations in the training set, and 91 observations in the test set. Imbalances in class distribution result in potential misrepresentation of model adequacy, as cross validation will determine that classifying the majority classes best (or sometimes exclusively) will result in the highest accuracy, resulting the minority or in this case "rare event" (less than 2%) classes likely going greatly misclassified. To combat this phenomenon, several sampling techniques were implemented in order to balance the classes. An oversampling method was applied to increase the minority class representation in the data through replication of the minority class observations.

Table 8. Proportion of Document Classes, Oversampling-balanced

1	2	3	4	5	6	7	8
0.0870643	0.0989822	0.0451933	0.0984549	0.4142804	0.1473923	0.0528925	0.0557401

From oversampling the minority class, the class distribution appeared far more balanced, with class 2 now representing 0.0989822 of the classes, equating to 1302 observations in the training set, and 87 observations in the test set. However, this required increasing the total number of observations, which increased computational time, and didn't fix the imbalance between the other classes and the majority class. Synthetic Minority Over-sampling Technique or SMOTE, was used to simultaneously oversample the minority class while undersampling random majority class observations. This allows for greater representation of minority class, while retaining the size of the imbalanced data, and some of the natural distribution of classes between the non-minority classes. Using SMOTE, class 2 represented 0.1035197 of

the classes, equating to 1266 observations in the training set, and 534 observations in the test set.

Table 9. Proportion of Document Classes, SMOTE-balanced

1	2	3	4	5	6	7	8
0.0853462	0.1035197	0.047504	0.1007016	0.4095353	0.1465378	0.0520474	0.0548079

These sampling techniques do not come without a cost. As more samples of the minority class are synthetically reproduced for training, it allows for over-fitting of the model [5]. Although model accuracy may increase of the minority class, it also produces a less generalized model, as the minority class sample synthetic reproduction does not increased the information gained by the model to the same extent as obtaining more data from the underrepresented class. For the SMOTE approach, the same over-fitting with the minority class was experienced, but also a loss of training data in the majority class. This may allow for greater model flexibility and less bias in classifying the majority class, but results in the loss of significant learning data which could prove detrimental to the model accuracy.

3.7.2 Weighted K Nearest Neighbors

K-nearest neighbor is a non-parametric form of classification, where an observation is classified by the class of the majority of k nearest observations. The premise being that documents belonging to the same class are more likely to be similar and therefore closer in distance [13]. For each data point The algorithm selects a observation at random, and it's class is determined by target value classifications of the k nearest surrounding observations by distance with ties broken at random. This is iteratively repeated until a decision boundary is formed. This trained decision boundary is then used to classify unseen observations, with test data falling on either side of the boundary being classified accordingly. The distance equation for which to determine the nearest neighboring points is below, where $q = 2$ would represent euclidean distance and $q = 1$ would calculate nearest neighbors by absolute distance.

$$d(x_i, x_b) = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^q \right)^{1/q}$$

This distance calculation would not suffice for the research, as the discrete nature of term frequency coupled with the number of features being considered allowed for an exorbitant number of ties being broken at random, essentially breaking the algorithm. This phenomenon disqualified the option of cross-validation and necessitated a more flexible similarity metric for use with k-NN. Therefore a kernel-weighted similarity method was used in it's place, in which the votes of each neighboring observation holds weight depending on a transformation of the normalized distance, with the transformation following several kernels: "rectangular", "triangular", "epanechnikov", "gaussian", "rank", and "optimal".

$$D(x, x_i) = \frac{d(x, x_{(i)})}{d(x, x_{(k+1)})}$$

The distance is then transformed into a weight using the respective kernel, and each observation (x, y) is classsified by the point with the heighest weight (similarity) respective to the following:

$$max_r \left(\sum_{i=1}^{k} K(D(x, x_{(i)})) I(y_{(i)} = r) \right)$$

This process is repeated itereatively until all points have been clasified and a classification boundary is learned. The classification boundary learned through training is then used to determine where the test set observations lie, and thus classifies new observations using information accumulated through previously encountered observations.

K-Nearest-Neighbors uses lazy learning allowing for quick but potentially inefficient classification, with a heavy dependency on selecting an appropriate number k. Cross-validation was completed to automatically select the optimal number k and kernel for each model.

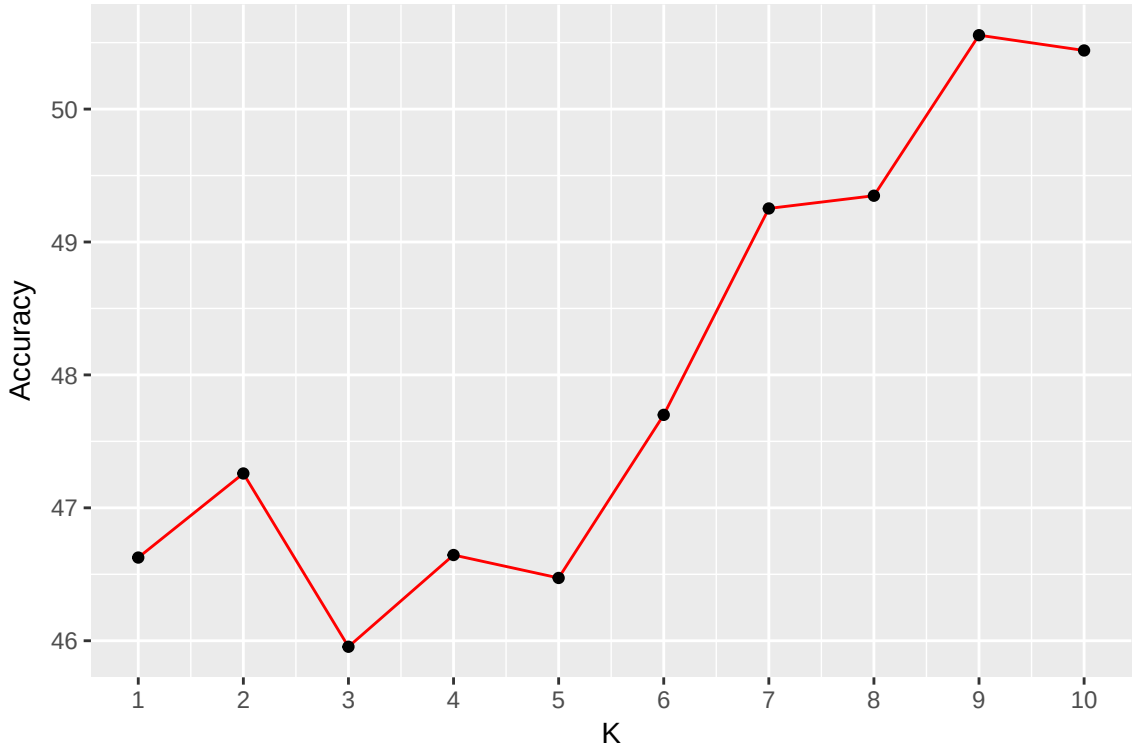

Figure 9. KNN Classifier Accuracy vs. K

Cross validation methods were used to tune the KNN model to an appropriate k value. The metric calculated to validate each model is mean training accuracy. The method implemented was 10-fold cross validation, in which the training data was partitioned into 10 equal subsets, trained on all but 1 of the partitions, and then evaluated on the partition left out of training. This is done for all 10 partitions, and the validation accuracy is averaged across all of the partitions. The k value with the highest accuracy was the selected model for subsequent test validation. In this case that value is k= 9.

3.7.3 Random Forest Classifier

Random Forests uses bootstrap aggregating of decision trees to create low-bias model from combination of de-correlated decision trees, thus mitigating the variance, in relatively quick computation time. Each decision tree is a hierarchical tree of training instances, in which the best performing features are used for that decision tree's classification model [25].

In the bootstrap aggregating (bagging) approach to random forests, every feature is used to fit a classification decision tree, using a random partition of the features (terms). As the forest grows, it picks the best selection of features, and continues to split until all features have been used in a subset tree, or a number of trees have been grown. The final random forest model is then the aggregate of all those trees. Bootstrap refers to the method of taking a random sample of the data with replacement. For this research, the number trees grown was set to 500, and the error was evaluated as the number of trees increase [12].

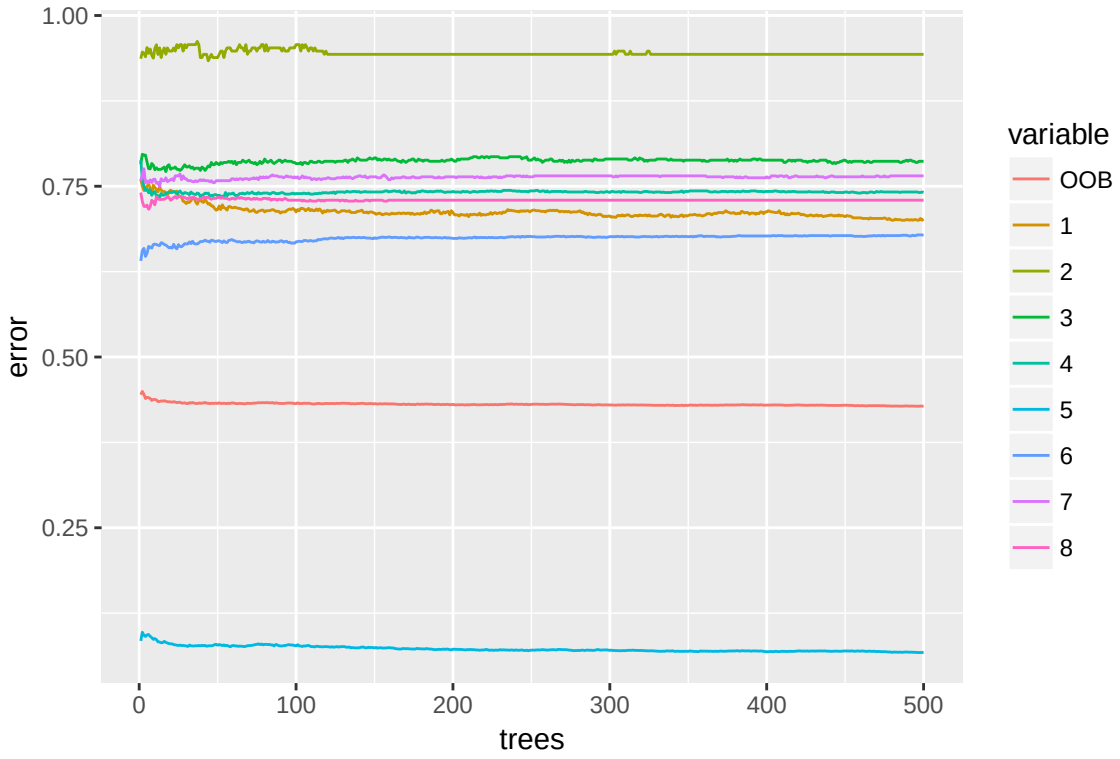

Figure 10. OOB Error by Number Trees

The Out Of Bag (OOB) error is the average classification error of those samples that were iteratively left out of the bootstrap sample. As the number of trees increases, the OOB error decreased, and it is show that the OOB error in this case converges at around 60 percent at less than 100 trees. At larger data set sizes, more trees would most likely be necessary to reach convergence of OOB error. The class-sepecific error was also examined, showing one of the classes as having a lower average error than the other, identifying that it is less

prone to misclassification. This is an indication how the 8-class sample performed without any cross-validation to determine parameter value before training the model. One class had significantly lower OOB error, and therefore could be expected to be the least frequently misclassified. This class happens to also hold the majority representation in the sample, and therefore the most data for which to train (and test) the model.

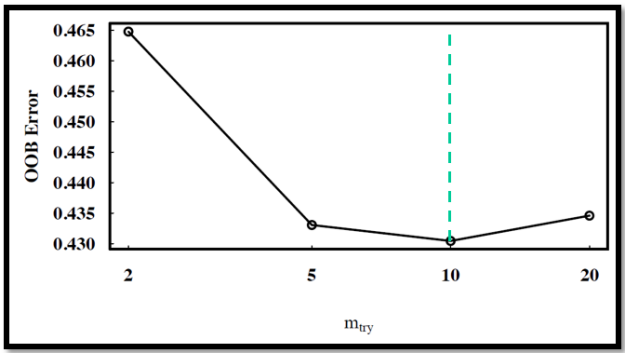

Figure 11. Random Forest CV Tuning for Mtry

Cross validation was again accomplished by evaluating the out-of-bag error at several values of mtry parameter. The mtry parameter represents the number of variables (DTM terms) to be included when splitting each tree node. Starting at the square root of the total number of variables, the number of variables included was increased or decreased by a step factor of 5. The model was assessed again at the new value of mtry, and tuning ceased when the next iteration was less than 0.1% improved over the previous iteration. In this case, the number of terms included for validation that results in the most accurate model was 5.

Supporting the assessment of model accuracy, the importance each of the terms bring to the model regarding their classification performance was analyzed. This feature was extracted through calculating the mean decrease in Gini index. This value represents the loss in classification performance if each of the respective terms were excluded from the model, providing insight as to which terms should be considered when providing descriptions for future contracts, as well as many terms that could be considered for trimming from the DTM for future models, if computational time becomes a constraint that must be overcame.

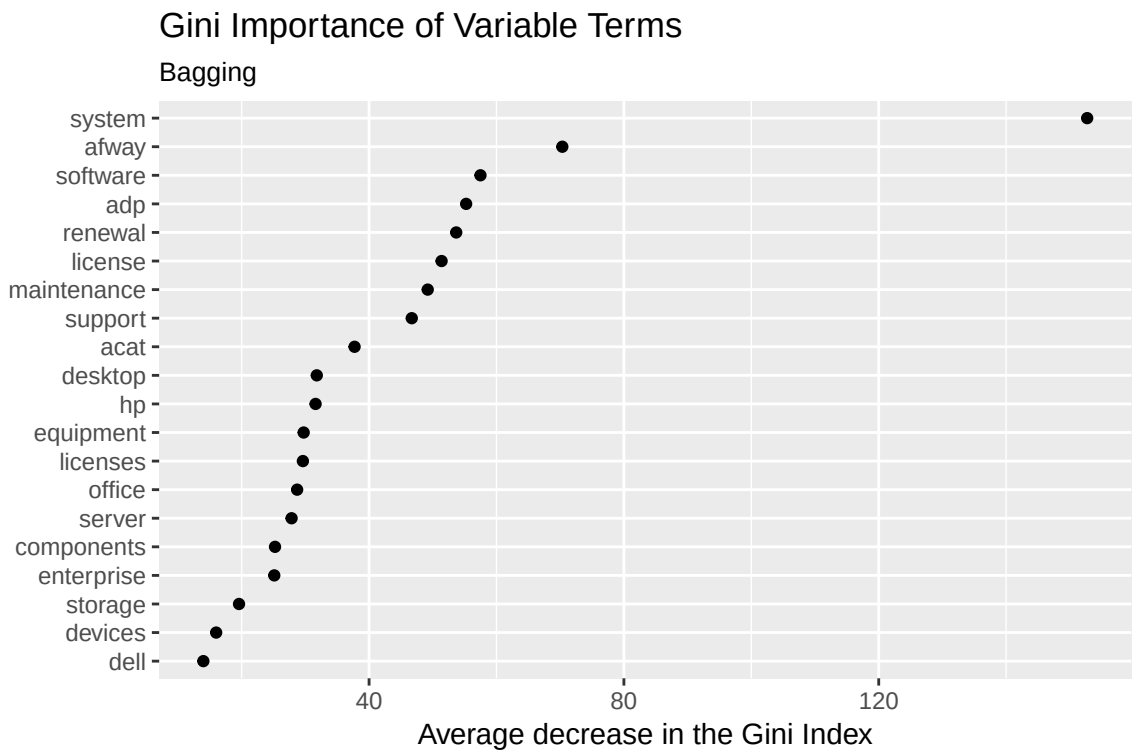

Figure 12. Feature Importance Plot

3.7.4 Support Vector Classifier

Support Vector Classifiers provide an extremely powerful, but computationally taxing algorithm for text classification. It functions based on the assertion that, in the event observations cannot be separated into classes in a 2-dimensional space, they may be projected into higher dimensional space and separated with a hyperplane as opposed to a line. Support Vector Classifiers are therefore linear classifiers, making classification decisions from a linear combination of document features [1]. In this way, predictor output can be represented by $y = \vec{a} \cdot \vec{x} + b$ as a separating hyperplane, where $\vec{x}$ is a vector of the term frequency or normalized frequency metric, and $\vec{a}$ is the vector of coefficients and b is a scalar [6, p. 273-297]. Support vectors are the closest observations to this separating hyperplane, that if removed, would shift the decision boundary, and therefore support it. As an added feature, if the data cannot be separated using a linear kernel or similarity function, in n-dimensions,

perhaps it can be separated by three dimensions employing a radial (gaussian) kernel. This choice of how the decision boundary is shaped adds increased flexibility to SVM models. However, the original SVM algorithm was developed with a linear hyperplane, and it has been found that text data, due to its sparse instances, is generally linearly separable [20]. Therefore, the radial kernel was not considered for this research. Support Vector Machines serve as an appropriate tool for text classification as it holds several helpful properties. SVMs use over-fitting protection, which does not depend on the feature space. This characteristic is invaluable as text classification inherently involve the training of many features, one for each term in the corpus.

To determine the most accurate model for implementation with the contract data, cross-validation was conducted by iteratively testing the model with changing several hyperparameters. These parameters of SVM include cost and gamma. *Cost* is the incurred penalty for allowing support vectors on the misclassified side of the decision boundary (for added leniency to form the margin). This increases model flexibility by smoothing the decision boundary. *Gamma* is a metric representing the the amount of influence support vectors further away from the margin influence its position.

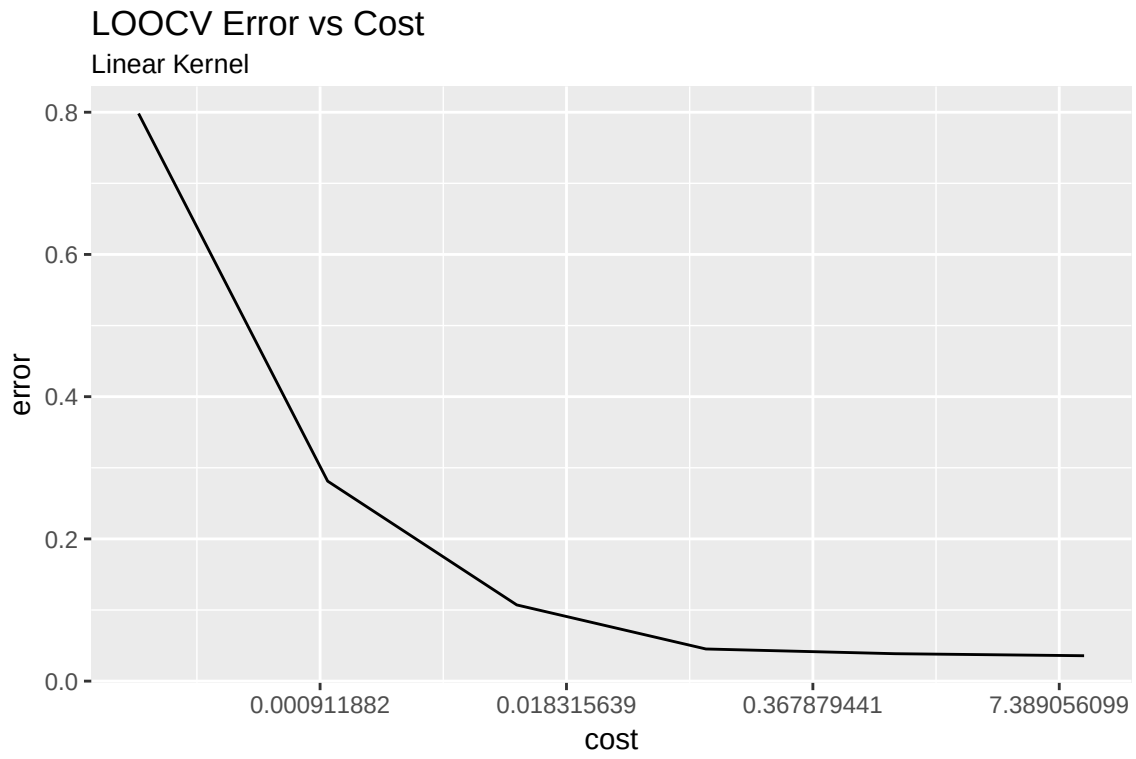

Figure 13. SVM Cross-Validation Plot

It is evident that there exists an optimal cost that correlates to a minimum test error. This best model was used for the model accuracy validation using the test partition of the data. Through Leave-One-Out Cross Validation (LOOCV), where one observation is left out of training, and iteratively validated until all observations have been tested, the performance of the model was assessed based on error for different cost values. As cost increased, the LOOCV error consequently decreased, until a certain value in which it increases again. This is due to the bias-variance trade-off. As a tighter decision margin is created through increased cost of incorrectly positioned support vectors, the bias of the model is also increased, it's performance on unseen data is potentially decreased, but allows it to perform well on the partitioned training data used for cross validation. After a certain cost value, the model is too flexible to even classify the test data but may perform better for a wider array of unknown data that may or may not resemble the training data as closely.

3.8 Confusion Matrix

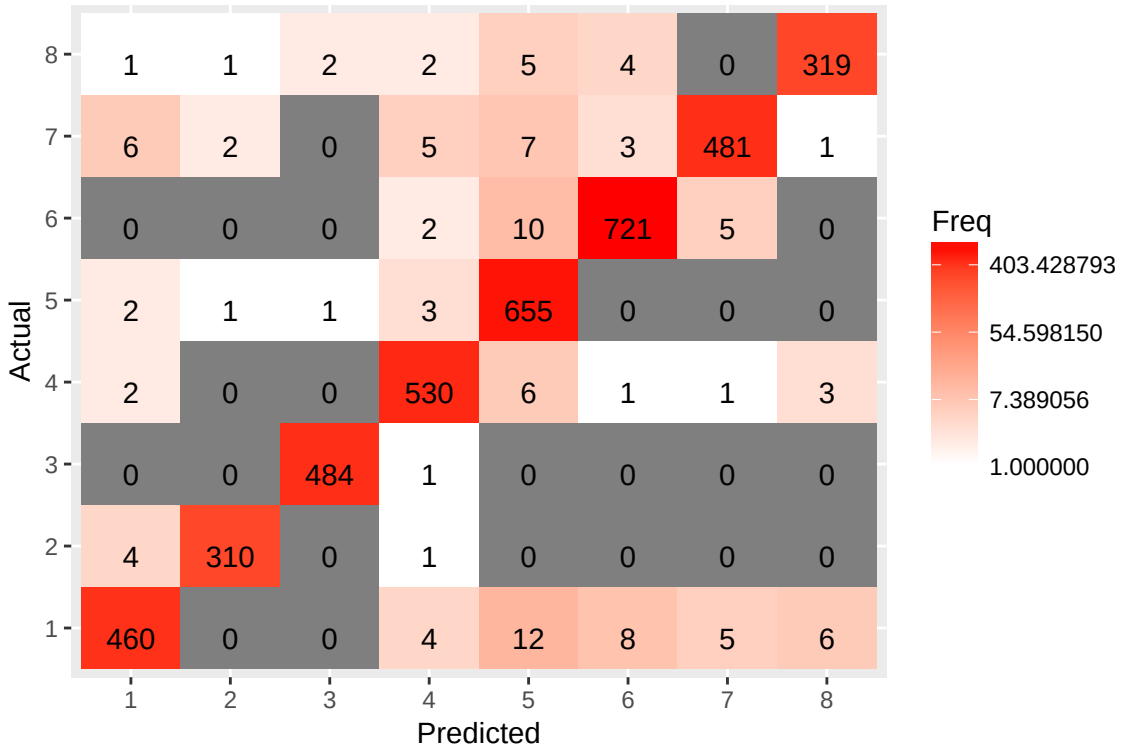

Figure 14. Confusion Matrix for Level 2 Classification of Category 70

Confusion matrices provide insight as to which class is being misclassified, and into which class they is being misclassified. This provides model performance information beyond only the accuracy of the model, but also allows for attributing misclassifications to specific documents, and explore the cause of the misclassification. In regards to practical application relevant to the data, misclassified documents can be assessed individually. These misclassification may not only be due to the error of the model, but perhaps also in the event that words were included that are not appropriate for that specific category. Alternatively, misclassification analysis could highlight documents submitted in error to that category. Each discovery would allow for provided feedback to the user as to how the contract category management system can be improved. In essence, confusion matrices can prove invaluable for investigating misclassifications in order to improve model building, but also best practices

for document description construction.

Table 10. Confusion Table Metrics

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.9292929	0.9958124	0.9684211	0.9902832	0.9684211	0.9292929	0.9484536	0.1214128	0.1128281	0.1165072	0.9625527
Class: 2	0.9841270	0.9989367	0.9872611	0.9986713	0.9872611	0.9841270	0.9856916	0.0772627	0.0760363	0.0770174	0.9915319
Class: 3	0.9979381	0.9991648	0.9938398	0.9997214	0.9938398	0.9979381	0.9958848	0.1189600	0.1187147	0.1194506	0.9985515
Class: 4	0.9760589	0.9949066	0.9671533	0.9963162	0.9671533	0.9760589	0.9715857	0.1331862	0.1299975	0.1344126	0.9854828
Class: 5	0.9894260	0.9882870	0.9424460	0.9979302	0.9424460	0.9894260	0.9653648	0.1623743	0.1606573	0.1704685	0.9888565
Class: 6	0.9769648	0.9952081	0.9782904	0.9949102	0.9782904	0.9769648	0.9776271	0.1810155	0.1768457	0.1807702	0.9860865
Class: 7	0.9524752	0.9969205	0.9776423	0.9933054	0.9776423	0.9524752	0.9648947	0.1238656	0.1179789	0.1206770	0.9746979
Class: 8	0.9550898	0.9973283	0.9696049	0.9959979	0.9696049	0.9550898	0.9622926	0.0819230	0.0782438	0.0806966	0.9762091

Confusion matrices also provide the added benefit of determining several characteristics of classification (and misclassifications) between the classes. These classification metrics are Sensitivity, Specificity, Pos Pred Value, Neg Pred Value, Prevalence, Detection Rate, Detection Prevalence, and Balanced Accuracy. Sensitivity represents the fraction of classifications that are true classifications, or the proportion of classifications that are correctly classified. Specificity represents the portion of classifications that were correctly classified as not being from a certain class. Most importantly, one metric is of interest, F1 or *F-measure*. F-measure represents the harmonic mean of the precision and recall and is regarded the "ultimate measure of performance of the classifier"[11, p. 1294]. It takes into consideration not only recall, the precentage of specific class that are classified as that class, but also precision or the precentage of classes classified as a certain class that were actual from that class. Further, the P-Value for the accuracy being greater than the NIR will be used to determine if the accuracy score can be attributed to the model rather than the distribution of classes. A P-Value less than 0.05 would support that the accuracy would signify a statistically significant probability that the accuracy is attributed to the inherent information gained from the distribution of the classes.

3.9 Latent Dirichlet Allocation

Latent Dirichlet Allocation(LDA) is a popular and effective generative topic model. In this model, each document is a mixture or distribution over words, while a topic is a distribution over topics [2]. It assumes a sparse Dirichlet prior distribution over topics in a document, using Gibb's sampling to generatively assign topic probabilities to each terms, and subsequently grouping documents into their respective topics.

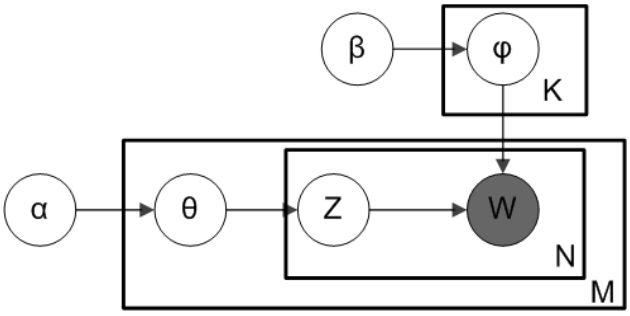

Figure 15. Plate Notation of LDA

The process is best represented via plate notation where the boxes represent replications, with the innermost box representing the replicated assignment of topics and words in a document, and the outermost box the documents assignment to a specific topic. w_{mn} are observable individual words, while θ_m (topic distribution for each document), φ_k (word distribution for each topic), and z_{mn} (assigned topic for each word in document) are unobserved latent variables, with α and β as sparse parameters controlling the degree in which the Dirichlet distribution is imposed on per-document topics and per-topic words respectively.

Given k topics, m documents, and n words per document, the algorithm chooses $\theta_i \sim Dir(\alpha)$ and $\varphi \sim Dir(\beta)$ for each document $i \in 1, ..., M$ and each topic $k \in 1, ..., K$. For each word w_{ij} in each document $i \in 1, ..., M$ and each position $j \in 1, ..., N$ choose a topic $z_{i,j} \sim Multinomial(\theta_i)$ and word $w_{i,j} \sim Multinomial(\varphi_{z_{i,j}})$ according to their respective distributions, providing a probability with which words are associated with topics.

As such, Latent Dirichlet Allocation was used to cluster the documents into topics based

on the words in each contract description. These constructed topics represent the suggested new Product Service Code sub-categories. In addition, the LDA model allows for extraction of word-per-topic probabilities and document-per-topic probabilities, which provides insights as to the composition of constructed categories. However, LDA relies on the user-defined variable k to determine the number of topics to which the documents would be classified. Therefore, further analysis was required for which to implement several proposed methods to determine the optimal number of topics to use with LDA.

3.9.1 Optimal Topic Number

The optimal number of topics for each PSC category was evaluated with respect to four metrics proposed in previous research. The metrics were used to tune the Latent Dirichlet Allocation clustering from which the new topic models are derived. In essence, the clustering quality is measured iteratively for each number of clusters from 2 to 30. From this it can be determined how many clusters would provide the most representative number of topics for which to use as a parameter for clustering using Latent Dirichlet Allocation, and thus provide insight as to how to better shape PSC categories more effectively in regards to future classification.

One method calls for the maximization of information divergence between pairs of topics. Deaveaud ([7]) proposed that a simple heuristic can be used to estimate the number of latent concepts in a set of documents by maximizing the information divergence D between all topic pairs of LDA's topics, estimating the number of topics with the following:

$$\hat{K} = \underset{K}{argmax} \frac{1}{K(K-1)} \sum_{(k,k') \in \mathbb{T}_K} D(k||k')$$

where K is the number of topics provided as the topics parameter, $\mathbb{T}_K$ is the set of topics modeled, and $D(k||k')$ is the Jensen-Shannon divergence, measuring the divergence between all pairs of topics by topic variation.

Alternatively, Juan [4] suggests a minimization of distance of topic densities. The metric used to evaluate the topic number is the pairwise cosine distance between the topics, first finding the correlation between the topics.

$$corre(T_i, T_j) = \frac{\sum_{v=0}^{V} T_{iv} T_{jv}}{\sqrt{\sum_{v=0}^{V} (T_{iv})^2} \sqrt{\sum_{v=0}^{V} (T_{jv})^2}}$$

where v is each word and T is each topic. A smaller correlation represents independence between topics. The average distance between structures or clusters is determine with the following:

$$distance(structure) = \frac{\sum_{i=0}^{K} \sum_{j=i+1}^{K} corre(T_i, T_j)}{K \times (K-1)/2}$$

where the distance between structures represents the total stability of the topic number K selected. A higher average distance is representative of higher stability and more optimal number of topics for LDA. Murzintcev [28] provided an R-package allowing for the calculation of these metrics simultaneously, where the metrics can be subsequently normalized and the results presented in an easily-interpretable plot.

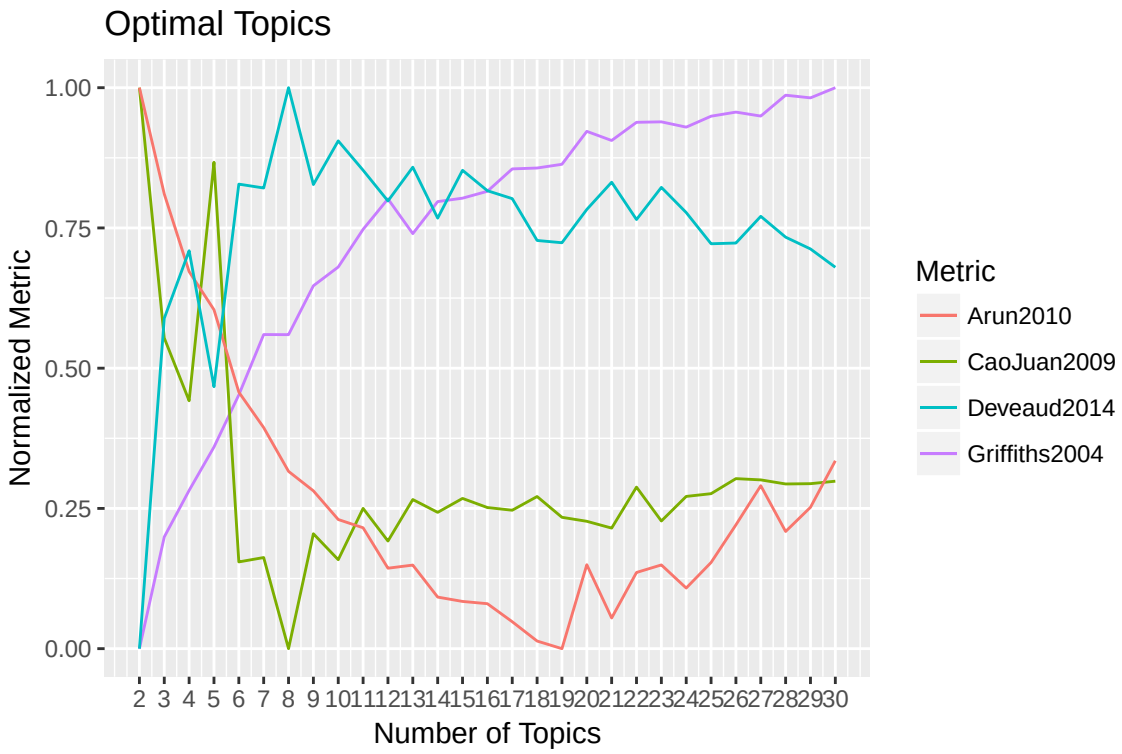

Figure 16. Topic Number Analysis

As these metrics have opposing goals, two of which require maximization and two minimization, a heuristic approach using subject matter expert knowledge would allow for determination of the optimal number of topics. However, assuming all four metrics are equally as informative, a more deductive approach provided a method to rank order the suggested number of topics for each PSC category. Topic numbers were ranked by each metric, with ranks summed across all four metrics, and sorted allowing for determination of the top number of topics for each category.

Table 11. Optimal Topics Analysis Results With Topic Ranking

topics	Griffiths2004	CaoJuan2009	Arun2010	Deveaud2014	Rank
8	0.5597292	0.0000000	0.3160586	1.0000000	1
10	0.6803777	0.1585956	0.2302237	0.9052150	2
6	0.4524238	0.1546385	0.4573301	0.8281174	3
21	0.9058831	0.2149058	0.0546839	0.8315242	4
7	0.5599217	0.1624469	0.3940984	0.8212288	5
9	0.6468854	0.2048380	0.2813023	0.8273240	5

From this it can be decided that a topic number of $k = 8$ would provide an optimal number of topics for use with LDA topic modeling. This supports that there are 8 underlying structures in the text data that can be leveraged for for the topic models. This parameter could be used in LDA to construct topic models for PSC Category 70. The new topics represented the restructured sub-categories which can be compared to the legacy categories regarding classification accuracy.

IV. Analysis And Results

4.1 Chapter Overview

In this chapter an assessment of analysis and a summary of results is examined. The baseline level classification accuracy is first discussed, constructing classification models for classifying documents to their corresponding level 2 categories based on their legacy category assignments as target values. Following this, topic modeling results are highlighted for construction of new modeled topics using the determined optimal number of underlying topics in the data. Using these new topics as optimal level 2 categories, the classification models are reassessed for their ability to classify to the new categories. Classification accuracy provided a metric for comparing model performance between the algorithms, while computational time contrasts the models' feasibility. Misclassifications are then assessed to determine culprit terms in misclassification. The process is then replicated to also evaluate a "re-modeling" of all of the level 1 IT categories.

4.2 Initial Accuracy Results

For the initial results,a subset of the IT data was used as a proof of concept of the methodology. This PSC category was designated `70` representing ADP software, support, and equipment. Test accuracy was used as the evaluation metric for each of the models, and noted the cross validation metric optimized, the value of that optimized metric, and the term frequency normalization of the document-term matrix, either term frequency (*tf*) or term frequency-inverse document frequency (*tf-idf*). Transforming the term frequency to a normalized *tf-idf* had little effect on the accuracy of the classification. Further, the Random

Forest technique provided the highest test accuracy with an accuracy of 69.4%. This model used term frequency as the observation value and necessitated 6 terms available to be split at each tree node.

Table 12. Classification with Crossvalidation Models, tf and tf-idf

Accuracy	Model	Metric	Value	Normalization
0.6762325	SVM	Cost	10	tf
0.6941378	RF	mtry	6	tf
0.6772136	K-NN	k	22	tf
0.3747854	SVM	Cost	10	tf-idf
0.5634045	RF	mtry	6	tf-idf
0.6811381	K-NN	k	22	tf-idf

4.3 Oversampling and SMOTE

This PSC required treatment to combat an extreme imbalance in classes, with the minority class representing less than 1 percent of the documents in the data. The use of oversampling and SMOTE were compared in regards to the test accuracy of each of the classification algorithms. SMOTE far outperformed the oversampling technique, despite having less total observation for training and testing. After balancing treatment, K-Nearest Neighbors with $k = 1$ marginally outperformed both SVM and Random Forest algorithms.

Table 13. Classification with Crossvalidation Models, Oversampling and SMOTE

Accuracy	Model	Metric	Value	Sampling
0.4281768	SVM	Cost	10	Oversampling
0.4603914	RF	mtry	7	Oversampling
0.6031308	K-NN	k	10	Oversampling
0.7255985	SVM	Cost	10	SMOTE
0.7440147	RF	mtry	14	SMOTE
0.7486188	K-NN	k	1	SMOTE

Although classification using the sampling treatment performed better than the unbalanced native data, it is possible that the increase is due to over-fitting of the model. As the number of replicated training data was increased for the minority class, the flexibility of the model to unseen data was decreased. Decreasing the number of documents represented from the largest class made classification of that class more generalized, but increased the potential for test variance, which could have limited the test accuracy. However, the classification accuracy was used as a baseline for which to compare the newly constructed topic model classification accuracy. If the topic modeling approach did not significantly aid in the classification feasibility, then the legacy categories would be more appropriate for the task.

4.4 Optimal Topics Analysis for Subcategories of PSC Category 70

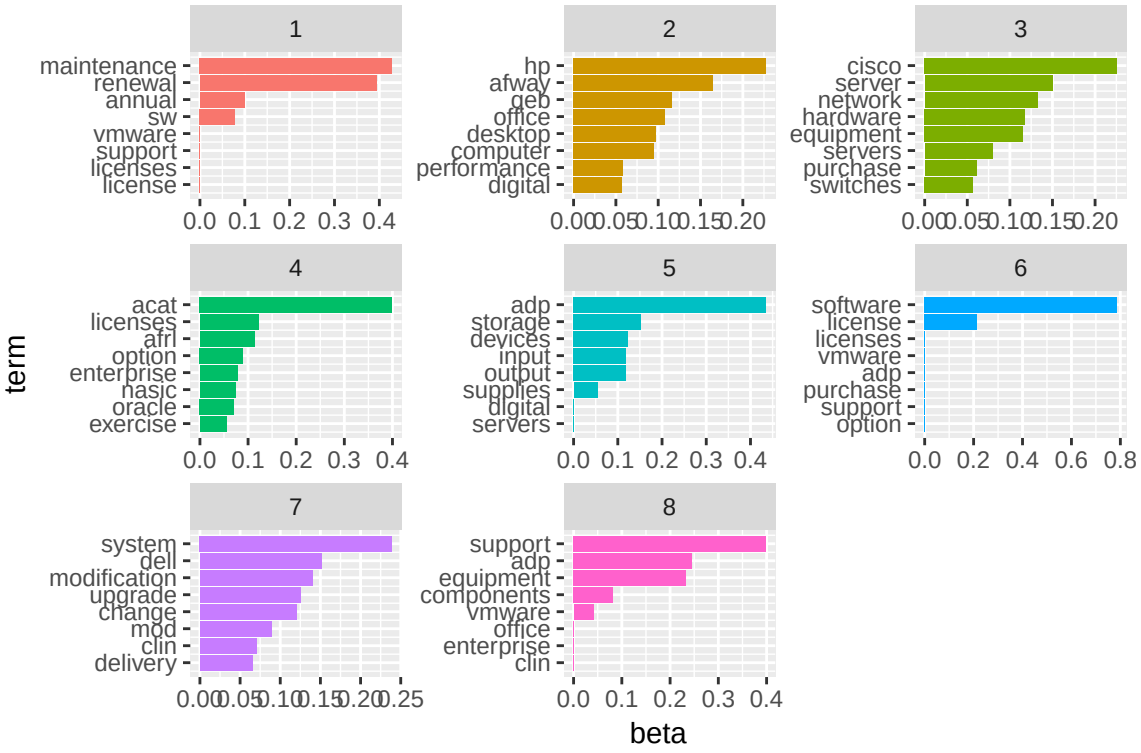

Figure 17. Per-Topic Per-Word Probabilities

Using the optimal number of topics of $k = 8$, the new categories were constructed using Latent Dirichlet Allocation. Extracting the per-topic per-word probabilities (β), the words which best represent each of their respective topics could be determined. Although each topic is a mixture of all of the words at varying probabilities, the most significantly associated words were used to best represent each of the topics. For example, topic 3 appeared to represent server hardware and other network equipment, while topic 6 appeared to be comprised of contracts pertaining exclusively to software and silencing. Many contracts relevant to Information Technology may have many terms that are shared between very distinct types of contracts. Consequently, some of the modeled topics appeared to have a shared term that significantly defined its respective topic or category. The term `adp` appeared as

a feature that is decidedly defining both topics 5 and 8. This may have affected the ability to distinguish these categories in classification, and therefore further investigation into the level of similarity between the topics was required in respect to the words that have a beta greater than .001, thus significantly associated with the topic. Using the log2 scale, it could be determined that the association of the term `adp` to topics 5 and 8 were similar in magnitude in respect to their per-topic probability. The terms `components` and `equipment` were also terms relatively common in both topics.

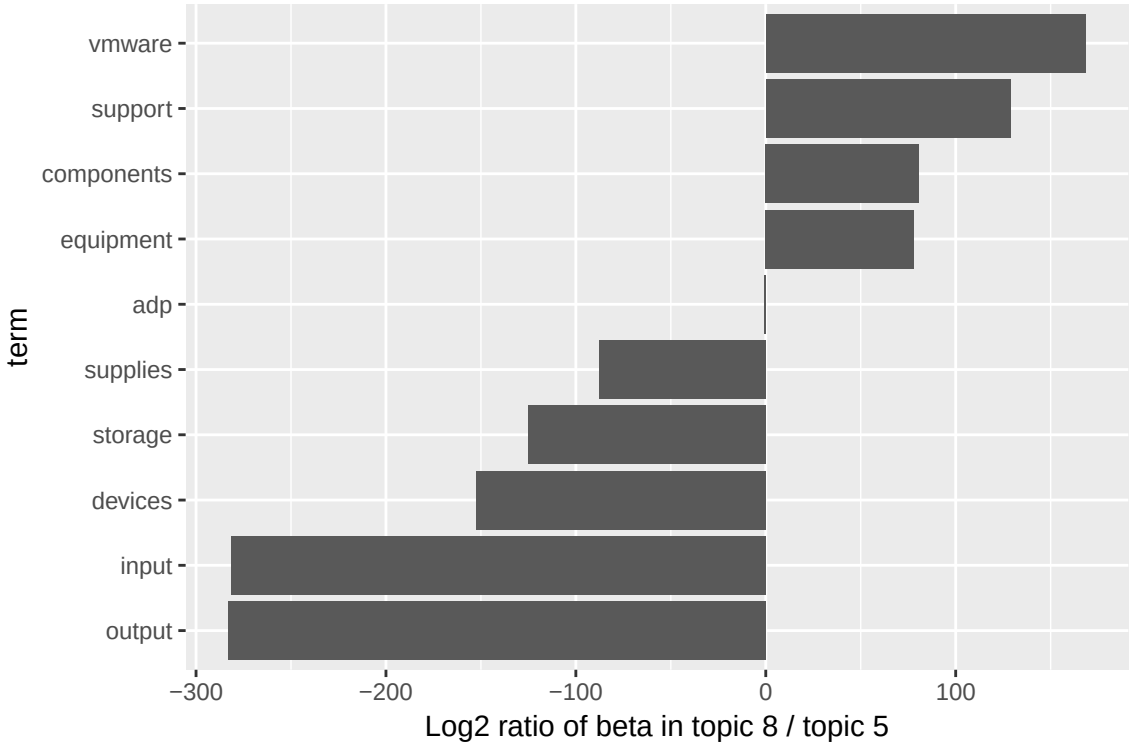

Figure 18. Pairwise Log2 Ratio of Beta Comparison

4.4.1 Accuracy of Contructed Category Classification

Table 14. Classification with Crossvalidation Models, tf and tf-idf, 8 Topics

Accuracy	Model	Metric	Value	Normalization
0.9744910	SVM	Cost	10	tf
0.9286240	RF	mtry	24	tf
0.9232279	K-NN	k	5	tf
0.2764287	SVM	Cost	10	tf-idf
0.9293598	RF	mtry	12	tf-idf
0.9298504	K-NN	k	8	tf-idf

Using the extracted topics from the LDA model with $k = 8$, it was found that allowing LDA to form topics for classification significantly improved the ability to classify the newly formed categories based on the the terms in their descriptions. The test accuracy increased to 97.4% using the Support Vector Machine classifier with a cost of 10 and term frequency.

Table 15. Accuracy Statistics of Classification Models

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue
RandomForest	0.9286240	0.9174944	0.9202872	0.9363400	0.1795438	0
SVM	0.9744910	0.9705202	0.9691754	0.9791109	0.1795438	0
KKNN	0.9232279	0.9113318	0.9146261	0.9312171	0.1795438	0
RandomForest.idf	0.9293598	0.9183805	0.9210602	0.9370375	0.1810155	0
SVM.idf	0.2764287	0.1235978	0.2627422	0.2904364	0.1810155	0
KKNN.idf	0.9298504	0.9189943	0.9215756	0.9375025	0.1810155	0

Exploring this measure of performance further, the difference in classification accuracy metrics was compared between the methods. In this case, support vector machine classification provided the best test accuracy for classification of contracts based on contract

description for the majority of the PSC categories. All classification techniques resulted in an accuracy above 90% except for SVM using *tf-idf* normalization. All models also resulted in statistically significant accuracy, with p-values less than 0.05. Therefore, there was enough data to establish any further categories using topic modeling with some statistical significance, which may not have been the case for PSC categories for which there was an unusable amount of data provided. Cohen's Kappa value compares the observed accuracy with that of random chance classification (or the expected accuracy). A higher Kappa represents a larger deviation of observed accuracy from expected accuracy, and thus could be used to evaluate classification models, and compare between classifiers for a given PSC Category. It was shown that SVM with *tf-idf* had a Kappa of 0.124, and therefore was almost indistinguishable from random classification. All other classification models resulted in classification accuracy better than that of accuracy attributed to differing class distribution. Although a Support Vector Machine model provided the most accurate model, the feasibility to implement this methodology to the end user was also considered. Investigating the time required to complete cross-validation, training, and testing, it was shown that Support Vector Machines required significantly more time. Random Forest required less than a minute, whereas Support Vector Machines took 6.22 minutes. The marginal performance increase in SVM over Random Forest classification may not have been worth the time required to build the higher performing models.

Table 16. Classification Computation Time

Time	Units	Model
6.223186	mins	SVM
48.945230	secs	RF
1.151569	mins	KNN

4.4.2 Evaluating Misclassification

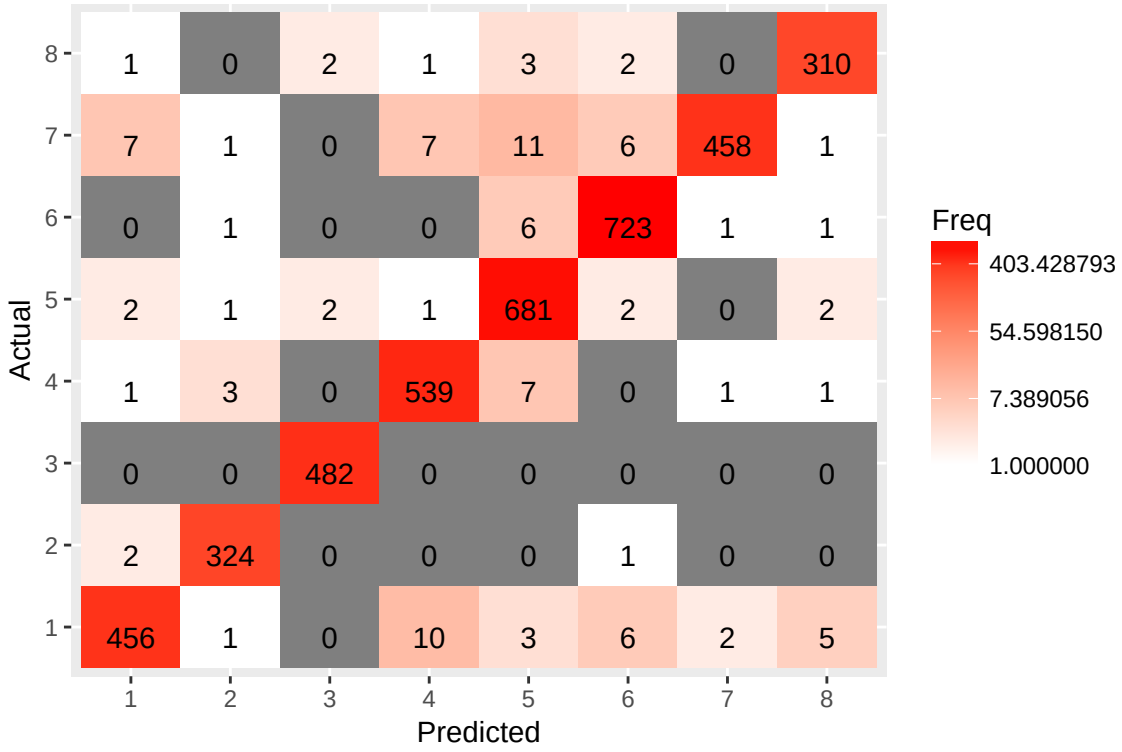

Figure 19. Confusion Matrix of PSC 70 Classifications with 8 Topics

Classification using SVM resulted in a clean confusion matrix diagonal, representing accurate classification of predicted categories to their actual category assignments. Predicting category 5, or the category consisting of contracts defined by the terms `adp`, `storage`, and `devices`, had the most misclassifications, with category 5 predictions misclassifying to category 7, which consisted of the terms `system`, `dell`, and `modification`, most frequently. These misclassifications were more closely analyzed by filtering the contracts that were misclassified, and evaluating the words that were most frequently used in misclassified contracts.

Table 17. Misclassified Contracts

document	psc_cat	psc	class	classification	Description
10646_70	70	7030	7	6	NON ACAT AFLCMC PZIT TECPLOT
544_70	70	7010	2	6	IDS MAINTENANCE LAPTOP DELL
1265_70	70	7010	7	4	NON ACAT AFLCMC XP OZ AFWAY
13532_70	70	7035	5	2	PN P WINDOWS SERVER DATAC
8499_70	70	7030	6	5	MATLAB MAINTENANCE RENEWAL MAT
2868_70	70	7021	4	2	CPU SERVER WORKSTATION
9998_70	70	7030	1	7	NON ACAT HPW MICROSTATION
1191_70	70	7010	7	4	NON ACAT AFLCMC PK HP MOBILE
1219_70	70	7010	7	4	NON ACAT AFLCMC WWO HP OFFIC
11748_70	70	7030	1	6	ADMIN CHANGE FOR COMPLETE TO P

The most frequently used words in misclassified documents were `license`, `software`, `acat`, `ada`, `application`, `change`, and `pro`. Frequently misclassified words could be used to better understand where use of terms should be avoided, or where certain terms could be encouraged to be used for contracts of specific categories. The term `acat` however, refers less to the contract and more to how large the contract requirement is. ACAT, or Acquisition Category represents the scope of the acquisition for which the contract was created. Terms like this which may not have contributed to the distinction of one contract's category over another, could be an example of words that can be expunged from the data for a cleaner representation of contract topics, or provide insight as to better description practices upon submittal.

Table 18. Most Frequently Misclassified Words

word	n
acat	16
software	12
maintenance	10
aflcmc	9
support	8
change	7
pk	7
server	7
correct	6
modification	6

4.5 Optimal Topics Analysis for IT Categories

This process was extended to modeling the general IT contract categories. First, the potential for an optimal number of underlying structures was evaluated. Implementing the same method as for identifying subcategories, the potential underlying structures were examined, again finding the points of maximum and minimum for the structure similarity and distance metrics.

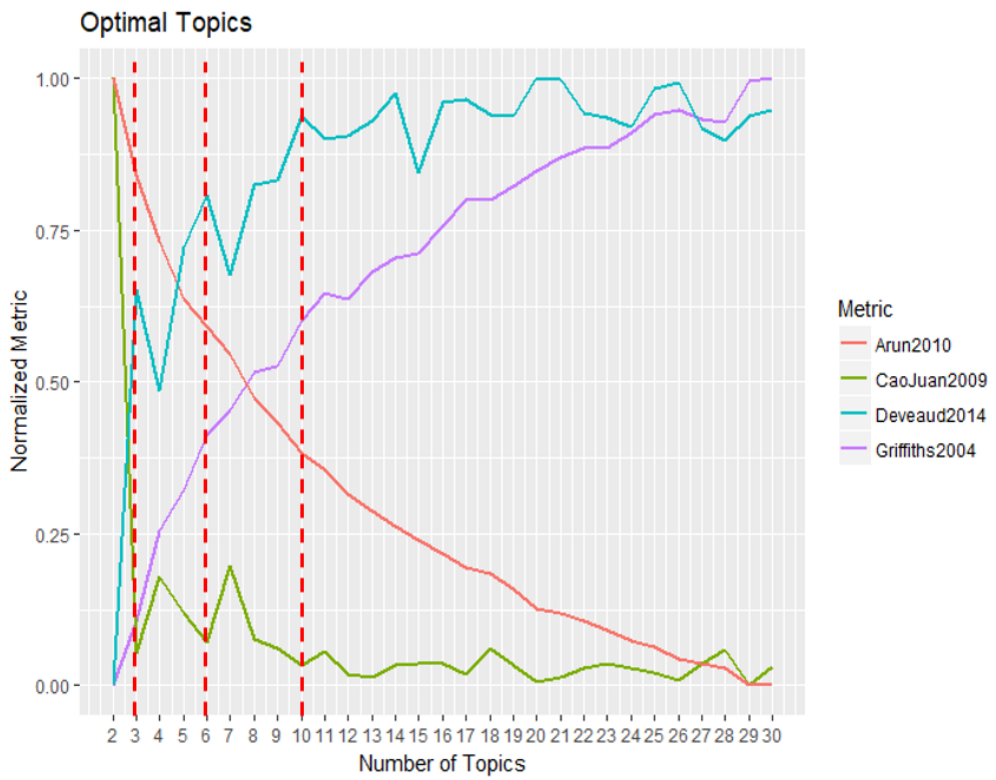

Figure 20. Topics Optimality Analysis on PSC Categories

In Figure 20, it can be determined that there were three local minima and maxima, presenting several options for evaluating topics for a certain number of K. These potential points of optimality were at K value of 3, 6 and 10. Griffiths and Arun metrics proved uninformative again in this case when investigating topics 2 to 30.

Table 19. Classification with Crossvalidation Models, tf, 10/6/3 Topics

Accuracy	Model	Metric	Value	Normalization	Topics	Time	Units
0.7601597	RF	mtry	16	tf	10	8.348244	mins
0.7441267	K-NN	k	14	tf	10	7.254917	mins
0.9212587	RF	mtry	16	tf	6	6.564267	mins
0.9126967	K-NN	k	10	tf	6	5.672580	mins
0.9206733	RF	mtry	16	tf	3	6.556303	mins
0.9027442	K-NN	k	10	tf	3	5.608623	mins

Determining classification accuracy of each candidate for topic number in Table 19, it can be seen that classification accuracy increased significantly from 10 modeled topics to 6. Further, the 6-topic model provided improved classification accuracy over the simpler 3-topic model, supporting that the 6-topic model would be best suited for the subsequent text classification objective. Although intuitively topic models with fewer topics should have been more accurately classified, it is likely that with so few topics for which to assign contracts, similar contracts were forced into different topics, thus increasing the classification error.

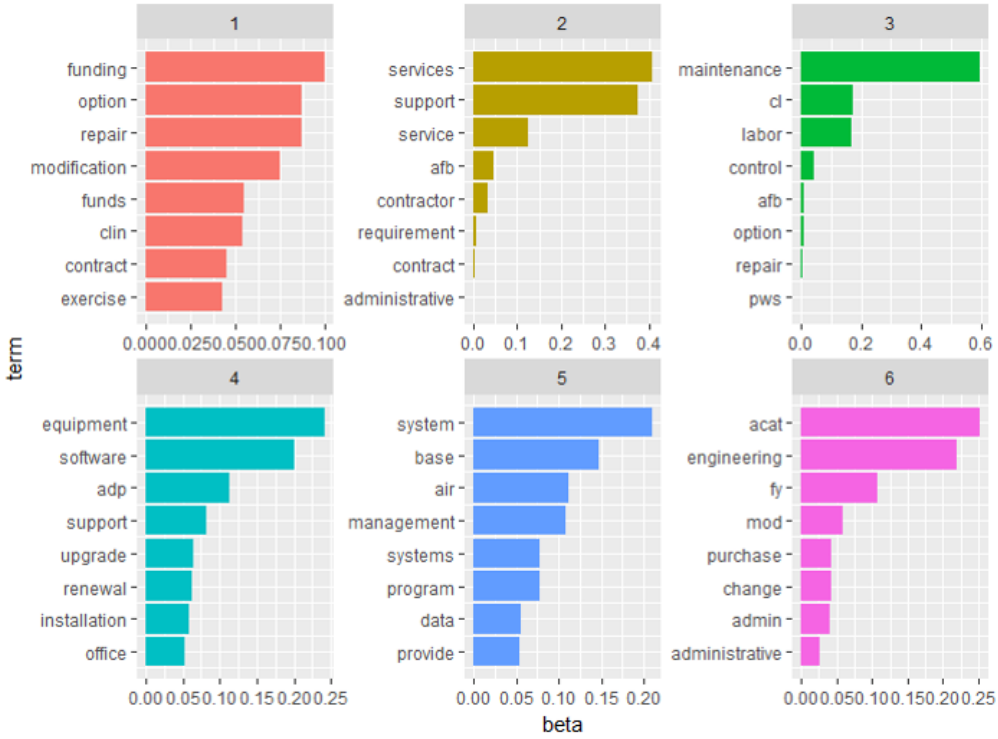

Figure 21. 6-topic Model on PSC Categories

The most associated terms with each topic could be assessed in the 6-topic model, and provide insight as to the composition of each of the newly formed categories. It can be deduced that these topics differentiate by "funding, and repair", "services and support", "maintenance", "equipment and software", "systems", and "acat and engineering" contracts, as evidenced in Figure 21. It is important to note, however, that the term `acat` was a reduced designation for Acquisition Category (ACAT) level projects. These designations

were ACAT I, II, and III, which during the pre-processing stage had been reduced to the same term. Therefore, the prominence of this term in contract descriptions could have affected the similarities between contracts that are otherwise dissimilar.

V. Conclusion and Future Research

Text analysis is a growing field in data science. Data collected or archived by the DoD provides a wealth of potential for analysis on text data, providing insights to decision makers and analysts that would otherwise go undiscovered. Leveraging text analysis tools to extract meaning and themes from text data can facilitate improved efficiency in data organization and information retrieval. The Air Force Installation Contracting Agency accumulates text data in the form of installation contract descriptions, but as this collection grows, unstructured data, specifically text fields, become increasingly unwieldy. Text analysis could create a framework of structure and organization to otherwise unused data.

A methodology is proposed for implementing topic modeling methods to construct new sub categories from the existing Product Service Code categories, providing organizational insights for potential category managers to shape installation support spend. Using Latent Dirichlet Allocation, contracts are clustered into categories based on the similarities of the terms included the description of the the contract requirement. Coupled with machine learning classification, with utilization of Support Vector Machines, Random Forests, and Weighted K-Nearest Neighbors, this methodology provides a procedural flow for model cross-validation, training, and validation. The research supports that topics can be more efficiently modeled, in respect to document classification accuracy, by determining the optimal number of topics for which to cluster the documents. As a result, more efficient contract classification could allow for potential DoD savings by limiting the number of contracts that are misclassified into a spend category unfit for their requirement.

This research did not come without limitations. Scarcity of data for many of PSC categories limited the analysis to the largest of the categories, PSC category 70. Regardless of sample limitations, the methodology can only improve with an increased number of terms, in

the form of more descriptive and lengthy descriptions, lacking in the data provided. As topic modeling and text classification depend an accurate depiction of the data by way for the corpus, including more terms to describe each of the contract could expand the potential for improved text analysis. In addition, computational power and the necessity for a practical approach to text analysis with interpretable results disqualified some more strenuous model building techniques, such as neural nets.

Future research could see more advanced text classification techniques be implemented, better accustomed to short text data. For example, use of recurrent neural networks, character-level convolutional networks, or gradient boosting machines, could allow for higher fidelity text classification solutions for Air Force contracts. Alternatively, research into the parameter selection of test splits, document term matrix sparsity, and LDA hyperparameters could tune and improve the suggested approach. Evaluation of the current level 1 categories could allow for an improved shaping of higher-level spend categories, allowing analysts to identify and correct level 1 misclassifications. More generally, a similar methodology could be researched for all Air Force or DoD contracts, broadening the scope for spend category shaping.

VI. Appendix

6.1 Packages

```
#packages used
library(psych)
library(data.table)
library(knitr)
library(readr)
library(tidyverse)
library(tidytext)
library(topicmodels)
library(stringr)
library(ggplot2)
library(scales)
library(magrittr)
library(class)
library(tm)
library(randomForest)
library(MASS)
library(e1071)
library(forcats)
library(kableExtra)
library(ldatuning)
library(caret)
library(gridExtra)
library(htmlTable)
library(xtable)
library(ROSE)
library(nnet)
library(kknn)
```

6.2 Functions

```
# numbered psc as classes
class_seq <- function(data) {

  data <- data %>%
    group_by(psc_cat) %>%
    mutate(class = as.numeric(factor(psc)))

  data <- data %>%
```

```
    subset(!(psc_cat == "74"|psc_cat == "L"|psc_cat == "K"|psc_cat == "S"))
  return(data)

}

#normalize optimal output
normalize_metrics <- function(values) {
  # normalize to [0,1]
  columns <- base::subset(values, select = 2:ncol(values))
  values <- base::data.frame(
    values["topics"],
    base::apply(columns, 2, function(column) {
      scales::rescale(column, to = c(0, 1), from = range(column))
    })
  )
  return(values) }

#optimal topic number
optimal_topics <- function(dtm){
result <- FindTopicsNumber(
  dtm,
  topics = seq(from = 2, to = 30, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009",
              "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 1234),
  mc.cores = 4L,
  #make sure this is appropriate number of cores you wish to use
  verbose = TRUE
)

return(result)
}

#construct DTM
tidyDTM <- function(text.df, sparse){
  sw <- add_row(stop_words,
                word = c("igf","ot", "ct"),
                lexicon = c("SMART", "SMART", "SMART"))

  #word counts
  word_counts <- text.df %>%
    unnest_tokens(word, description) %>%
    anti_join(sw) %>%
    count(document, word, sort = TRUE) %>%
    ungroup()

  #cast dtm
  dtm <- word_counts %>%
    cast_dtm(document, word, n)

  dtm$dimnames$Terms <- gsub("function",
                             "functio",
                             dtm$dimnames$Terms)
```

```
  dtmNoSparse <- removeSparseTerms(dtm, sparse)

  return(dtmNoSparse)
}

explDTM <- function(text.df) {

  sw <- add_row(stop_words,
                word = c("igf","ot", "ct"),
                lexicon = c("SMART", "SMART", "SMART"))

  #word counts
  word_counts <- text.df %>%
    unite(document, psc, document) %>%
    unnest_tokens(word, description) %>%
    anti_join(sw) %>%
    count(document, word, sort = TRUE) %>%
    ungroup()

  #cast dtm
  dtm <- word_counts %>%
    cast_dtm(document, word, n)

  dtm$dimnames$Terms <- gsub("function",
                             "functio",
                             dtm$dimnames$Terms)

  return(dtm)

}

tidyDTMidf <- function(text.df, sparse){
  sw <- add_row(stop_words,
                word = c("igf","ot", "ct"),
                lexicon = c("SMART", "SMART", "SMART"))

  #word counts
  word_counts <- text.df %>%
    unnest_tokens(word, description) %>%
    anti_join(sw) %>%
    count(document, word, sort = TRUE) %>%
    ungroup()

  word_counts <- word_counts %>%
    bind_tf_idf(document, word, n)

  #cast dtm
  dtm <- word_counts %>%
    cast_dtm(document, word, tf_idf)

  dtm$dimnames$Terms <- gsub("function",
                             "functio",
                             dtm$dimnames$Terms)

  dtmNoSparse <- removeSparseTerms(dtm, sparse)
```

```
  return(dtmNoSparse)
}

removeDuplicates <- function(dtm, data){
  #remove rows not in dtm
  duplicates <- data[duplicated(data$document),]
  data <- data[!(duplicated(data$document)),]
  data <- data[(data$document %in% dtm$dimnames$Docs),]
  return(data)
}



dataMatrix <- function(dtm, data) {
  dtm.mat <- as.data.frame(as.matrix(dtm))
  dtm.mat$targetCat <- as.factor(data$class[match(rownames(dtm.mat),
                                                  data$document)])
  #dtm.mat$targetCat <- as.factor(data$class)
  return(dtm.mat)
}

trainSplit <- function(dtm.mat) {
  p=0.7
  #holdout
  train.idx <- sample(nrow(dtm.mat),ceiling(nrow(dtm.mat) * p))
  return(train.idx)}
testSplit <- function(dtm.mat, train.idx){
  test.idx <- (1:nrow(dtm.mat))[-train.idx]
  return(test.idx)
}
dtmCat <- function(dtm.mat){
  #targets
  dtm.cat <- dtm.mat[,"targetCat"]
  return(dtm.cat)
}

dtmMatNl <- function(dtm.mat){
  dtm.mat.nl <- dtm.mat[, !colnames(dtm.mat) %in% "targetCat"]
  return(dtm.mat.nl)
}



doKNNCV <- function(dtm.mat.nl, dtm.cat, train.idx, test.idx){
  t1 <- Sys.time()
  knn.cross <- tune.knn(x = dtm.mat.nl,
                        y = dtm.cat,
                        k = 1:20,
                        l = 0,
                        tunecontrol=tune.control(sampling = "cross"),
                        cross=10)
  k <- as.numeric(knn.cross$best.parameters[1,])
  pred.model <- knn(dtm.mat.nl[train.idx,],
                    dtm.mat.nl[test.idx,],
                    dtm.cat[train.idx],
                    k = k,
                    use.all = TRUE)
```

```
  Test_Obs <- dtm.cat[test.idx]
  Predicted <- pred.model

  conf <- table(Predicted, Test_Obs)

  f.conf <- confusionMatrix(conf)
  #stats <- f.conf$overall
  time <- Sys.time() - t1
  stats <- list(f.conf$overall,
                knn.cross,
                f.conf,
                pred.model,
                time)
  return(stats)
}


doKKNN <- function(dtm.mat,train.idx, test.idx){
  t1 <- Sys.time()
  train.kknn <- train.kknn(targetCat~.,
                           dtm.mat,
                           kmax = 25,
                           kernel = c("rectangular",
                                      "triangular",
                                      "epanechnikov",
                                      "gaussian",
                                      "rank",
                                      "optimal"))
  k <- as.numeric(train.kknn$best.parameters$k)
  kernel <- train.kknn$best.parameters$kernel
  pred.model <- kknn(targetCat~.,
                     dtm.mat[train.idx,],
                     dtm.mat[test.idx,],
                     k = k,
                     kernel = kernel)

  Test_Obs <- dtm.mat$targetCat[test.idx]
  Predicted <- pred.model$fitted.values

  conf <- table(Predicted, Test_Obs)

  f.conf <- confusionMatrix(conf)
  #stats <- f.conf$overall
  time <- Sys.time() - t1
  stats <- list(f.conf$overall,
                pred.model,
                f.conf,
                k,
                kernel,
                time)
  return(stats)
}

doKKNNprev <- function(dtm.mat,dtm.cat, train.idx, test.idx, k){
  knn.pred <- kknn(targetCat~.,
```

```
                    dtm.mat[train.idx,],
                    dtm.mat[test.idx, ],
                    k = k)
  conf.mat <- table("Predictions" = knn.pred$fitted.values,
                    Actual = dtm.cat[test.idx])
  accuracy <- sum(diag(conf.mat)/length(test.idx) *100)
  return(accuracy)
}

doRFtune <- function(dtm.mat.nl, dtm.cat) {
  tune.rf <- tuneRF(dtm.mat.nl,
                    dtm.cat,
                    doBest = TRUE,
                    trace = FALSE,
                    plot = FALSE)
  return(tune.rf$mtry)
}

doRF <- function (dtm.mat, train.idx, test.idx, n, m) {
  t1 <- Sys.time()

  model <- randomForest(targetCat~.,
                        data = dtm.mat,
                        subset = train.idx,
                        ntree = n,
                        mtry = m,
                        importance = TRUE)
  pred.model <- predict(model, dtm.mat[test.idx,])

  Test_Obs <- dtm.mat[test.idx,]$targetCat
  Predicted <- pred.model

  conf <- table(Predicted, Test_Obs)

  f.conf <- confusionMatrix(conf)
  #stats <- f.conf$overall
  time <- Sys.time() - t1
  stats <- list(f.conf$overall,
                model,
                f.conf,
                pred.model,
                time)
  return(stats)
}

AccStats <- function(model, dtm.mat, test.idx){
  pred.model <- predict(model, dtm.mat[test.idx,])

  Test_Obs <- dtm.mat[test.idx,]$targetCat
  Predicted <- pred.model

  conf <- table(Predicted, Test_Obs)

  f.conf <- confusionMatrix(conf)
```

```
  return(f.conf$overall)
}

doRFerr <- function (dtm, dtm.mat, train.idx, n) {
  tree.fit <- randomForest(targetCat~.,
                           data = dtm.mat,
                           subset = train.idx,
                           ntree = n,
                           importance = TRUE)
  error <- as.data.frame(tree.fit$err.rate)
  return(mean(error))
}

doSVM <- function (dtm.mat, cost, gamma, kernel) {
  svm.fit <- svm(targetCat~.,
                 dtm.mat,
                 kernel = kernel,
                 cost = 10,
                 gamma = 1)
  return(svm.fit)
}

doSVMerr <- function(dtm.mat, test.idx){
  t1 <- Sys.time()
  tune.out <- tune(svm, targetCat~.,
                   data = dtm.mat,
                   kernel = "linear",
                  ranges = list(cost = c(0.0001, 0.001, 0.01, 0.1, 1, 10),
                                scale = FALSE))
  model <- tune.out$best.model
  #SVMerror <- tune.out$best.performance
  #return(SVMerror)
  pred.model <- predict(model, dtm.mat[test.idx,])

  Test_Obs <- dtm.mat[test.idx,]$targetCat
  Predicted <- pred.model

  conf <- table(Predicted, Test_Obs)

  f.conf <- confusionMatrix(conf)
  time <- Sys.time() - t1
  stats <- list(f.conf$overall,
                model,
                f.conf,
                pred.model,
                time)
  return(stats)
}


rfplot.error <-function(randomForest.fit) {
  # Get OOB data from plot and coerce to data.table
  oobData <- as.data.table(randomForest.fit$err.rate)

  # Define trees as 1:ntree
```

```
  oobData[, trees := .I]

  # Cast to long format
  oobData2 <- melt(oobData, id.vars = "trees")
  setnames(oobData2, "value", "error")

  # Plot using ggplot
  plt <- ggplot(data = oobData2,
                aes(x = trees,
                    y = error,
                    color = variable)) +
    geom_line()

  return(plt)
}

rfplot.importance <- function(randomForest.fit){
  data_frame(var = rownames(importance(randomForest.fit)),
             MeanDecreaseGini = importance(randomForest.fit)[,1]) %>%
    top_n(20, MeanDecreaseGini) %>%
    mutate(var = fct_reorder(var,MeanDecreaseGini, fun = median)) %>%
    ggplot(aes(var, MeanDecreaseGini)) +
    geom_point() +
    coord_flip() +
    labs(title = "Gini Importance of Variable Terms",
         subtitle = "Bagging",
         x= NULL,
         y = "Average decrease in the Gini Index")
}

ldafun <- function(dtm, k) {
  lda <- LDA(dtm, k, control = list(seed = 1234))
  return(lda)
}

topic_terms <- function(topics.beta) {
  top_terms <- topics.beta %>%
    group_by(topic) %>%
    top_n(10, beta) %>%
    ungroup() %>%
    arrange(topic, -beta)

  plt <- top_terms %>%
    mutate(term = reorder(term, beta)) %>%
    ggplot(aes(term, beta, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free") +
    coord_flip()
  return(plt)
  }

tidybeta <- function(lda){

  #extract topic betas
  topics <- tidy(lda, matrix = "beta")
```

```
  return(topics)
}

tidygamma <- function(lda){

  #extract topic betas
  topics <- tidy(lda, matrix = "gamma") %>%
    separate(document,
             c("psc_cat", "psc"),
             sep = "_",
             convert = TRUE)

  return(topics)
}

gammaPlots <- function(gamma) {
  plt <- gamma %>%
    mutate(psc_cat = reorder(psc_cat, gamma * topic)) %>%
    ggplot(aes(factor(topic), gamma)) +
    geom_boxplot() +
    facet_wrap(~ psc_cat)

  return(plt)
}


LDAclassify <- function(gamma){

  #classification

  classifications <- gamma %>%
    group_by(psc_cat, psc) %>%
    top_n(1,gamma) %>%
    ungroup()
  return(classifications)
}

LDAtopics <- function(classifications) {
  topics <- classifications %>%
    count(psc_cat, topic) %>%
    group_by(psc_cat) %>%
    top_n(1,n) %>%
    ungroup() %>%
    transmute(consensus = psc_cat, topic)
  return(topics)
}

misclass <- function(classifcations) {

  class <- classifications %>%
    inner_join(topic, by = "topic") %>%
    filter(psc_cat != consensus)
  return(class)
}
```

```
LDAconfusion <- function(lda, dtm) {
  #confusion matrix

  assignments <- augment(lda, data = dtm)

  assignments <- assignments %>%
    separate(document, c("psc_cat", "psc"),
             sep = "_", convert = TRUE) %>%
    inner_join(topics, by = c(".topic" = "topic"))

  plt <- assignments %>%
    count(psc_cat, consensus, wt = count) %>%
    group_by(psc_cat) %>%
    mutate(percent = n / sum(n)) %>%
    ggplot(aes(consensus, psc_cat, fill = percent)) +
    geom_tile() +
    scale_fill_gradient2(high = "red", label = percent_format()) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1),
          panel.grid = element_blank()) +
    labs(x = "psc_cat words were assigned to",
         y = "psc_cat words came from",
         fill = "% of assignments")
  return(plt)
}
```

6.3 Methodology Code

```
#import clean
#import data
ITdata <- fread("IT_FPDSNG.csv")
FullData <-fread("AF_FPDS.csv")

ITdata <- data_full

ITdata <- ITdata %>%
  dplyr::select(unique_transaction_id,
                psc_cat,
                productorservicecode,
                descriptionofcontractrequirement)

ITdata <- ITdata %>%
  separate(productorservicecode,
           into = c('psc', "psc_desc"), sep = 4)

#clean
ITdata$descriptionofcontractrequirement <- gsub('[[:digit:]]+', ' ', ITdata$descriptionofcontractrequirement)
ITdata$descriptionofcontractrequirement <- gsub('[^[:alnum:]]', ' ', ITdata$descriptionofcontractrequirement)

#ITdata$descriptionofcontractrequirement %<>%
 # gsub('[[:digit:]]+', '',.) %>%
```

```
  #gsub("igf", "", ., ignore.case = TRUE) %>%
  #gsub("ot", "", ., ignore.case = TRUE) %>%
  #gsub("ct","", ., ignore.case = TRUE)
ITdata <- ITdata[!(is.na(ITdata$descriptionofcontractrequirement))]
#ITdata <- ITdata[!(is.na(ITdata$dollarsobligated))]
ITdata <- ITdata[!(is.na(ITdata$psc_cat))]
ITdata <- ITdata[!(is.na(ITdata$productorservicecode))]
ITdata <- ITdata[!(is.na(ITdata$psc))]

#id observations by contract ID and transaction ID
ITdata <- ITdata %>%
  separate(unique_transaction_id,
           into = c('contract_id', 'transaction_id'),
           sep = 13)

#create tibble
textframe <- tibble( document = ITdata$contract_id,
                     psc_cat = ITdata$psc_cat,
                     psc = ITdata$psc,
                     description = as.character(ITdata$descriptionofcontractrequirement))

#collapse by contract
textframe <- aggregate(description ~ document + psc_cat + psc,
                       data = textframe, paste, collapse = " ")

#save data

saveRDS(textframe, file = "AF_FPDS_clean.RDS")

FPDS <- textframe

data_total <- rbind(textframe, data_raw)

data_total <- aggregate(description ~ document + psc_cat + psc,
                        data = textframe, paste, collapse = " ")

saveRDS(data_total, "data_total.RDS")

#investigate duplicates
duplicates <- data[duplicated(data$document),]

duplicates %>%
  group_by(document) %>%
  count()

duplicates %>%
  subset(document == "FA252114FG001") %>%
  View()

data <- data %>%
  mutate(ID = as.factor(row_number())) %>%
  dplyr::select(-document) %>%
  rename("document" = ID)

data$document <- paste(data$document, "58", sep="_")
```

```r
# Sampling

dtm.mat.imb <- dtm.mat
dtm.mat.imb$SMOTECOL <- NA
dtm.mat.imb$SMOTECOL <- as.factor(ifelse(dtm.mat.imb$targetCat == 2, "1" , "0"))
dtm.mat.bal.both <- ovun.sample(SMOTECOL ~ ., dtm.mat.imb, method = "both")
dtm.mat.bal.over <- ovun.sample(SMOTECOL ~ ., dtm.mat.imb, method = "over")

dtm.mat.bal.OVER <- subset(dtm.mat.bal.over$data, select = -SMOTECOL)
dtm.mat.bal.SMOTE <- subset(dtm.mat.bal.both$data, select = -SMOTECOL)

train.idx.bal.OVER <- trainSplit(dtm.mat.bal.OVER)

test.idx.bal.OVER <- testSplit(dtm.mat, train.idx.bal.OVER)

dtm.cat.OVER <- dtmCat(dtm.mat.bal.OVER)

dtm.mat.nl.bal.OVER <- dtmMatNl(dtm.mat.bal.OVER)

train.idx.bal.SMOTE <- trainSplit(dtm.mat.bal.SMOTE)

test.idx.bal.SMOTE <- testSplit(dtm.mat, train.idx.bal.SMOTE)

dtm.cat.SMOTE <- dtmCat(dtm.mat.bal.SMOTE)

dtm.mat.nl.bal.SMOTE <- dtmMatNl(dtm.mat.bal.SMOTE)
# Optimal Topics
sparse <- 0.98

dtm <- tidyDTM(data, sparse)

rowTotals <- apply(dtm , 1, sum) #Find the sum of words in each Document

dtm <- dtm[rowTotals> 0, ]   #remove all docs without words

lda.J <- ldafun(dtm, 3)

beta <- tidybeta(lda.J)

topics.terms <- topic_terms(beta)

gamma <- tidy(lda.J, matrix = "gamma")

classifications <- gamma %>%
  group_by(document) %>%
  top_n(1, gamma) %>%
  ungroup() %>%
  dplyr::select(-gamma) %>%
  as.data.frame()

data.optimal <- merge(data, classifications, by = "document")

data.optimal <- data.optimal %>%
  dplyr::select(-class) %>%
  rename("class" = topic)
```

```
prop.table(table(data.optimal$class))

# Optimal Topics
sparse <- 0.983

dtm <- tidyDTM(data, sparse)

rowTotals <- apply(dtm , 1, sum) #Find the sum of words in each Document

dtm <- dtm[rowTotals> 0, ]   #remove all docs without words


optimal_topics <- FindTopicsNumber(
  dtm,
  topics = seq(from = 2, to = 10, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009","Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 1234),
  mc.cores = 4L, #make sure this is appropriate number of cores you wish to use
  verbose = TRUE
)

optimal_topics.idf <- optimal_topics(dtm.idf)

saveRDS(optimal_topics, "optimal_topics_J_10.RDS")

optimal_topics_norm <- normalize_metrics(optimal_topics)

topics_grid <- lapply(optimal_topics_norm, topics_grid)

topics_collapsed <- ldply(optimal_topics_norm, data.frame)

names(topics_collapsed)[names(topics_collapsed) == '.id'] <- 'PSC_cat'

saveRDS(topics_collapsed, "topics_collapsed_70_30.RDS")

optimal_topics_70 <- ggplot(optimal_topics_norm, aes(x=topics)) +
  labs(title="Optimal Topics", x="Number of Topics", y="Normalized Metric", colour = "Metric") +
  geom_line(aes(y = Griffiths2004, color = "Griffiths2004")) +
  geom_line(aes(y = CaoJuan2009, color = "CaoJuan2009")) +
  geom_line(aes(y = Arun2010  , color = "Arun2010")) +
  geom_line(aes(y = Deveaud2014, color = "Deveaud2014")) +
  scale_x_continuous(
    breaks = topics_collapsed$topics[seq(1, length(optimal_topics$topics),by = 1)])

optimal_topics_70

topics_collapsed_ranked <- topics_collapsed

topics_collapsed_ranked <- optimal_topics_norm %>%
  mutate("Griffiths" = rank(-Griffiths2004),
         "Cao" = rank(CaoJuan2009),
         "Arun" = rank(Arun2010),
         "Deveaud" = rank(-Deveaud2014)) %>%
  mutate(Sum = rowSums(select_(.,"Griffiths", "Cao","Arun", "Deveaud"))) %>%
```

```
  mutate(Rank = as.integer(rank(Sum)))

optimal_topics_ranked <- topics_collapsed_ranked %>%
  group_by(PSC_cat) %>%
  top_n(3,-Rank) %>%
  dplyr::select(PSC_cat, topics, Rank)


topics_ranked$Griffiths = unlist(with(topics_collapsed_ranked,
                                      tapply(Griffiths2004,
                                             PSC_cat,rank)))
topics_ranked$Cao = unlist(with(topics_collapsed_ranked,
                                tapply(-CaoJuan2009,
                                       PSC_cat,rank)))
topics_ranked$Arun = unlist(with(topics_collapsed_ranked,
                                 tapply(-Arun2010,
                                        PSC_cat,rank)))
topics_ranked$Deveaud = unlist(with(topics_collapsed_ranked,
                                    tapply(Deveaud2014,
                                           PSC_cat,rank)))

topics_collapsed_ranked$Sum =
  rowSums(topics_collapsed_ranked[,c("Griffiths", "Cao","Arun", "Deveaud")])
topics_collapsed_ranked$Rank = unlist(with(topics_collapsed_ranked,
                                           tapply(-Sum,PSC_cat,rank)))

#classification optimal
data <- data.optimal

sparse <- 0.978

dtm <- tidyDTM(data, sparse)

dtm


#train test split

dtm.mat <- dataMatrix(dtm, data)


train.idx <- trainSplit(dtm.mat)

test.idx <- testSplit(dtm.mat, train.idx)

dtm.cat <- dtmCat(dtm.mat)

dtm.mat.nl <- dtmMatNl(dtm.mat)


AccStats.SVM <- doSVMerr(dtm.mat, test.idx)
AccStats.SVM.Accuracy <- t(data.frame(AccStats.SVM[1]))
AccStats.SVM.Accuracy <-
  AccStats.SVM.Accuracy[,colnames(AccStats.SVM.Accuracy) != "McnemarPValue"]
```

```
mtry.list <- doRFtune(dtm.mat.nl, dtm.cat)
AccStats.RF <- doRF(dtm.mat, train.idx, test.idx, n = 500, mtry.list)
AccStats.RF.Accuracy <- t(data.frame(AccStats.RF[1]))
AccStats.RF.Accuracy <-
  AccStats.RF.Accuracy[, colnames(AccStats.RF.Accuracy) != "McnemarPValue"]

AccStats.KNN <- doKNNCV(dtm.mat.nl, dtm.cat, train.idx, test.idx)
AccStats.KNN.Accuracy <- t(data.frame(AccStats.KNN[1]))
AccStats.KNN.Accuracy <-
  AccStats.KNN.Accuracy[, colnames(AccStats.KNN.Accuracy) != "McnemarPValue"]

saveRDS(AccStats.SVM, "AccStats_SVM_total_optimal.RDS")
saveRDS(AccStats.RF, "AccStats_RF_total_optimal.RDS")
saveRDS(AccStats.KNN, "AccStats_KNN_total_optimal.RDS")
#all IT data

data <- readRDS("IT_total.RDS")

sparse <- 0.99

dtm <- tidyDTM(data, sparse)
rowTotals <- apply(dtm , 1, sum)
dtm <- dtm[rowTotals> 0, ]

optimal_topics <- FindTopicsNumber(
  dtm,
  topics = seq(from = 2, to = 30, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009","Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 1234),
  mc.cores = 4L,
  verbose = TRUE
)

saveRDS(optimal_topics, "total_optimal_30.RDS")

optimal_topics_norm <- normalize_metrics(optimal_topics)

topics_grid <- lapply(optimal_topics_norm, topics_grid)

topics_collapsed <- ldply(optimal_topics_norm, data.frame)

names(topics_collapsed)[names(topics_collapsed) == '.id'] <- 'PSC_cat'

saveRDS(topics_collapsed, "topics_collapsed_30.RDS")

optimal_topics_total <- ggplot(optimal_topics_norm, aes(x=topics)) +
  labs(title="Optimal Topics",
       x="Number of Topics",
       y="Normalized Metric",
       colour = "Metric") +
  geom_line(aes(y = Griffiths2004,
                color = "Griffiths2004"),
            size = 1) +
  geom_line(aes(y = CaoJuan2009,
```

```
                color = "CaoJuan2009"),
            size = 1) +
  geom_line(aes(y = Arun2010  ,
                color = "Arun2010"),
            size = 1) +
  geom_line(aes(y = Deveaud2014,
                color = "Deveaud2014"),
            size = 1) +
  scale_x_continuous(
    breaks = optimal_topics_norm$topics[seq(1,
                                            length(optimal_topics_norm$topics),
                                            by = 1)])

optimal_topics_total

topics_collapsed_ranked <- topics_collapsed

topics_collapsed_ranked <- optimal_topics_norm %>%
  mutate("Griffiths" = rank(-Griffiths2004),
         "Cao" = rank(CaoJuan2009),
         "Arun" = rank(Arun2010),
         "Deveaud" = rank(-Deveaud2014)) %>%
  mutate(Sum = rowSums(select_(.,"Griffiths", "Cao","Arun", "Deveaud"))) %>%
  mutate(Rank = as.integer(rank(Sum)))

optimal_topics_ranked <- topics_collapsed_ranked %>%
  group_by(PSC_cat) %>%
  top_n(3,-Rank) %>%
  dplyr::select(PSC_cat, topics, Rank)

lda <- readRDS("lda_total_6.RDS")

beta <- tidybeta(lda)

topics.terms <- topic_terms(beta)

gamma <- tidy(lda, matrix = "gamma")

classifications <- gamma %>%
  group_by(document) %>%
  top_n(1, gamma) %>%
  ungroup() %>%
  dplyr::select(-gamma) %>%
  as.data.frame()

data.optimal <- merge(data, classifications, by = "document")

data.optimal <- data.optimal %>%
  #dplyr::select(-class) %>%
  rename("class" = topic)

prop.table(table(data.optimal$class))

saveRDS(data.optimal, "data_optimal_total_10.RDS")
```

```r
lda <- readRDS("lda_total_6.RDS")

lda <- ldafun(dtm, 6)



topics <- tidy(lda, matrix = "beta")
top_terms <- topics %>%
  group_by(topic) %>%
  top_n(8, beta) %>%
  ungroup()

top_terms <- top_terms %>%
  ungroup() %>%
  arrange(topic, beta) %>%
  mutate(.r = row_number())

ggplot(top_terms,aes(.r, beta,
                     fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_x_continuous(
    breaks = top_terms$.r,
    labels = top_terms$term)+
  xlab("term")+
  coord_flip()
```

Bibliography

[1] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] F. Cao, J. Z. Huang, and J. Liang. A fuzzy sv-k-modes algorithm for clustering categorical data with set-valued attributes. *Applied Mathematics and Computation*, 295: 1–15, 2017.

[4] J. Cao, T. Xia, J. Li, Y. Zhang, and S. Tang. A density-based method for adaptive lda model selection. *Neurocomputing*, 72(7):1775 – 1781, 2009. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2008.06.011. URL http://www.sciencedirect.com/science/article/pii/S092523120800372X. Advances in Machine Learning and Computational Intelligence.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[6] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] R. Deveaud, E. SanJuan, and P. Bellot. Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique*, 17(1):61–84, 2014.

[8] M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2017. URL https://CRAN.R-project.org/package=data.table. R package version 1.10.4-3.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *Knowledge Discovery and Data Mining*, volume 96, pages 82–88, 1996.

[10] R. Feldman and I. Dagan. Knowledge discovery in textual databases (kdt). In *Knowledge Discovery and Data Mining*, volume 95, pages 112–117, 1995.

[11] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(Mar):1289–1305, 2003.

[12] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[13] E.-H. S. Han, G. Karypis, and V. Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia Conference on Knowledge Discovery and Data Mining*, pages 53–65. Springer, 2001.

[14] T. Hastie, R. Tibshirani, and J. Friedman. Hierarchical clustering. *The Elements of Statistical Learning*, 2, 2009.

[15] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[16] D. A. Hull et al. Stemming algorithms: A case study for detailed evaluation. *JASIS*, 47(1):70–84, 1996.

[17] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[18] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 2013.

[19] J. Ji, T. Bai, C. Zhou, C. Ma, and Z. Wang. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing*, 120:590–596, 2013.

[20] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142. Springer, 1998.

[21] A. Khan, B. Baharudin, L. H. Lee, and K. Khan. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4–20, 2010.

[22] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3): 18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

[23] U. Manber and G. Myers. Suffix arrays: A new method for online string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[24] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. URL https://CRAN.R-project.org/package=e1071. R package version 1.6-8.

[25] T. M. Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37): 870–877, 1997.

[26] W. A. Muir. Department of the air force spend analysis: A report of installation support spend by dod taxonomy. In *Spend Analysis Report*. United States Air Force, 2013.

[27] W. A. Muir. Category management: A concept of operations for improving costs at the air force installation. *USAF CONOPS*, 1, 2014.

[28] M. Nikita. *ldatuning: Tuning of the Latent Dirichlet Allocation Models Parameters*, 2016. URL https://CRAN.R-project.org/package=ldatuning. R package version 0.2.0.

[29] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL http://www.R-project.org/.

[30] D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408, 2002.

[31] W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2017. URL https://CRAN.R-project.org/package=psych. R package version 1.7.8.

[32] A. Rung. *Transforming the Marketplace: Simplifying Federal Procurement to Improve Performance, Drive Innovation, and Increase Savings*. Executive Office of the President, 2014.

[33] H. Saif, M. Fernández, Y. He, and H. Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *Ninth International Conference on Language Resources and Evaluation*. LREC, 2014.

[34] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[35] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[36] K. Schliep and K. Hechenbichler. *kknn: Weighted k-Nearest Neighbors*, 2016. URL https://CRAN.R-project.org/package=kknn. R package version 1.3.1.

[37] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[38] J. Silge and D. Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016. doi: 10.21105/joss.00037. URL http://dx.doi.org/10.21105/joss.00037.

[39] J. Silge and D. Robinson. *Text Mining with R*, volume 1. O'Reilly, 2017.

[40] C. Silva and B. Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 1661–1666. IEEE, 2003.

[41] M. Steinbach, G. Karypis, V. Kumar, et al. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, volume 400, pages 525–526. Boston, 2000.

[42] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440, 2007.

[43] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0.

[44] J. J. Webster and C. Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th Conference on Computational Linguistics-Volume 4*, pages 1106–1110. Association for Computational Linguistics, 1992.

[45] H. Wickham. *scales: Scale Functions for Visualization*, 2017. URL https://CRAN.R-project.org/package=scales. R package version 0.5.0.

[46] H. Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.2.1.

[47] M. Yamamoto and K. W. Church. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1): 1–30, 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
22-03-2018	Master's Thesis	Sept 2016 - March 2018

4. TITLE AND SUBTITLE

Text Classification of Installation Support Contract Topic Models for Category Management

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Sevier, William, C, 1LT , USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-18-M-161

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

Air Force Installation Contracting Agency (HQ AFICA/KA)
1940 Allbrook Drive, Bldg 1, Area A
Wright-Patterson AFB, OH 45433
Darin.ashley@us.af.mil, DSN: 787-7963

10. SPONSOR/MONITOR'S ACRONYM(S)

AFICA/KA

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE;DISTRIBUTION IS 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

Category management is being implemented Air Force-wide, requiring appropriate categorization of Air Force contracts into newly created, manageable spend categories. It has been recognized that current composite categories can be further distinguished into sub-categories leveraging text analytics on the contract descriptions. Upon establishing newly constructed categories, future contracts must be classified into these newly constructed categories in order to be strategically managed. This research proposes a methodological framework for using Latent Dirichlet Allocation to sculpt categories from the natural distribution of contract topics, and assesses the appropriateness of supervised learning classification algorithms such as Support Vector Machines, Random Forests, and Weighted K-Nearest Neighbors models to classify future unseen contracts. The results suggest a significant improvement in modeled spend categories over the existing categories, facilitating more accurate classification of unseen contracts into their respective sub-categories.

15. SUBJECT TERMS

text, classification, machine, learning

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. Bradley C. Boehmke
U	U	U	UU	95	19b. TELEPHONE NUMBER *(include area code)* 937-271-4242