Second Semester CAPSTONE Final Report

Table of Contents

I. Executive Summary

Project Goals / Purpose:

The primary objective of this project is to address the dynamic intersection of Airmen and vendor requests within the realm of Air Force technology innovation. Our focus targets the intricate bridge between Airmen and customer requests, Small Business Innovation Research, and the developmental phases of SBIR/STTR Phase I and II projects. We aimed to proactively identify and capitalize on key correlations that emerge between these critical domains.

Our overarching goal was to conduct a comprehensive analysis of the core SBIR dataset and the comparative tesseract request dataset, encompassing Airmen and vendor requests. Through this analysis, we established a foundational understanding of the SBIR and STTR Phase I and II landscape. This understanding enabled us to identify elements contributing to success, discern trends, and potentially make recommendations or predictions based on our findings. To achieve these objectives, we have employed various methodologies, each tailored to extract meaningful insights from the dataset:

Methods:

1. Tokenization, Stemming, & Lemmatization: Fundamental natural language processing (NLP) techniques involving breaking down textual data into smaller units (tokenization), reducing words to their base/root form (stemming), and transforming words into their base canonical form (lemmatization).
2. Unigrams, Bigrams, & Trigrams (n-grams): Approach to analyzing a data set with sequences of one, two, three, or n adjacent words to capture contextual information
3. Bag of Words: Represents documents, or text as an unordered set of words; this simplifies a complex structure for analysis and keyword identification
4. KeyBERT with embeddings: An NLP approach that incorporates embeddings to exact and prioritize key information from the datasets.
5. t-SNE: T-distributed Stochastic Neighbor Embedding used for dimensionality reduction for two or three dimensions, which can be plotted
6. Batch Processing: Employed to efficiently handle and analyze large volumes of data in manageable chunks, which hopefully speeds up processing time.
7. Advanced Topic Modeling: Techniques to extract latent topics from the data, provide a more refined understanding of potential underlying themes
8. Cosine Similarities: Measures and evaluates the similarity between data points, which helps with clustering and classification tasks
9. TF-IDF with Logistic Regression: Term Frequency-Inverse Document Frequency technique used for effective text classification and predictive modeling using keywords
10. Individual Contributions: Collaborative efforts made by our team members to address the Capstone project needs.

Summary of Results/Key Findings:

Through comprehensive analysis of the core SBIR dataset and comparative Tesseract dataset, our primary objective was to demonstrate the feasibility of understanding the textual data within the context of Tesseract AF mission to bring technology innovation to the field.

Experimentation with the KeyBERT library along with the TF-IDF vectorizer enables us to extract semantic understanding of the textual data. This allowed us to compare Airmen and vendor requests for innovation to SBIR/STTR phase II DoD projects in Phase II. Results are presented in terms of cosine similarity scores for review.

Our team generated the 'Tesseract_Results' dataset, which includes two important columns: 'matchedKeywords' and 'keywordWeightedScore.' These columns were populated by extracting top-performing keywords from Projects in Phase II development in the SBIR dataset. Subsequently, we parsed though the 'solutionStatement' column of the Tesseract dataset to locate and summarize matches.

This approach allowed us to identify matches based on the weighted score, which represents the sum of all top keywords and their frequency within each row. While the occurrence frequency of keywords does not provide a definitive measure, it serves as a visual aid for presorting and understanding requests. Analysis was further expanded to include associating each identified keyword with the top 2 contract IDs from the SBIR dataset.

We've also produced detailed codebooks written in PEP 8 code style guidelines (*link to PEP8 style guide:* https://peps.python.org/pep-0008/), complete with comprehensive inline comments. These codebooks serve as an invaluable resource for stakeholders and future developers, providing guidance for future comparisons between 'solution statement' and 'award descriptions.' This fosters informed decision-making, mitigates duplication of effort, and supports Tesseract's ongoing technological innovation within the Air Force.

Conclusions & Recommendations:

More specialized analysis and work need to be done for keyword extraction and comparison. The provided weighted score and linking to existing keywords is still in preliminary stages and would need to be refined before expanding beyond the DoD-specific Phase II corpora.

Perhaps someone could implement a dropdown field within the Airmen request portal that filters or has choices for select technology keywords that were identified using our methods so that requests may be filtered or sorted in a way that makes them easy to compare to existing Phase II contracts in later steps. Additionally, this could help streamline the request processing phase by identifying duplicate requests or highlighting ones with trending technology keywords used.

II. Project Introduction

This project explored the world of Small Business Innovation Research (SBIR) and Small Business Technology Transfer (STTR) Phase I and Phase II project requests along with Tesseract's Airmen request data with the aim of bridging correlations. Namely, identifying characteristics, keywords, or phrases shared with successful requests (Phase II) and finding a way to showcase potential Phase II requests from the Tesseract Airmen requests. Ultimately by doing so, we could facilitate strategic decision-making, streamline existing processes, reduce redundancy, match existing projects, and overall help foster effective innovations in this ever-changing dynamic landscape that is Air Force technology.

To assist in our aim, the use of Natural Language Processing techniques would help bridge keywords and insights from requests to keywords in the core dataset. However, to do so effectively, we utilized a foundational element of our project; Cross-Industry Standard Process for Data Mining (CRISP-DM) principles to strategically address the challenges encountered during our four sprints and Exploratory Data Analysis. Initial research revealed that harnessing NLP techniques necessitated extensive data preparation and cleaning steps, which continued into our final sprint for better optimization while employing unique approaches discussed later in the methods section.

Some key insights gained from our EDA are as follows:

- Certain agencies possess over 2,000 Phase II contracts, with only four agencies exhibiting a higher Phase II count than Phase I. This could indicate that there are potential factors contributing to their success.
- Disparities exist in contract performance, with some contracts consistently advancing to Phase II while others demonstrate repeated failures.
- We categorized SBIR descriptions into topic groups, offering valuable insights into the overall content of SBIR grants.
- Experimentation with different topic group clusters highlighted the efficacy of smaller clusters in providing meaningful insights.
- Some contract descriptions were missing spaces between words, which would need to be addressed for proper processing later
- Department of Defense-specific contracts could be subsetted out from the main dataset to provide a more related and specified dataset to work with to develop tools and methods more efficiently.
- Focusing on technology keywords from phase II contracts specifically helped us dial in on our target

With that in mind, we set off to use and explore NLP techniques to connect SBIR/STTR Phase I & II projects with airmen requests through our sprints.

Sprint 1: We focused on preparing our data for analysis and modeling while also implementing techniques to extract meaningful insights from text data. We began by handling null values, formatting variables, and removing outliers to ensure the integrity of our analysts. Additionally, we streamlined the text data by removing excessively long words (any over the English word length limit of 45 characters), stop words, and punctuation, which would help improve the quality of text corpora in future processes.
We then incorporated the use of Bag of Words to create a structured vocabulary of terms with frequencies and then visualize them in a meaningful way. We experimented with Term Frequency-Inverse Document Frequency (TF-IDF) technique to assign weights to terms based on their importance. By then integrating the Latent Dirichlet Allocation (LDA) algorithm, we could extract topics from the text data and uncover underlying patterns. Through hyperparameter tuning, we attempted to optimize the LDA model's performance. This sprint emphasized meticulous data preparations, advanced text analysis, and provided a framework for us to continue in the next sprint.

Sprint 2: In this sprint, we primarily focused on refining our analysis by narrowing our dataset to Department of Defence (DoD) contracts and enhancing our pre-processing techniques. We further optimized models by fine-tuning parameters and improving feature extraction, including the exploration of bigrams/trigrams, and updated previous methods (BoW) with the new DoD-tailored data.
Based on sponsor feedback, we refined stopwords and generated new visuals to make our analysis more precise. Statistical inference was conducted on our findings, showcasing discoveries and we also began preparation on the Tesseract request comparative dataset. Throughout the sprint, our goals were to integrate BoW and LDA features into machine learning models, develop well-performing models to enhance data analysis, and refine methods for the next sprint.

Sprint 3: This sprint focused on bridging the core data with the Tesseract request data in a meaningful way through NLP. Initially, we constructed and refined a prefiltered core dataset, specifically targeting DoD phase II contracts, indicating successful contracts. From there we encoded text data from both datasets using pre-trained BERT modeling and calculated cosine similarities between their embeddings. To further optimize this process, we explored batch processing and sampling techniques.
KeyBERT modeling was expanded to identify keywords and their weight vs. frequency to then employ predictive values through Logistic Regression binary analysis. Our main goals were to improve data processing times and embeddings for enhanced analysis, optimize findings for predictive measures, and showcase meaningful connections between the core and target datasets. This sprint emphasized the integration of advanced NLP techniques to drive capabilities and set us up for the final sprint

Sprint 4: Our final sprint prioritized the preprocessing of Phase I & II data using keyBERT, building on the work from sprint 3. We fine-tuned TF-IDF settings, experimented with sample sizes to optimize results, and employed methods to remove non-technology-based terms. Cosine similarity calculations were iteratively refined to ensure accuracy and embeddings were normalized based on original text length to mitigate biases stemming from length variations.

New columns were added to the Tesseract requests dataset featuring weight keyword score, keywords found, and matching contract IDs for the top matched keywords in the existing DoD Phase II dataset. The resulting data was exported, providing a filtered view of matches for analysis. Our goals included enhancing result accuracy, simplifying code blocks for efficiency, converting our project's narrative effectively, and providing tangible progress to our sponsor. Ultimately, we hope to have provided a stepping stone for further progress built upon these efforts.

III. Methods and Results

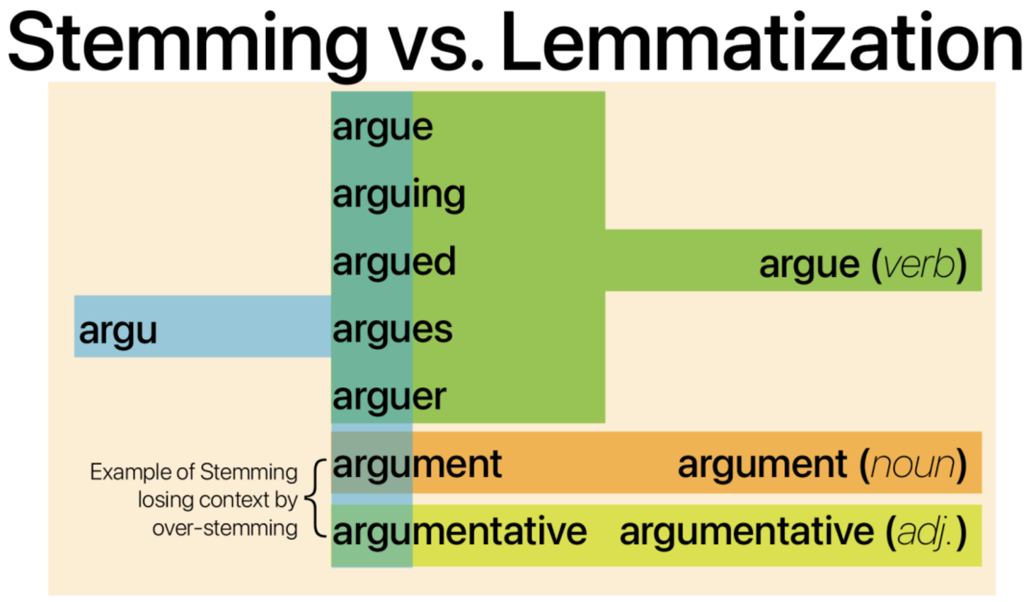
Tokenization, Stemming, & Lemmatization

In order to address the technical aspects of our project, preprocessing of the textual data was critical as it lays the foundation for subsequent analysis and modeling tasks. These steps involve several techniques aimed at standardizing the textual data. In this section, we will discuss tokenization, stemming, and lemmatization.

Tokenization is the process of breaking down the text into smaller units (tokens), which can be words, phrases, symbols, or other meaningful elements for NLPs without losing context. This allows algorithms to more easily identify patterns. The level of granularity is customizable; for this project we went with the word tokenization method breaking up the text data where white space is found.

Stemming & Lemmatization reduce words to the root forms. Both of these techniques are common, but they have slight differences in their approach. Stemming aims to reduce words to their simplest for by removing suffixes, while lemmatization derives the canonical form of words based on their dictionary meanings.

Figure 1: Stemming vs. *Lemmatization*

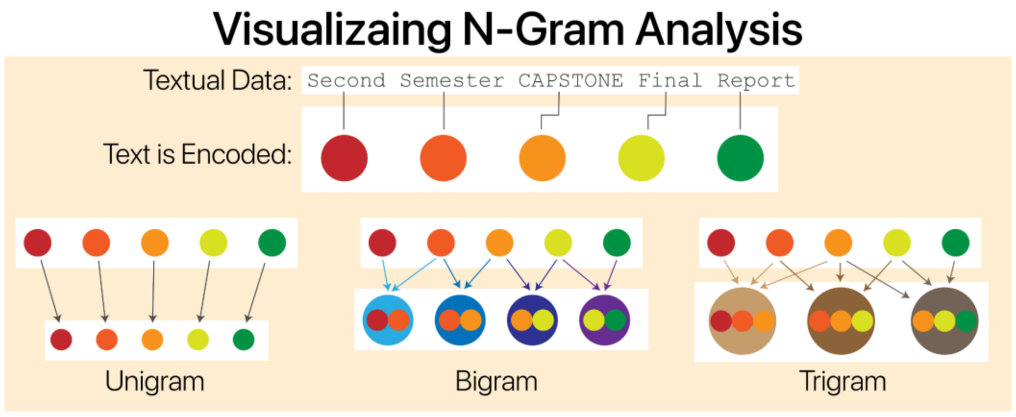


In our experimentation, we compared the effectiveness of stemming and lemmatization for preprocessing Airmen Solution Statements and SBIR Award Descriptions. While both techniques offered benefits in reducing word variations and standardizing the text representations, we found that lemmatization produced more consistent results and the lemmatized text exhibited improved semantic coherence of the underlying language structure. The only drawback to lemmatization was the computational power used, but this was deemed a small cost when establishing a solid foundation for our NLP pipeline.

Unigrams, Bigrams, & Trigrams (N-Grams)

Unigrams, Bigrams, and Trigrams are fundamental concepts in NLP to represent and analyze textual data. Unigrams refer to single words, Bigrams refer to a sequence of two adjacent words, and Trigrams consist of sequences of three consecutive words. In NLP, these n-grams serve as building blocks for various tasks such as text classification, sentiment analysis, and language modeling; however, their effectiveness relies heavily on the context and complexity of the text being analyzed.

Figure 2: N-Gram Analysis of a Sample Text


Visualizaing N-Gram Analysis
Textual Data:
Second Semester CAPSTONE Final Report
Text is Encoded:
Unigram
Bigram
Trigram

There are three main areas where n-grams suffer: (1) Data sparsity, (2) Model Size, and (3) Incorrect modeling by way of incomplete sentences. Since the textual data to be analyzed was collected in chunks (airmen requests, award descriptions, etc..) the model could not identify keywords at low frequency and over-emphasized less important words that were more saturated. This method was run using a predetermined stop-word list and a subsequent custom list of stop-words was identified to assist in mitigating this issue. A bigger and more fundamental problem to n-grams is the incorrect modeling of incomplete or grammatically wrong sentences–which is a common occurrence in real-world textual data collection and is difficult to correct at the raw data collection level.

Despite efforts to address the challenges of implementing n-grams into our NLP process, results were inconclusive. It was anticipated that n-grams would provide valuable insights into the textual data; however, we were unable to derive significant results using n-grams alone. Nevertheless, the direct hands-on experience with n-grams proved useful experience-wise as we made the decision to adopt the KeyBERT and the TF-IDF libraries in later Sprints, both of which offered more robust support for n-gram analysis and keyphrase extraction by examining ranges of n-gram configurations simultaneously.

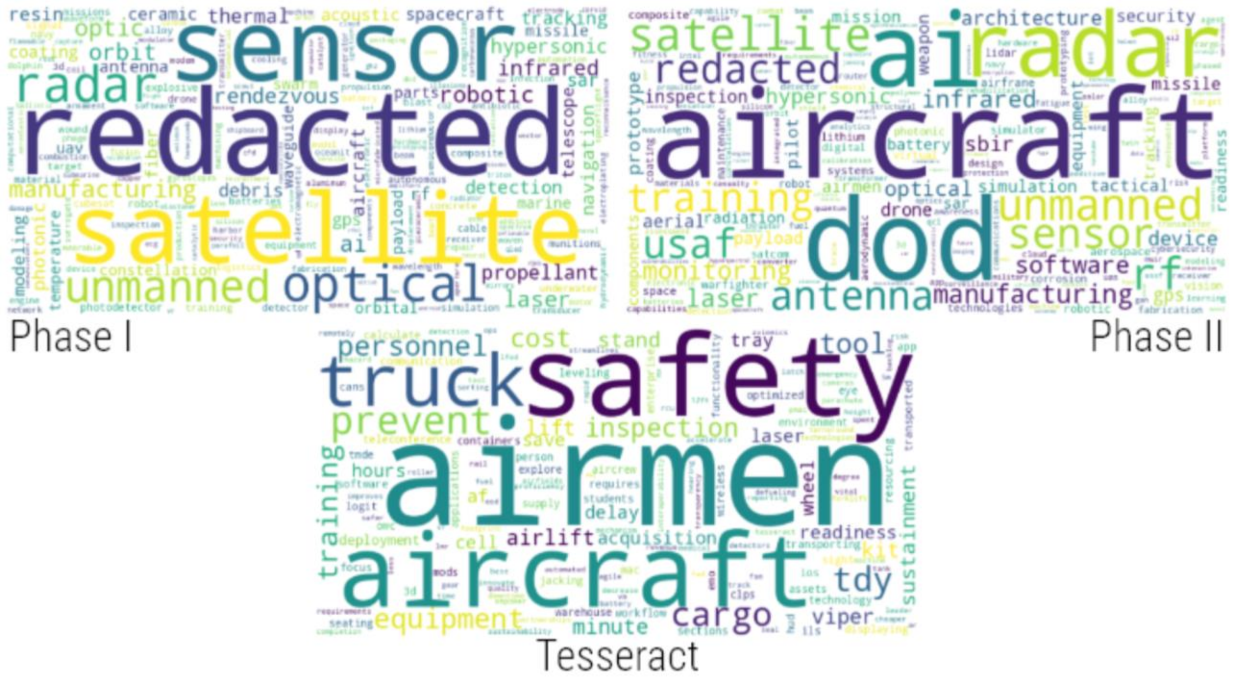
Bag of Words

The BoW method was used in every sprint we conducted, it provides a great visual tool for understanding what is going on in the model in terms of keywords and their frequencies. Tokenization of the text was implemented initially to create a vocabulary of distinct words. Next, we removed common English stopwords, which is a standard step and a few custom words that were noticeably impacting results, i.e; “sbir”, “I”, and “II” for phase I and phase II. This allowed us to then transform each description_award column into a vector representation; where the elements denoted each word’s frequency.

The CountVectorizer module within the scikit-learn library in Python was essential in the process of the tokenization and vectorization mentioned above. We were able to customize the number of words and set thresholds for frequency for a more tailored result.
This method is a simple and effective tool that captures fundamental frequency-based characteristics of text data, which given the focus of our project, where word frequency holds major importance, the model emerged as a very pragmatic approach for discerning and displaying relevant information across the datasets used. However, it has an inherent disregard for word order and structure, so it was used more as a tool for understanding.

To visually represent the outcomes of the Bag of Words analysis; Figure 3 below showcases word clouds generated by BoW representation, offering a summary of the predominant words in the core SBIR data set (both phase I and phase II partitions) and the comparative tesseract dataset:

Figure 3: *Word Clouds for Phase I & II, as well as Tesseract Airmen requests*

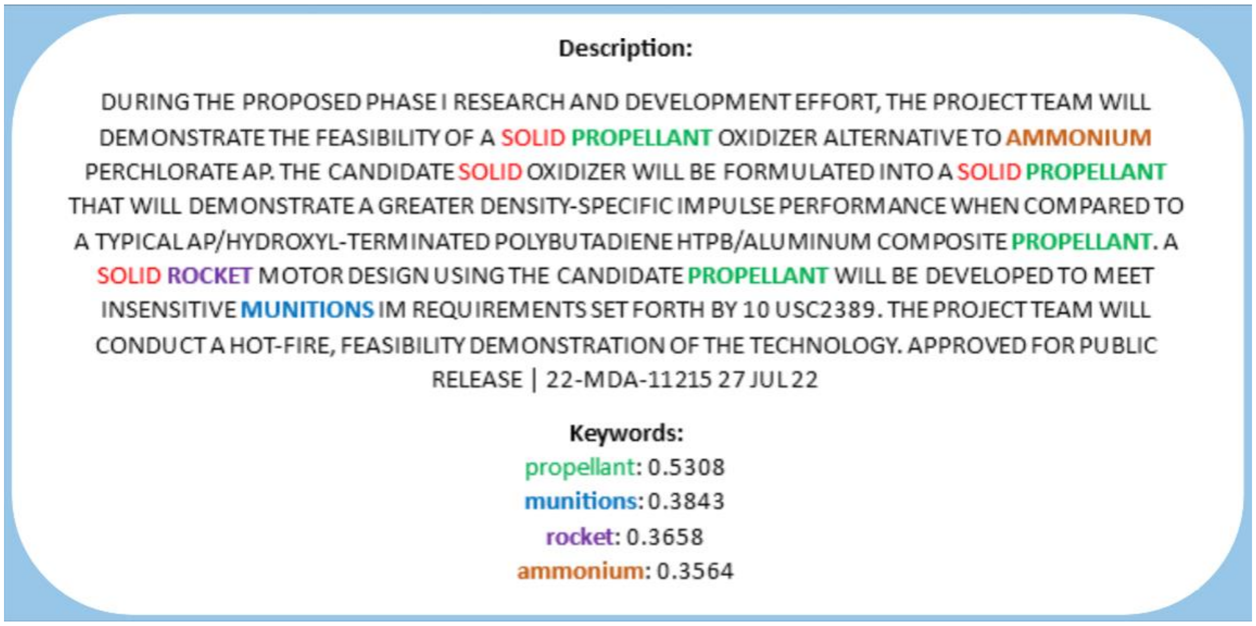


The Bag of Words method produced visuals highlighting the significance of certain words, such as: redacted, aircraft, airmen, dod, sensor, and satellite. This insight reassured us that common stop words were correctly removed and frequency assignment was functional. Additionally, it assisted us by providing a visual when moving forward to focus on technology-related terms/keywords specifically for comparative analysis.

KeyBERT

KeyBERT v0.8, (Bidirectional Encoder Representations from Transformers), developed by Maarten Grootendorst (2023), is a natural language processing tool that leverages embeddings to extract and rank information or keywords from a given text. We chose this method to gain contextual understanding in identifying key terms. it was able to discern semantic relationships and prioritize terms based on the context; which is a complicated nuance missing from several other approaches. Figure 4 below provides an example of a description_award cell and its corresponding keyBERT keywords with their importance score. It is important to note that importance is not the same as frequency. BoW and other methods rate keywords by frequency, but like the aforementioned, keyBERT interprets the context.

Figure 4: *KeyBERT Interpretation with a keyword importance score*



Description:

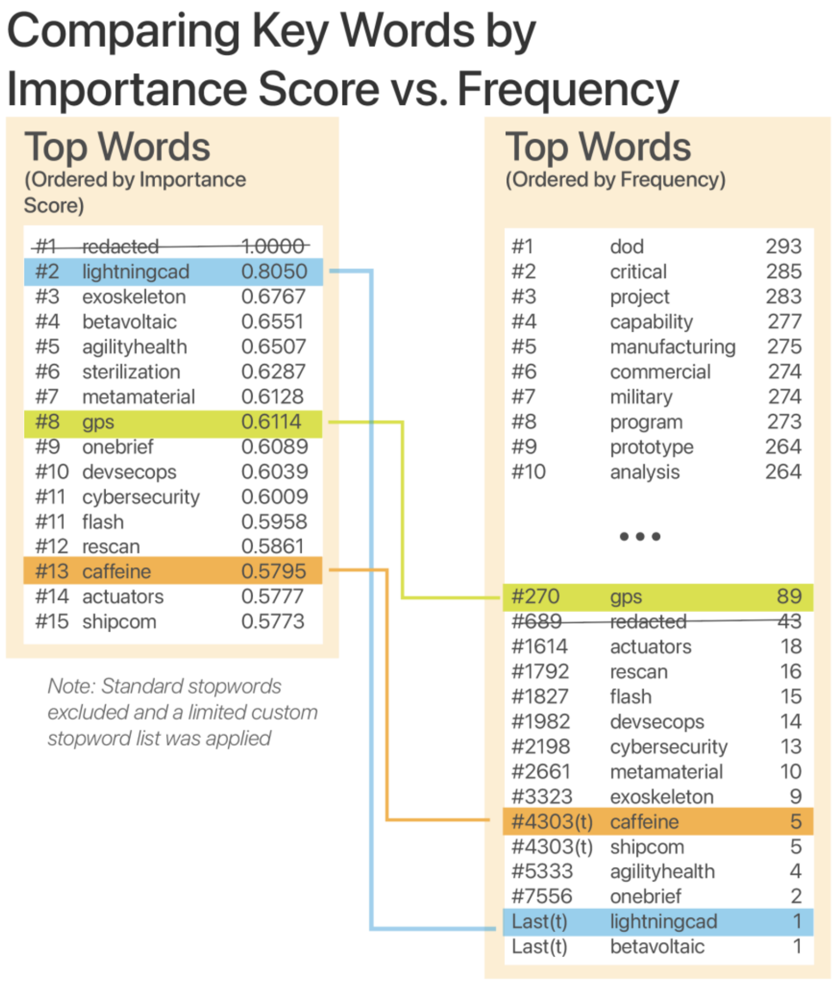
DURING THE PROPOSED PHASE I RESEARCH AND DEVELOPMENT EFFORT, THE PROJECT TEAM WILL DEMONSTRATE THE FEASIBILITY OF A SOLID PROPELLANT OXIDIZER ALTERNATIVE TO AMMONIUM PERCHLORATE AP. THE CANDIDATE SOLID OXIDIZER WILL BE FORMULATED INTO A SOLID PROPELLANT THAT WILL DEMONSTRATE A GREATER DENSITY-SPECIFIC IMPULSE PERFORMANCE WHEN COMPARED TO A TYPICAL AP/HYDROXYL-TERMINATED POLYBUTADIENE HTPB/ALUMINUM COMPOSITE PROPELLANT. A SOLID ROCKET MOTOR DESIGN USING THE CANDIDATE PROPELLANT WILL BE DEVELOPED TO MEET INSENSITIVE MUNITIONS IM REQUIREMENTS SET FORTH BY 10 USC2389. THE PROJECT TEAM WILL CONDUCT A HOT-FIRE, FEASIBILITY DEMONSTRATION OF THE TECHNOLOGY. APPROVED FOR PUBLIC RELEASE | 22-MDA-11215 27 JUL 22

Keywords:

propellant:	0.5308
munitions:	0.3843
rocket:	0.3658
ammonium:	0.3564

Interestingly, we can see that three words, “munitions”, “rocket”, and “ammonium”, were selected yet only appeared once in the description. In contrast, the word “solid” appeared four times but was not considered to be important in this iteration. The importance score ranges from zero to one, so we can gather that “propellant” was moderately significant in this particular request.

Figure 5: *Comparing KeyBERT Importance Score to Frequency*



Comparing Key Words by Importance Score vs. Frequency

Top Words (Ordered by Importance Score)

~~#1~~	~~redacted~~	~~1.0000~~
#2	lightningcad	0.8050
#3	exoskeleton	0.6767
#4	betavoltaic	0.6551
#5	agilityhealth	0.6507
#6	sterilization	0.6287
#7	metamaterial	0.6128
#8	gps	0.6114
#9	onebrief	0.6089
#10	devsecops	0.6039
#11	cybersecurity	0.6009
#11	flash	0.5958
#12	rescan	0.5861
#13	caffeine	0.5795
#14	actuators	0.5777
#15	shipcom	0.5773

Note: Standard stopwords excluded and a limited custom stopword list was applied

Top Words (Ordered by Frequency)

#1	dod	293
#2	critical	285
#3	project	283
#4	capability	277
#5	manufacturing	275
#6	commercial	274
#7	military	274
#8	program	273
#9	prototype	264
#10	analysis	264
	•••	
#270	gps	89
~~#689~~	~~redacted~~	~~43~~
#1614	actuators	18
#1792	rescan	16
#1827	flash	15
#1982	devsecops	14
#2198	cybersecurity	13
#2661	metamaterial	10
#3323	exoskeleton	9
#4303(t)	caffeine	5
#4303(t)	shipcom	5
#5333	agilityhealth	4
#7556	onebrief	2
Last(t)	lightningcad	1
Last(t)	betavoltaic	1

KeyBERT's processing and ranking proved instrumental during the implementation of other models by providing a solid foundation to build. Due to our core dataset's size, processing time was impacted. Subsetting and batch processing were explored.

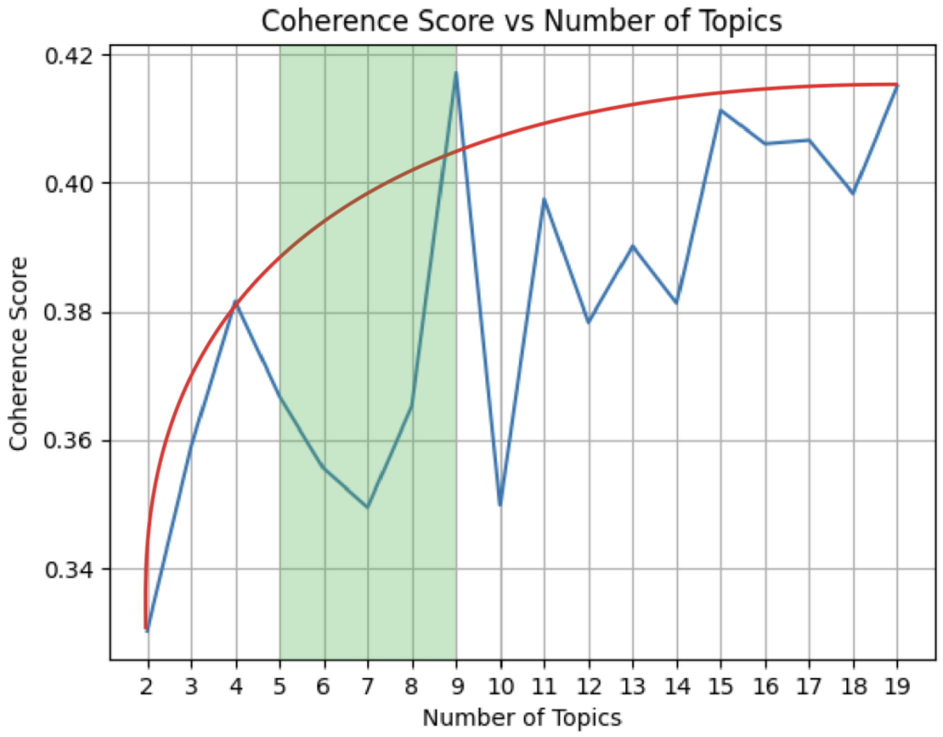
Advanced Topic Modeling

Advanced topic modeling techniques aim to enhance the quality and interpretability of topics generated from textual data. This project used the Latent Dirichlet Allocation (LDA) model to generate various numbers of topics ranging from 2 to 19 to determine an optimal number of topics to best represent the underlying structure of the corpus. To evaluate the quality of the topics generated coherence scores were computed.

We anticipate coherence scores will peak and plateau at a certain range of the number of topics, this indicates the topics have become more interpretable and coherent as the number increases to a certain point; however, beyond the optimal range, coherence scores either level off or being to decrease as the topics become overly fragmented and less representative of the underlying themes of the textual data. The goal is to find the correct amount of granularity.

During our trials, the most consistent number of topics with the best coherence scores (before a plateau) were between 5 and 9. Within this range, results had a great deal of variability which can be partly attributed to the random nature of the sample data.

Figure 6: Example Trial of Advanced Topic Modeling and Coherence Scores

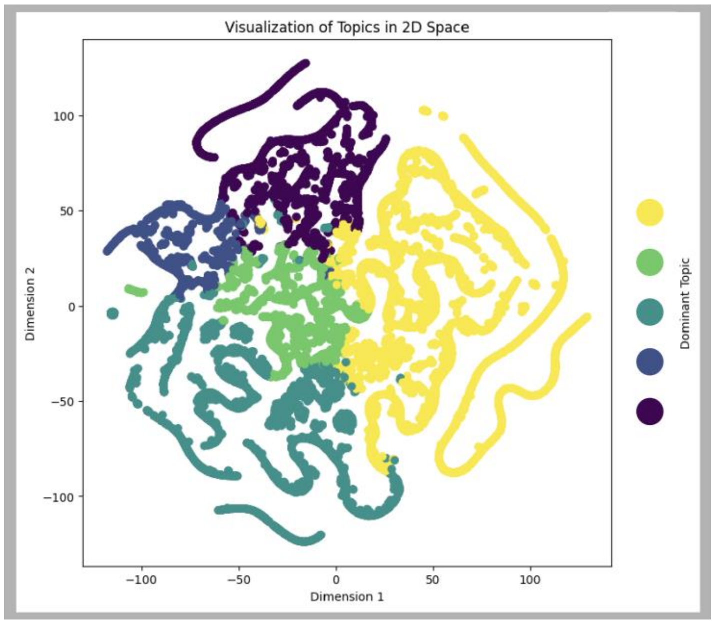


The figure above visualizes coherence scores across different numbers of topics in one of our trials. The blue line represents the actual coherence scores obtained from the LDA models, while the red line depicts an ideal pattern. The green area highlights the range of numbers of topics where we observed the highest nad most stable coherence scores.

t-Distributed Stochastic Neighboring Embedding (t-SNE)

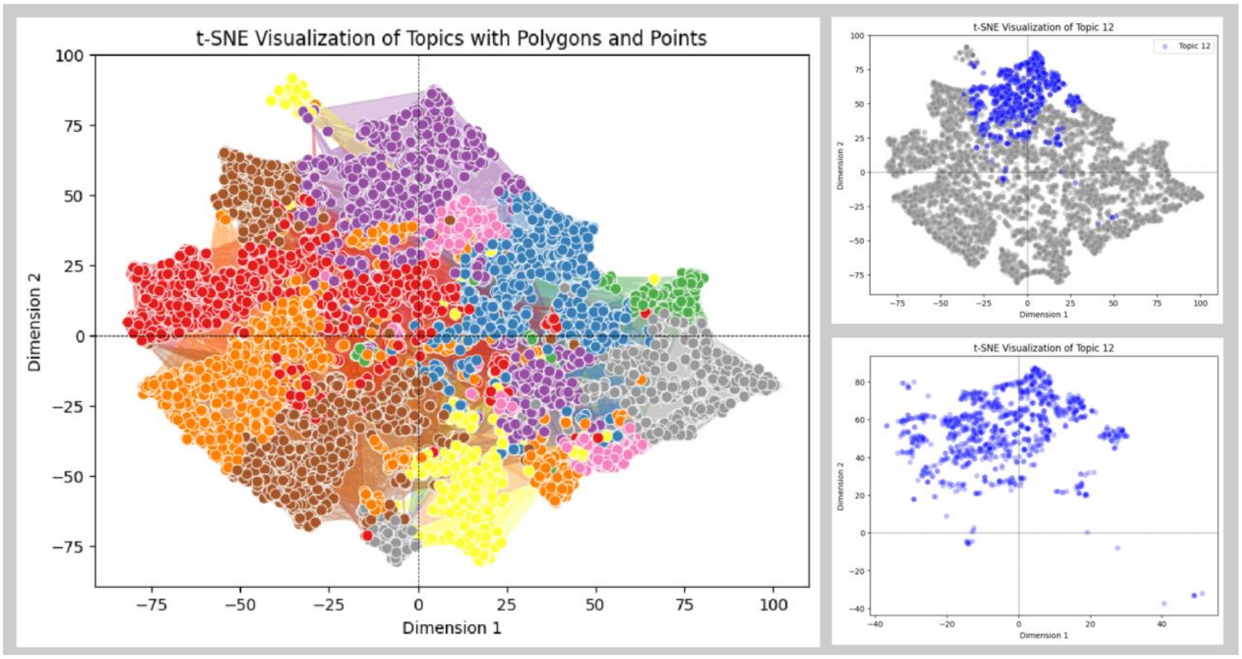
The t-SNE library is a machine learning and data visualization package used for unsupervised non-linear dimensionality reduction and high-dimensional data visualization. It is particularly effective for visualizing complex datasets in lower-dimensional space while preserving the local substructure of the data points. The t-SNE algorithm works by modeling the similarity between pairs of data points in high-dimensional space and in lower-dimensional space. It utilizes a cost function to minimize differences between the data points and in doing so, it maps the higher-dimensional data onto a 2-dimensional plane where similar data points cluster together and dissimilar points are further apart helping to visualize the underlying structure of the data. Figure 7 below was produced as a result of topic modeling the description_award data into 5 distinct topics.

Figure 7: t-SNE Visualization of Topic Generation Created



The visualization above depicts the categorized 'description_award' raw textual data to one of 5 topics generated from an LDA model and color-coded to a specific topic. There is obvious data clustering which validated our ability to conduct topic modeling to group projects. The visualization also indicated that outliers existed and that there would be some margin of error that would need accounting for.

Figure 8: t-SNE Higher Number Topic Generation/Topic Isolation

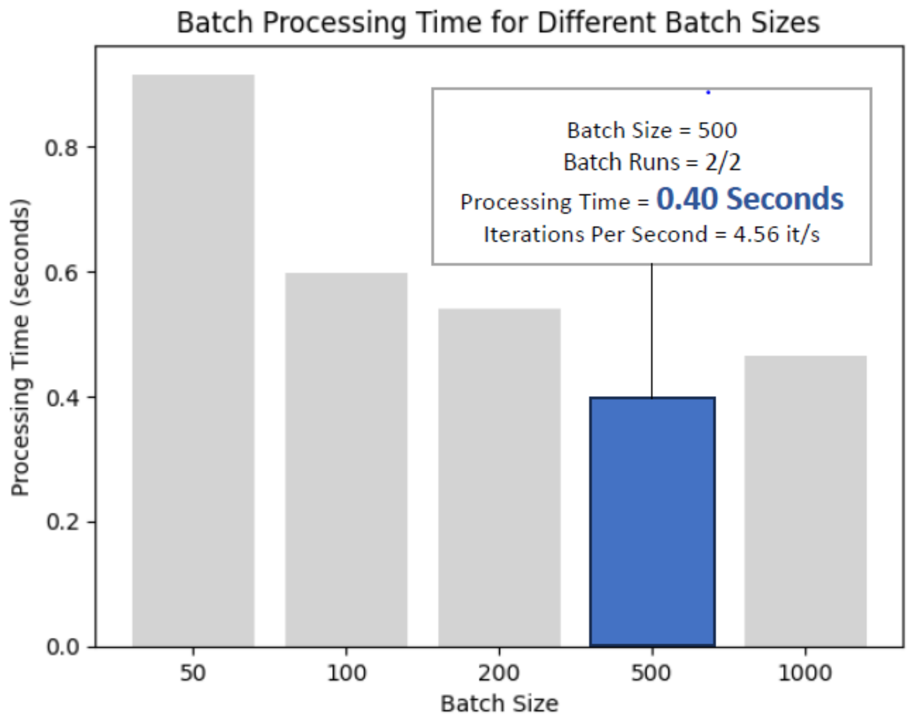


The team further utilized t-SNE as a guide in determining an optimal number of topics for modeling and then delved into the intricate details of the individual topics. In Figure 8 above, 20 topics were generated creating a very busy visualization with somewhat less distinct clusters. Topics were isolated and examined on an individual basis to better infer insights from the data. This iterative process of visual analysis facilitated a deeper understanding of the underlying structure of the textual data and guided refinement of the topic modeling approach.

Batch Processing

Batch processing was used in Sprint 3 and improved in Sprint 4. Batch processing is a method for efficiently processing a large volume of data by dividing that data into smaller volumes, called batches. The number of batches run can affect the overall efficiency of the batch processing method. The optimal number of batches varies by use case and depends on several factors - i.e. size of the data, processing capabilities, methods used, and more. To optimize batch processing in our case, we measured iteration speed and overall processing time for variable batch sizes. The batch size associated with the lowest overall processing time was deemed the most efficient.

Figure 9: Batch Processing Efficiencies for Different Batch Sizes

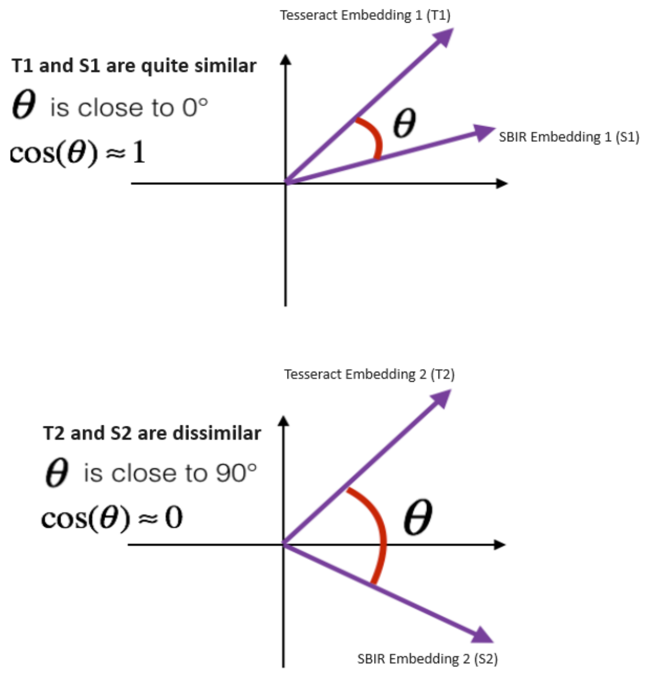


Our primary reason for implementing this method was to attempt to reduce the time to process cosine similarities between 'sbir_embeddings' and 'tesseract_embedding'. Our dataset is small, relative to the magnitude of the processing power at our disposal. Because of that, differences in batch processing efficiencies were sensitive to confounding factors - such as other tasks concurrent on the processor. As a result, our methods did not consistently indicate the same optimal batch size on each iteration. Additionally, differences in processing times were overall negligible. Larger data sets, which are highly processor intensive, would likely yield more informative variations in processing efficiency across batch sizes. For this reason, we recommend that these methods be trialed with larger datasets. The work done for this project should be interpreted more so as a proof of concept that altering batch sizes does impact processing efficiency and likely would be more meaningful as it is scaled up.

Cosine Similarities

Cosine similarity was introduced in Sprint 3 and improved in Sprint 4. Cosine similarity involves computing the cosine of the angle between two vectors, which produces a value between -1 and 1. It is used to determine how similar two vectors are. In natural language processing (NLP), text can be converted to vector representations based on their semantics. Cosine similarity values can be computed for these vector representations. A cosine similarity of 1 means the vectors are perfectly aligned, pointing in the same direction. A cosine similarity of 0 means the vectors are orthogonal (perpendicular), indicating no similarity. A cosine similarity of -1 means the vectors are perfectly opposed, pointing in opposite directions.

Figure 10: Cosine Similarity for SBIR and Tesseract Analysis

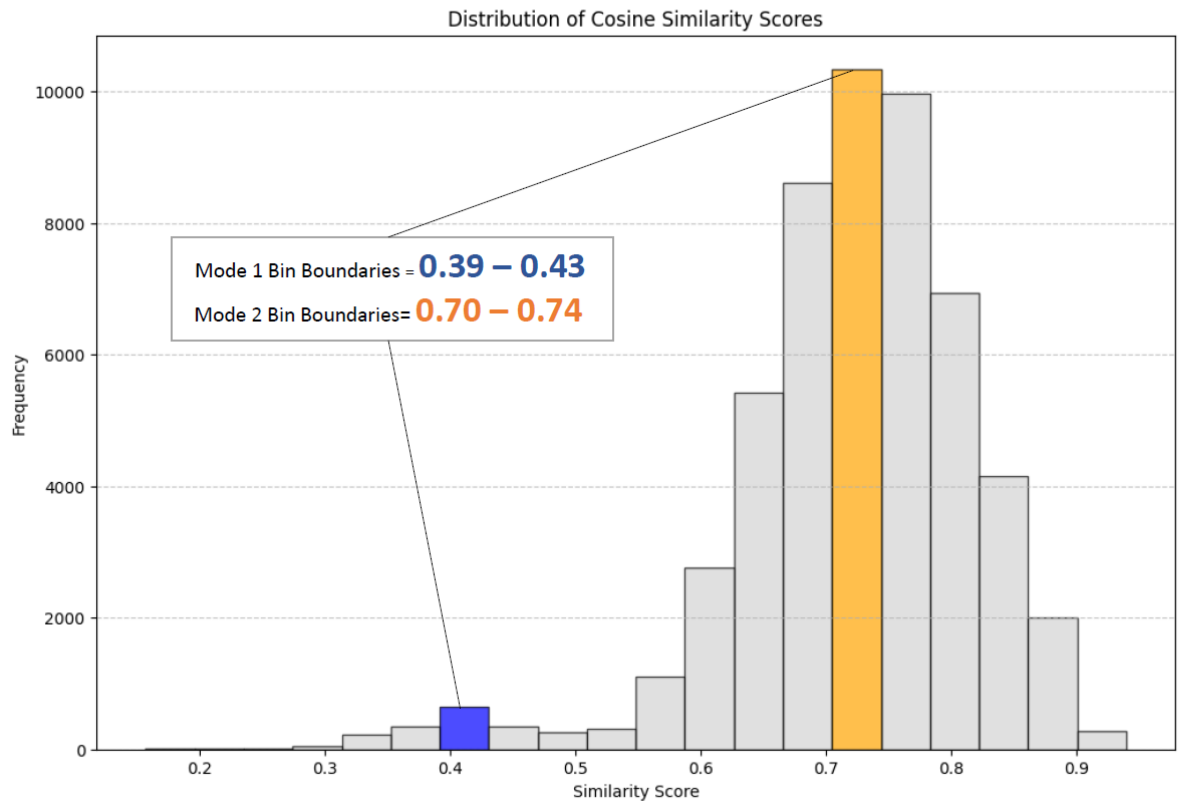


Analytics Vidhya. (04/13/2024). Best NLP algorithms to get document similarity. *Medium*. Retrieved from https://medium.com/analytics-vidhya/best-nlp-algorithms-to-get-document-similarity-a5559244b23b

We utilized a BERT model to convert SBIR descriptions and Tesseract solution statements into vector representations called BERT embeddings (Bidirectional Encoder

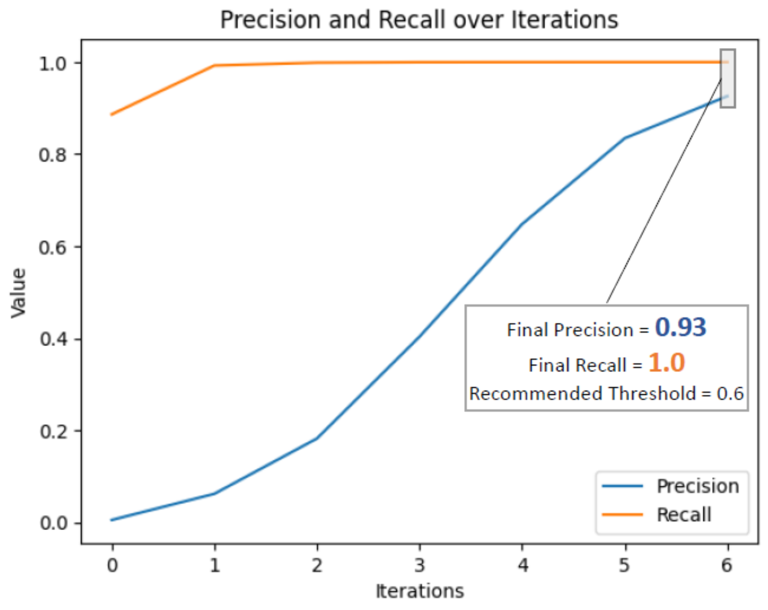
Representations from Transformers). BERT embeddings were utilized to calculate cosine similarity scores between SBIR descriptions and Tesseract data, offering insight into similarities between existing and proposed projects. Notably, the scores appeared bimodally distributed, with modes near .4 and .7.

Figure 11: Distribution of Cosine Similarity Scores



In Sprint 4, we aimed to iteratively refine the cosine similarity calculation. The cosine similarity calculation relies on a threshold parameter, which defines the minimum cosine similarity needed to be included in the output of the computation. Depending on this threshold, there can be varying numbers of false positives, false negatives, and true positives. Precision is defined as the ratio of true positives to the sum of true positives and false positives. Recall is defined as the ratio of true positives to the sum of true positives to false negatives. Each iteration involved increasing or decreasing the threshold, if either the precision or recall was below 90%. A value of 90% is used, because it ensures high recall and precision, while aiming to reduce the risk of overfitting. This process allowed us to determine a threshold value (Threshold=0.6), which is associated with high precision and recall. This threshold may not be optimal and is limited in its generalizability, but offers an evidence based approach to threshold setting versus an arbitrary approach. In future work on this method, we would recommend that this iterative process be repeated on new datasets. Perhaps, the average threshold over time should be used. Additionally, we envision this method being implemented in conjunction with other methods to reduce grant redundancy.

Figure 12: Precision and Recall over Iterations



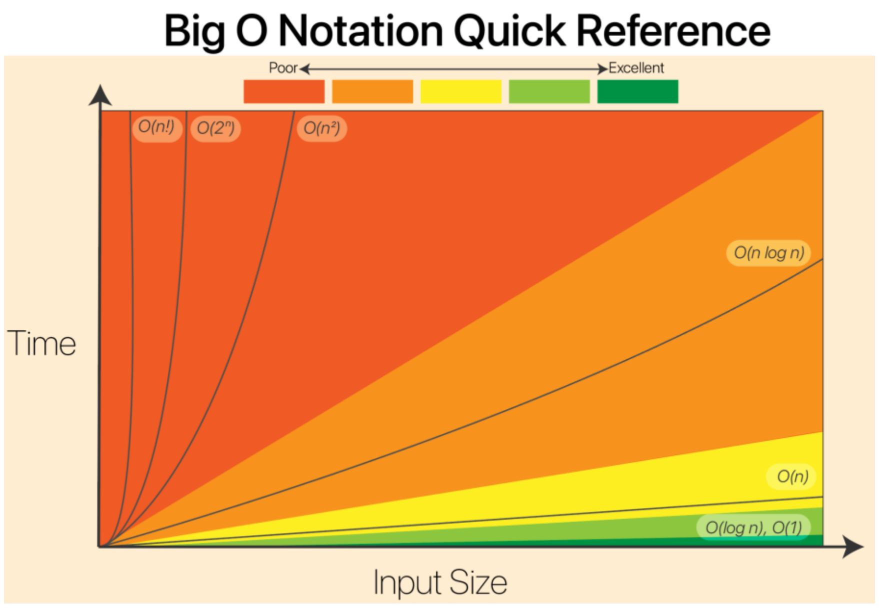
Big O Notation (Time Complexity)

Big O notation was included in the final codebook and is a mathematical notation used to describe the time complexity of an algorithm. Time complexity is representative of the upper bound or worst-case scenario for the growth rate of an algorithm's runtime or memory usage as the size of the input data increases.

In practical terms, the inclusion of Big O notation in the codebook is beneficial because it provides important insights into the efficiency of the algorithms used in our code. It serves as documentation for developers, allowing them to understand the expected performance characteristics and make informed decisions when optimizing the code for better performance.

The codebook assumes a baseline understanding of this concept; however, there are various online resources where developers can find refresher courses on Big O notation.

Figure 13 Big O Notation Quick Reference



TF-IDF with Logistic Regression

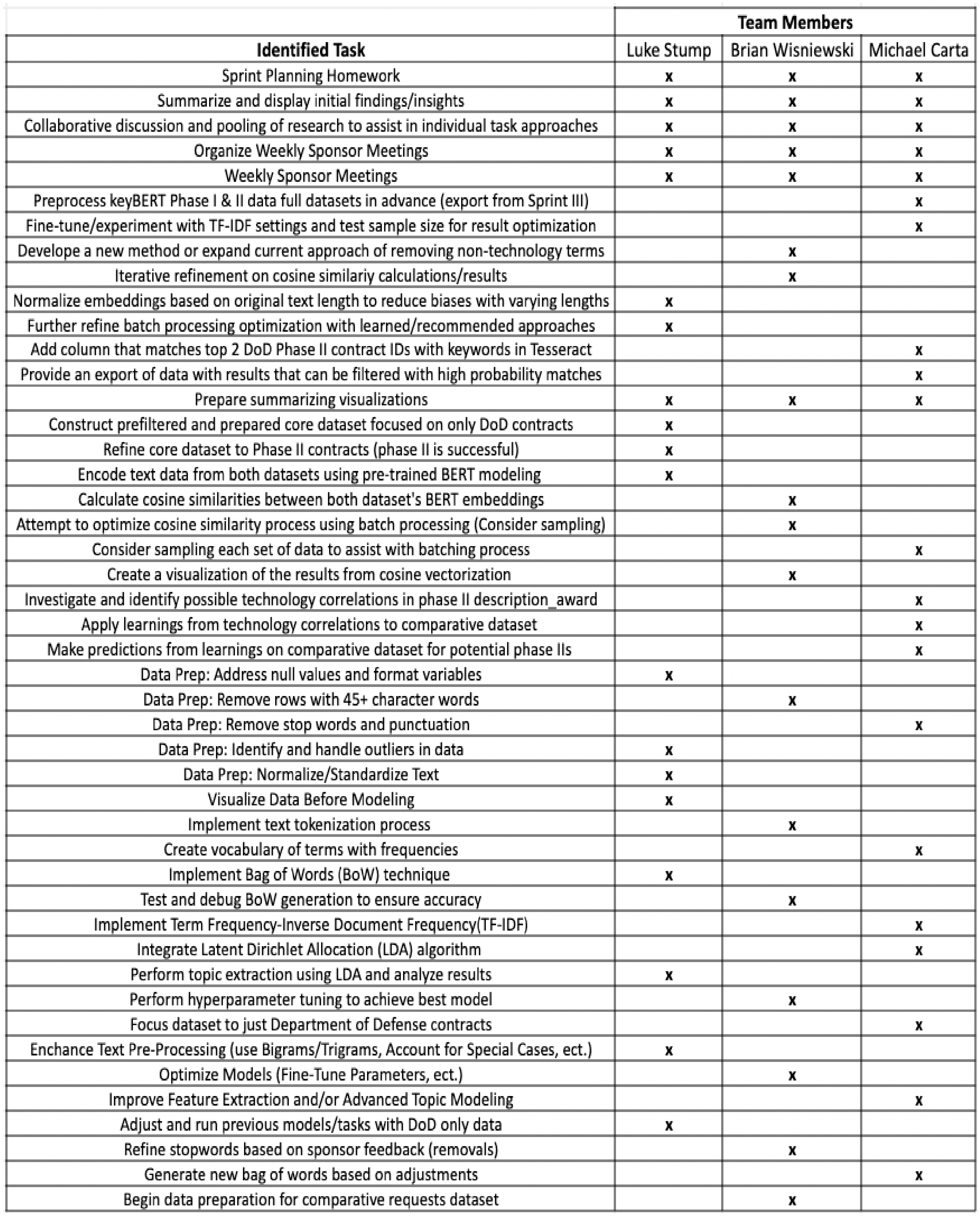
The team employed a combination of TF-IDF (Term Frequency-Inverse Document Frequency) and Logistic Regression as chosen methods for analyzing the datasets. It is a widely accepted natural language processing technique that evaluates word importance within a document relative to the over-collection of documents/samples; it focuses on not only the frequency but also the rarity of a term across the entire dataset. Specifically, we utilized the "TfidfVectorizer" from the scikit-learn library to transform the raw DoD Phase II data into a representation with each word being assigned a weight based on its significance. Given the nature of our core dataset, which included textual descriptions of SBIR awards, TF-IDF was suitable for capturing unique characteristics in our context.

Logistic Regression, a binary classification algorithm, was then employed to model the relationship between the TF-IDF features and our target variable (classification of a term of being phase 2 or not). The resulting model captured important terms indicative of phase 2 projects and generated the probability of extracted Tesseract Airmen request corpus values of being phase 2. The classification report, accuracy, and confusion matrix allowed us to view the comprehensive insights into the performance.

These results contribute valuable insights to our project goal, aiding in the identification of potential phase 2 requests in the Tesseract Airmen request dataset.

Individual Contributions

Figure 14: *Task Table*



	Team Members		
Identified Task	Luke Stump	Brian Wisniewski	Michael Carta
Sprint Planning Homework	x	x	x
Summarize and display initial findings/insights	x	x	x
Collaborative discussion and pooling of research to assist in individual task approaches	x	x	x
Organize Weekly Sponsor Meetings	x	x	x
Weekly Sponsor Meetings	x	x	x
Preprocess keyBERT Phase I & II data full datasets in advance (export from Sprint III)			x
Fine-tune/experiment with TF-IDF settings and test sample size for result optimization			x
Develope a new method or expand current approach of removing non-technology terms		x	
Iterative refinement on cosine similariy calculations/results		x	
Normalize embeddings based on original text length to reduce biases with varying lengths	x		
Further refine batch processing optimization with learned/recommended approaches	x		
Add column that matches top 2 DoD Phase II contract IDs with keywords in Tesseract			x
Provide an export of data with results that can be filtered with high probability matches			x
Prepare summarizing visualizations	x	x	x
Construct prefiltered and prepared core dataset focused on only DoD contracts	x		
Refine core dataset to Phase II contracts (phase II is successful)	x		
Encode text data from both datasets using pre-trained BERT modeling	x		
Calculate cosine similarities between both dataset's BERT embeddings		x	
Attempt to optimize cosine similarity process using batch processing (Consider sampling)		x	
Consider sampling each set of data to assist with batching process			x
Create a visualization of the results from cosine vectorization		x	
Investigate and identify possible technology correlations in phase II description_award			x
Apply learnings from technology correlations to comparative dataset			x
Make predictions from learnings on comparative dataset for potential phase IIs			x
Data Prep: Address null values and format variables	x		
Data Prep: Remove rows with 45+ character words		x	
Data Prep: Remove stop words and punctuation			x
Data Prep: Identify and handle outliers in data	x		
Data Prep: Normalize/Standardize Text	x		
Visualize Data Before Modeling	x		
Implement text tokenization process		x	
Create vocabulary of terms with frequencies			x
Implement Bag of Words (BoW) technique	x		
Test and debug BoW generation to ensure accuracy		x	
Implement Term Frequency-Inverse Document Frequency(TF-IDF)			x
Integrate Latent Dirichlet Allocation (LDA) algorithm			x
Perform topic extraction using LDA and analyze results	x		
Perform hyperparameter tuning to achieve best model		x	
Focus dataset to just Department of Defense contracts			x
Enchance Text Pre-Processing (use Bigrams/Trigrams, Account for Special Cases, ect.)	x		
Optimize Models (Fine-Tune Parameters, ect.)		x	
Improve Feature Extraction and/or Advanced Topic Modeling			x
Adjust and run previous models/tasks with DoD only data	x		
Refine stopwords based on sponsor feedback (removals)		x	
Generate new bag of words based on adjustments			x
Begin data preparation for comparative requests dataset		x	

IV. Conclusions, Next Steps, and Recommendations

Conclusion:

The Tesseract_Results dataset that gets exported at the end of the Sprint 4 codebook is filtered based on the two newly generated columns; “matchedKeywords” and “keywordWeightedScore”. What we were able to do was take the top performing keywords from the Phase II SBIR dataset and parse through the Tesseract Airmen request's solutionStatement column row-by-row to locate and summarize matches. The dataset is filtered from the highest weighted score down; the weighted score is the sum of all top keywords and their frequency within the row. The frequency of use of a keyword should not be used as a definitive measure, but rather as a visual tool to help presort and understand requests. Furthermore, we matched each of the row's discovered keywords with the top 2 contract IDs from the core DoD Phase II SBIR dataset for comparative analysis possibilities.

Some methods that we utilized are at a state that we didn't have time to bring to a conclusive end-state. These methods are rich with opportunities for next steps. Examples include the cosine similarity and batch processing code included in the Sprint 3 & Sprint 4 codebook. Cosine similarity may be used to drive efficiency in funding and effort allocation, by informing of semantic similarity across historical and new project data. Batch processing streamlines workflows by increasing computational efficiency. They are useful for these purposes in their current state, but don't produce neatly packaged and high-level results. These methods, amongst others, stand primarily as preliminary foundations for future work.

1. Our exported dataset, Tesseract_Results, is refined using advanced keyword matching and weighted scoring, offering a comprehensive overview of project relevancy.
2. Leveraging top-performing keywords from the Phase II SBIR dataset, we efficiently identify and summarize relevant matches within the Tesseract Airmen requests, facilitating streamlined analysis.
3. The weighted scoring system provides nuanced insights into request relevance, guiding prioritization and resource allocation strategies.
4. Integration of matched keywords with top contract IDs from the Phase II SBIR dataset enables strategic comparative analysis, fostering informed decision-making.
5. While methods like cosine similarity and batch processing are in their nascent stages, they demonstrate promising potential for optimizing resource allocation and workflow efficiency.
6. Cosine similarity analysis offers a sophisticated approach to identifying semantic similarities across project data, aiding in strategic funding allocation.
7. Batch processing enhances computational efficiency, laying the groundwork for future optimizations in project analysis and decision-making processes.

Proposed Next Steps:

Expand approaches to process more than just DoD phase I or II data; due to the large size of the dataset, OSC or another supercomputer resource would be recommended. This will complicate results, as there are variations in the agencies submitting requests.

Gather more “solutionStatement” requests from Tesseract for a more thorough representation of incoming data for keyword matching.

Make Cosine Similarity Outputs more intuitive and high-level. Implement a user-friendly interface that allows users to input project descriptions or lists and receive an exported file containing the most semantically similar results. This streamlines the process and eliminates the need for users to deal with technical details.

Use batch processing in other methods. This might not reduce processing time, but testing whether it does is straightforward with the code we included.

Recommendations:

Aside from implementing the proposed next steps, we do have some recommendations:

1. We recommend utilizing high-performance computing resources, such as the OSC, for processing the datasets more efficiently.
2. We recommend enhancing data collection and leveraging our methods with larger volumes of data, to increase the explanatory power of conclusions.
3. Some of our results are hard to interpret without a technical background, so we recommend improving user interaction by abstracting out technical details in the results.
4. As with any data science project, we recommend always keeping up with new methods that may be more efficient than the ones we used.
5. We also recommend ensuring balanced data handling by maintaining a balance between result comprehensiveness and strategies for improving computational efficiency, such as dataset subsetting.
6. Finally, certain methods could be expanded to other columns. For example, there is a column called “technologyType”, which contains keywords not listed in the solutionStatement column. It could be valuable to process those in tandem.

V. References & Additional Information

Grootendorst, M. (2023). *KeyBERT v0.8*: Minimal keyword extraction with BERT. Retrieved from [https://maartengr.github.io/KeyBERT/index.html]

Analytics Vidhya. (04/13/2024). Best NLP algorithms to get document similarity. *Medium*. Retrieved from https://medium.com/analytics-vidhya/best-nlp-algorithms-to-get-document-similarity-a5559244b23b

Zimmermann, Jaime T., Champagne, Lance, E., Dickens, John M., Haze, Benjamin T. (2024). Approaches to Improve Preprocessing for Latent Dirichlet Allocation Topic Modeling [working paper]. Decision Support Systems.

For any questions or concerns, please feel free to reach out to the following contacts:

- Luke Stump: stump204@buckeyemail.osu.edu
- Brian Wisniewski: wisniewski.62@osu.edu
- Michael Carta: carta.2@osu.edu