Proceedings of the ASME 2021 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2021
August 17–20, 2021

DETC2021/CIE-70694

KNOWLEDGE DISCOVERY FOR EARLY FAILURE ASSESSMENT OF COMPLEX ENGINEERED SYSTEMS USING NATURAL LANGUAGE PROCESSING

Sequoia R. Andrade
HX5, LLC.
Moffett Field, California

Hannah S. Walsh
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, California

ABSTRACT

Emerging complex engineered systems may have unexpected safety issues due to novel operational environments, increasing autonomy, human-machine interaction, and other factors. To prevent failures in operation or testing that necessitate costly redesign, it is desirable to predict likely failure modes early in the design process. Information about past engineering failures in natural language format presents one possible solution by enabling the retrieval of information that can inform new designs. However, identifying documents containing usable information and extracting the required information can be prohibitively time-consuming when implemented at scale. In this research, an automated natural language processing (NLP) framework is proposed to discover relevant knowledge from documents containing failure-related design information. The framework is applied to NASA's Lessons Learned Information System (LLIS), which is publicly available. Documents containing usable information are filtered using two different NLP-based models. Next, from the identified usable documents, a failure taxonomy is extracted using a partitioned hierarchical topic modeling approach. Partitions of the document describe different sections of the failure taxonomy – i.e., failure, cause of failure, and recommendations – as indicated by the structure of the original document. The extracted failure taxonomy can be leveraged in early design failure assessment methods. Moreover, the framework can be used to identify documents containing usable failure-related design information from other databases and extract relevant information from these documents.

1 INTRODUCTION

Failure taxonomies are used to describe engineering failures, and are used to facilitate certain failure analysis methods. Building failure taxonomies relies on historical knowledge and precedent. Increasingly, designers have access to large amounts of data from past projects. Design repositories [1] and NASA's Lessons Learned Information System (LLIS) [2] are some such examples. These repositories contain rich case studies and lessons from previous designs, enabling designers to discover and leverage knowledge from past designs to improve new designs. However, with the sheer volume of information available, it can be challenging to identify, extract, and find meaning in such information. Moreover, existing repositories of information are often mixed-quality, loosely structured, and unspecialized. This can create challenges in effectively extracting high quality information. Natural language processing presents a solution to discovering knowledge from such repositories. In particular, topic modeling approaches, such as Latent Dirichlet Allocation (LDA), provide a means of obtaining the underlying themes, or topics, within a corpus. LDA is unsupervised, which is additionally useful for long, complex documents which may be prohibitively time-consuming to label. Topic modeling is useful for extracting higher-level concepts within a corpus.

Hierarchically organized failure taxonomies are of particular interest in the design of emerging operational concepts that include technology with little historical precedent, such as autonomy. For such systems, knowledge obtained from historical data will only be as useful as it is generalizable. The utility of obtaining a hierarchy rather than simple correlations between

risk factors and failure modes is that the discovered knowledge can be understood at different levels of abstraction. .Generally, the more highly abstracted knowledge is more easily extensible to new problems, and by discovering underlying patterns and themes in addition to design-specific knowledge, we can more effectively leverage knowledge gained from past lessons for new designs. By using hierarchical topic modeling, specifically hierarchical Latent Dirichlet Allocation (hLDA), rather than conventional LDA, this research extracts knowledge that can be understood at multiple levels of abstraction, thus improving generalizability to new design problems.

Assembling a failure taxonomy necessitates the extraction of multiple disparate aspects. Thus, standard topic modeling of entire documents in a corpus will fail to distinguish between the different aspects that generally comprise entries in a failure taxonomy. Specifically, three aspects are extracted – cause, failure, and recommendation – following conventions in Failure Modes and Effects Analysis (FMEA) and other failure taxonomies in the literature [3] to the extent possible from information provided in the corpus. This research augments hLDA by partitioning the documents in a corpus, in this case by section, such that each unique aspect can be identified. After topic modeling, these aspects are assembled into a complete failure taxonomy.

Topic modeling is a powerful tool, but it tends to suffer from the "garbage in, garbage out" paradigm that plagues many data science techniques. Therefore, it is important to confirm that the documents used in the topic model are of significant relevance to the desired task. Repositories of design information tend to be relatively unspecialized and usable for various types of applications, but this means that, for example, only a fraction of the corpus may be relevant for a particular use case. Thus, before implementing the partitioned hLDA framework, we additionally propose a supervised NLP-based solution for selecting relevant documents. This step improves the quality of the inputs to the topic model and, consequently, the quality of the topic outputs.

In this research, we propose a natural language processing framework for (1) identifying and (2) extracting knowledge related to engineering failures in a mixed quality, loosely structured, unspecialized corpus and assemble this knowledge into a failure taxonomy. We apply this framework to NASA's LLIS, which contains information about historical failures in natural language format. First, relevant documents are identified using a supervised NLP-based solution for selecting relevant documents in an unspecialized, mixed-quality corpus. Second, knowledge used to build the failure taxonomy is extracted using a proposed partitioned hierarchical topic modeling method. The extracted knowledge is assembled into a complete failure taxonomy. The proposed framework can be used to extract specialized information from any general repository of natural language information, while the assembled failure taxonomy can be used to inform early stage design of novel complex engineered systems.

2 BACKGROUND

The use and reuse of knowledge is critical to the design process. The use of knowledge prevents designers from spending time and resources developing and assessing ideas that have already been thoroughly investigated. Further, and with respect to failure analysis in particular, the use of knowledge can help designers prevent repeating mistakes from previous designs. In safety critical systems, this becomes incredibly important. Prior research has identified barriers to knowledge reuse in organizations [4]. Systematic approaches have also been used to manage knowledge [5]. Other research has focused on extracting knowledge from existing datasets, i.e. knowledge discovery [6]. Knowledge discovery generally refers to a process by which new knowledge is extracted from data [7]. Many such processes make use of natural language processing and/or data mining algorithms [7], with a broad range of applications. In this research, we apply such approaches to the problem of discovering a failure taxonomy in design. We first review existing failure taxonomies in design. Then, we explore prior use of natural language processing in design and failure analysis more specifically.

2.1 Failure Taxonomies in Design

Failure taxonomies facilitate descriptive classifications of failures and are key components of methods such as Failure Modes and Effects Analysis (FMEA). For instance, Tumer et al. describe a failure taxonomy based on physics-based descriptions of failures [8]. Taxonomies have been built to capture historical knowledge of failures and from observational studies [9]. Classification schemes that are hierarchical in particular are understood to result in higher quality classifications of failures [3]. Other hierarchical classification schemes include the functional basis [10]. In this research, by learning from historical project outcomes in the LLIS, it is possible to extract a failure taxonomy that is well-suited to system analysis, and accounts for operational characteristics alongside design characteristics. Additionally, the use of NLP enables large numbers of projects to be analyzed, enabling the discovery of common threads between apparently distinct types of projects as well as relatively rare types of failures.

2.2 Natural Language Processing in Design and Failure Analysis

Text mining and natural language processing (NLP) have already been applied to various problems within design research. In particular, NLP has been applied to a meta-analysis of engineering design research [11], analyzing team performance [12], and analyzing the design tradespace [13, 14]. Text mining has also been used for knowledge discovery in design, including for extracting information about functions [15, 16] and for design innovation [17]. More specifically, NLP has been applied to accident and failure analysis. Product reliability has been assessed

through text mining of consumer reviews [18]. Zhang et al. performed accident analysis in the construction industry using natural language processing on accident investigation reports [19]. Such analyses can enable mitigation strategies [19]. Similarly, text mining has been applied in occupational accidents in the steel industry, specifically to create a text mining based predictive model using a fault tree and Bayesian network [20]. In general aviation, text mining has been applied to accident reports to find patterns and associations among accidents [21].

In this research, we extract relevant failure-related design knowledge from historical data of lessons learned at NASA. While other research has similarly leveraged text mining and NLP to learn about failures, this research specifically extracts a failure taxonomy with hierarchical properties, improving its generalizability. We additionally present an approach for identifying usable documents from a mixed-quality, non-specialized, unstructured database, which can be leveraged to improve results in other knowledge discovery research as well. Finally, by using a corpus of lessons learned and not simply accident reports, we can learn from situations that led to undesirable or suboptimal outcomes, even if they did not necessarily lead to an accident or loss. These reports are often especially detailed in issues during design and development, which is relevant to extracting a failure taxonomy that can be used during the system's design phase.

3 METHODOLOGY

The proposed methodology is divided into two parts. First, lessons containing meaningful failure-related information are identified using two different models – one using NLP and one using a lexicon. Second, the identified lessons are assembled into a corpus and are used to extract a failure taxonomy. This approach is applied to the LLIS, described next. The methodology is summarized in Fig. 1. A detailed explanation of each part of the methodology follows.

3.1 Database

The database of interest is the NASA Lessons Learned Information System (LLIS), which is available to the public [2]. For external users, it is possible to build a web scraper to extract the raw text from the database. The complete database consists of 2096 documents from different contributors from 1972 to present. Each document is comprised of semi-unstructured text, with sections that are used at the discretion of the contributor. The information and sections include: lesson number, submitter, title, organization, date lesson occurred, date of publication, abstract, lesson(s) learned, recommendation(s), driving event, evidence of recurrence control effectiveness, project/program relation, project/program phase, mission directorate(s), and topic(s). Since sections are optional, some lessons are completely filled, whereas others only contain an abstract with a brief explanation

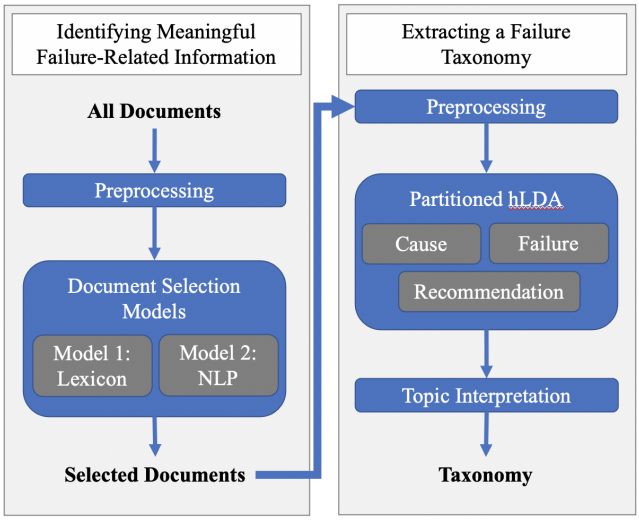

FIGURE 1: METHODOLOGY OVERVIEW. THE APPROACH IS DIVIDED INTO: (1) IDENTIFICATION OF MEANINGFUL FAILURE-RELATED INFORMATION AND (2) EXTRACTION OF A FAILURE TAXONOMY.

of the event. There is a wide variety of topics of documents, ranging from specific engineering failures, such as a critical weakness in a certain type of valve configuration, to agency relations with regulating organizations, such as the FAA.

3.2 Identifying Lessons Containing Meaningful Failure-Related Information

Prior to extracting information from each document, unusable documents must be filtered out. To this end, two supervised machine learning classification models are created and trained. After building these two models, their performance is compared and their classifications are used in tandem to most effectively remove unusable lessons. One model uses conventional natural language processing algorithms, while the other model is lexicon-based and inspired by sentiment analysis algorithms. The lexicon-based approach explicitly considers user-defined classification criteria as to which kinds of information indicate a usable document, while the natural language processing approach makes no assumptions about these rules and instead learns characteristics from a human-tagged training set of usable and unusable lessons. The NLP model is simpler to implement and is generalizable, while the lexicon-based model is built using lexicons relevant to specific classification criteria, including design engineering and failure information. The results from both models are compared and used for final document selection.

3.2.1 Classification Criteria Both models aim to filter documents in accordance with classification criteria devel-

oped by the researchers. Classification criteria are selected based on the desired scope and content of the taxonomy to be constructed using the identified lessons. However, in general, edge cases are included rather than omitted. The first classification model, Model 1, uses the criteria directly through lexicons as well as indirectly through the training data, which is labeled according to the rules. Model 2 learns the criteria through the training data. The classification criteria are outlined as follows:

1. The models shall exclude lessons that do not state or imply a hazard or failure. Although the research is focused primarily on failures that occur while the system is operating, development issues, such as significant delays, or accidents during maintenance or manufacturing are also of interest.
2. The selection models shall exclude lessons above the project level, such as lessons that are at a program or agency-wide level.
3. Lessons about metadata, or any lesson that is a compilation of other lessons, shall be excluded.
4. The models shall exclude project-agnostic lessons related solely to work culture or personnel, such as agency-wide human resources recommendations.

3.2.2 Preprocessing Preprocessing is performed on the data set prior to training and implementation of the models. The "topic(s)" section is used to filter out documents that contain only irrelevant topics, such as "business processes" and "contractor relationships". The combined "subject", "abstract", "driving event", and "lesson(s) learned" sections are used to determine the usability of the lesson. Standard text cleaning is performed, including tokenization, removal of punctuation and stop words, and lemmatization. Preprocessing improves the quality of the resultant topics and, subsequently, the resultant classification. Text preparation can be subdivided into the following steps:

1. *Clean text.* This includes the removal of punctuation, digits, and figure and reference labels.
2. *Tokenize text.* In this step, the document, which is initially a single string, is converted into a list of words or terms. For instance, "the o-ring failed" becomes "the", "o-ring", and "failed".
3. *Convert text to lowercase.* This step ensures that the words "Fail" and "fail" are considered as the same word.
4. *Lemmatize text.* Lemmatization reverts words into their root form, e.g. "failed" becomes "fail". This ensures that, in this example, "failed" and "fail" are not counted separately.
5. *Remove stop words.* Along with stop words common in most NLP applications, the names and locations of NASA centers are also added as stop words. The names of countries, months, and terms of measurement that occur in the database are also included as stop words.

The training and test set consists of a total of 398 human-tagged documents which are randomly selected from the database. The human taggers classified each document as "usable" or "unusable" based on the rules stated above. There is substantial or moderate agreement between the rater's initial classifications, with Cohen's $\kappa = 0.6150$ and Cronbach's $\alpha = 0.7635$ using scales of $[-1,1]$ and $[0,1]$ respectively. A consensus is reached for each document after the initial classifications. The two classes are imbalanced, with 325 documents in the usable category and only 73 documents in the unusable category. Of the training and test set, 298 documents are randomly selected to be used for training and testing for hyper parameter tuning of the models, as well as 5-fold cross-validation. The remaining 100 documents form a separate validation set used to compare performance across optimized models. For the final lesson selection, each model is trained on all 398 human-tagged documents prior to classifying the unlabeled data.

3.2.3 Model 1: Lexicon-Based Classification Model 1 is a random forest machine learning classification model from python's Scikit-learn package [22]. The optimized random forest model has a maximum tree depth of 4 and includes 300 estimators. Conventional sentiment analysis algorithms motivate the lexicon-based approach of this model as the failure terms in relevant documents are akin to negative terms in sentiment analysis. However, in this application the documents use highly specialized verbiage, and thus conventional sentiment analysis fails to capture the nuances between usable and unusable documents established in the classification criteria. Instead, Model 1 classifies documents using numeric scores calculated from lexicons based on the classification criteria outlined in Section 3.2.1. Feature inputs for the model consist of an engineering score, design score, management score, failure score, success score, positive sentiment, negative sentiment, neutral sentiment, and text length, which are utilized to output a classification of a document as usable or unusable. The failure, success, and simple sentiment analysis scores help support the implementation of criterion 1. The use of the engineering, design, and management scores support criteria 2 and 4. Custom lexicons relevant to design engineering are used for this purpose. The inclusion of text length supports criterion 3.

The lexicon-based approach is an adaptation of the sentiment analysis algorithms proposed by Pan et al. and Palanisamy et al. [18, 23]. Palanisamy et al. establish a lexicon-based sentiment analysis algorithm for detecting positive and negative product tweets by building both a general lexicon and a category specific lexicon, where the category specific lexicon includes a product catalog, a feature catalog, and category specific sentiment words, such as "overheated" and "glare" [23]. Pan et al. analyze and extract product failure information from reviews using a three-part failure dictionary constructed from general failure

terms, product specific failure terms, and corpus specific failure terms [18]. In this research, a failure lexicon is developed by gathering general synonyms for "failure" and "fault" from online dictionaries [24–26] and category-specific failure terms through examining LLIS documents [2]. A similar process is performed to generate a success lexicon, since the negation of a success term implies failure. A general engineering lexicon is created by aggregating the terms from online engineering glossaries into one list [27]. A design engineering lexicon is created from terms found on design-centric web pages, including Iowa State University's aerospace engineering page [28–30]. An additional lexicon of management terms is created from terms in the LLIS that are unusable due the selection rules, as determined by a human reader.

The inputs to the classification model are conventional sentiment scores, text length, and features scores calculated from the lexicons. Sentiwordnet [31] is used to calculate document-level scores for positive, negative, and neutral sentiment. Document scores are calculated for each lexicon category (engineering, failure, success, design, and management). For each document, an engineering, design, and management score is calculated as the percentage of terms in the document from the respective lexicon, illustrated in Eq. 1 where $w_i = 1$ if word i is in the lexicon, $w_i = 0$ if word i is not in the lexicon, and n is the number of words in the document.

$$score = \frac{\sum_{i=1}^{n} w_i}{n} \tag{1}$$

The failure and success scores are also calculated according to Eq. 1, where $w_i = 0$ if word i is not in the success or failure lexicon. If word i appears in the failure or success lexicon, then the previous three terms are checked for negation, down toner, and intensifier words. If one of the preceding words is an intensifier, such as "very", then $w_i = 1.5$. If one of the preceding words is a down toner, such as "hardly", then $w_i = 0.5$. If no intensifiers or down toners are present, then $w_i = 1$. If one of the preceding words is a negation, then the score is added to the opposite score, which is the success score if the word appears in the failure lexicon, or the failure score if the word appears in the success lexicon. The score is then divided by the total number of words in the document to obtain a percentage.

TABLE 1: PERFORMANCE OF LEXICON-BASED CLASSIFICATION MODELS ON THE VALIDATION SET.

	N. B.	Log. Reg.	SVC	R. F.	AdaBoost	XGBoost	K-Nearest
F1Score	0.459	0.470	0.577	0.833	0.691	0.739	0.438

Multiple classification models are examined after hyper parameter tuning, including Naïve Bayes, logistic regression, Support Vector Machine Classification (SVC), random forest, AdaBoost, XGBoost, and K-Nearest Neighbors. The performance of the classification models on the validation set is seen in Table 1. The Naive Bayes classification models use conditional probabilities learned during training to classify documents, and are less accurate in this application due to a lack of parameters [32]. SVC classifiers, which project the feature data into n-dimensional space in order to create boundaries for the classes, result in unsatisfactory accuracy, as did logistic regression. For this task the best SVC model uses a linear kernel and the logistic regression uses a library of linear models for a solver. The K-Nearest Neighbors model classifies this data best with 8 neighbors. Boosting algorithms, such as AdaBoost and XGBoost, are considered state-of-the art and have recently gained popularity due to their high accuracy across different applications [33, 34]. The AdaBoost model performed best with an ensemble of 180 decision tree models, each with a maximum depth of 7 and balanced class weight. For the XGBoost model, hyper parameter tuning optimized the model to contain 25 ensemble models with a maximum depth of 4. Random forests are increasingly used and well-regarded, specifically due to their ability to handle unbalanced data and over-fitting [34,35]. The random forest models classify documents through the use of a random collection of decision trees of varying depths and features. For the final classification Model 1, a random forest with 300 tree estimators of depth 4 is selected as it has the best performance with $F1 = 0.833$.

Overall, Model 1 performs well with some drawbacks. The process of assembling lexicons is tedious and does not necessarily generalize well to other databases. However, the use of context specific words and terms known to be important to this database provides some verification that the model is classifying documents in accordance to the classification criteria with high performance ratings. Since the lexicon based method in Model 1 requires some effort and is context specific, a more automatic and generalizable document classification model is built using a supervised natural language processing approach. This NLP model requires only a training set to learn the classes and the remainder of the implementation requires less human effort.

3.2.4 Model 2: NLP-Based Classification

Model 2 is developed using conventional natural language processing algorithms for document classification [36–38]. Using Scikit learn, SpaCy, and NLTK, [22, 39, 40] a pipeline is developed for the model. The first step in the pipeline is additional data cleaning, followed by Term Frequency Inverse Document Frequency (TF-IDF) vectorization, and finally the classifier. TF-IDF is commonly used for NLP tasks, such as classifying documents, and is chosen over the bag-of-words vectorization due TF-IDF's vector weighting based on importance. In TF-IDF, a word has a greater

weighting when it appears more frequently in a given document than when it occurs in the corpus of documents, thus words relevant to document topics, such as "valve", are weighted more heavily than common words, such as "there". The classifiers undergo the same testing as those for Model 1, including hyper parameter tuning using a training and test set, followed by performance comparison on the validation set. Of the SVC, random forest, and XGBoost models, the SVC model with a linear kernel and $C = 1.9$ performs best overall on the validation set, with $F1 = 0.833$. The random forest with 85 estimators of depth 2 scores $F1 = 0.637$, and XGBoost with 75 estimators of depth 7 scores $F1 = 0.707$. Class imbalances, in which one classification has significantly more data than the other, must be considered for optimal performance of the SVC model.

Class imbalances are handled in one of three ways. The first approach is over-sampling, which is performed by repeatedly sampling from the minority class until the classes are balanced in the training set. This improves the training and test performance of the model, but the validation performance is significantly lower, indicating high levels of over-fitting and poor generalizability. The next approach to handling imbalanced data is the implementation of a commonly used algorithm known as Synthetic Minority Oversampling Technique, or SMOTE [41]. SMOTE creates synthetic cases of the minority class by interpolating the data already present in the class. SMOTE is applied after the first two steps of the pipeline to create synthetic vectorized data. In this application, SMOTE performs no better than over-sampling. When over-sampling is applied prior to SMOTE, results improve; however, over-fitting still occurs. The final approach to overcome class imbalances utilizes a class weight parameter in Scikit-learn's SVC model. This optional parameter surmounts the issues of imbalanced data and model over-fitting by balancing the class weights. This parameter behaves similarly to oversampling algorithms [22], but the over-fitting is ameliorated.

Overall, Model 2 performs well and provides a more general approach to the problem of identifying usable documents in a database. The most time-consuming task needed for this approach is building a labeled training and test set. The document classifications from Model 2 are cross-checked against those of Model 1, resulting in a final set of usable documents that are composed into a corpus for extraction of a failure taxonomy.

3.3 Extraction of a Failure Taxonomy

The second part of the proposed methodology presents an approach for extracting a failure taxonomy from a loosely structured natural language database. First, the text is prepared for natural language processing. Then, a partitioned approach to hLDA is presented in which the documents are subdivided according to sections representing unique aspects of the failure taxonomy. Finally, a taxonomy is constructed using the identified topics. Standard text preparation procedures are performed, as in Section 3.2.2. In addition to these steps, a few additional preprocessing steps are added for the taxonomy extraction:

1. *Remove documents with incomplete entries.* Some authors of the lessons learned opted to complete some sections of the document, but not others. When the aspects utilized in the partitioned hLDA approach are incomplete, the lesson is excluded. These instances account for a relatively small segment of the corpus.
2. *Remove extremes.* This involves the removal of words that occur fewer than five times in the entire corpus or in more than 30% of documents in the corpus. This step has the effect of removing very common, low-information words as well as highly project-specific words.

3.3.1 A Partitioned Approach to hLDA Our approach to using hLDA to extract a taxonomy involves applying hLDA to three distinct sections of the documents, enabling the extraction of multiple aspects of a failure taxonomy. The corpus for each hLDA topic model is a compilation of sections of documents. The *hierarchical* nature of the topic model aims to improve generalizability, while the *partitioned* nature of the approach extracts disparate aspects from the documents. In this model, a topic refers to the topic of a partition, rather than the topic of an entire document. Each row of the failure taxonomy consists of three topics, one from each partition, correlating to one or more documents in the LLIS.

hLDA is implemented using the tomotopy package in python, which follows the approach outlined by Blei et al. [42] and is described in the remainder of this section. Topic modeling finds topics, which are defined as probability distributions of words that tend to co-occur, in a corpus. This differs slightly from clustering algorithms in that the same word can have significant influence in multiple topics, which is better suited to understanding natural language. In the LLIS, take for example words like "electrical", which are likely to be associated with multiple different topics. Topic models are defined by reverse engineering an imagined process that generates documents. This generative model produces documents from distributions over topics, which are distributions over words. Like other topic modeling methods, hLDA is a form of unsupervised machine learning. In contrast to standard LDA, however, hLDA organizes topics into a hierarchy, with more highly abstracted topics above more specific topics. The more highly abstracted topics do not simply share similar distributions with the lower level topics; instead, the generative model produces documents using a path from root to leaf and, therefore, more accurately reflects the notion of specificity in text [42]. While LDA requires a parameter dictating the number of topics, hLDA decides the number of topics independently and the number of levels in the hierarchy is a parameter. The hierarchy is produced using a nested Chinese Restaurant Process

(nCRP).

The Chinese Restaurant Process (CRP) imagines a scenario in which customers are seated at tables in a Chinese restaurant with an infinite number of tables. The first customer to enter the restaurant is seated at the first table. Subsequent customers are seated at tables according to a probability distribution based on the number of customers already seated at a table. Customers are more likely to be seated at tables with more customers than those with few or no customers. The probability of the mth customer c sitting an a table is given by Eq. 2, where m_i is the number of customers currently sitting at table i and γ is a parameter that influences how frequently a customer will choose to sit at an empty table as opposed to an occupied table and must be a real number [42].

$$p(i|c_1,...,c_m) = \begin{cases} \frac{m_i}{\gamma+m-1} & \text{if } i \text{ is occupied} \\ \frac{\gamma}{\gamma+m-1} & \text{if } i \text{ is unoccupied} \end{cases} \quad (2)$$

To elevate the CRP to the nCRP, there exists an infinite amount of Chinese Restaurants, each with infinite tables of infinite seats, where each table has a card that lists the name of another Chinese restaurant to which the customer will go next. Each restaurant's name uniquely appears on a single table in only one restaurant, resulting in an infinite tree. In hLDA, each document is analogous to a customer while each topic is analogous to a restaurant. The number of restaurants a customer visits, or the number of topics in a document's path, is the number of levels in the hierarchy. All documents contain the root topic, or the first restaurant, in their topic path. From the root topic, a document's path progresses to a level one topic, which is a restaurant whose name appears on a card at a table in the first restaurant. Subsequent level topics are drawn in the same way, where the nth topic is produced from the $n-1$th topic. In terms of the nCRP, the nth restaurant a customer visits comes from a card placed on a table at the $n-1$th restaurant the customer visits.

In the hLDA generative model, for each node k of the infinite tree, a topic distribution β_k is generated from the symmetric Dirichlet with prior η. The hyperparameter η is tuned such that the number of topics identified balances the objective of generalizable taxonomy entries, i.e., those that contain more than one lesson, and high coherence. The term weighting scheme used is TF-IDF. Documents are generated by first selecting a root to leaf path, C_d, for each document d, where the path is generated by using the infinite tree generated by $nCRP(\gamma)$ as a prior. Then, given a path C_d, the distribution over topic levels on the path, θ_d, is generated from a GEM distribution. Finally, words are generated using a mixture of topics along the designated path in the tree and mixing proportions θ. hLDA necessitates identifying a posterior given infinite possible combinations of topics and trees – this task requires a method for approximation. For this purpose, a Markov Chain Monte Carlo (MCMC) algorithm, specifically a Gibbs sampling approach, is used to sample from the posterior nCRP and associated topics. In particular, collapsed Gibbs sampling is used to speed convergence [43]. Given a corpus, this process results in a final finite tree, or hierarchy of topics, which is created for each partition separately in order to construct a taxonomy.

3.3.2 Taxonomy Construction Once the topic model has been constructed for each aspect of the corpus, the taxonomy can be assembled. The three aspects that are extracted from the partitioned hLDA approach correspond to disparate elements in the taxonomy: cause, failure, and recommendation. Since each topic model is hierarchical, there are two levels of each aspect. These three aspects are selected for inclusion in the failure taxonomy based on two factors: (1) information available in the LLIS and (2) conventions in other failure taxonomies.

In this case, we are constrained to extracting aspects described in one of the sections of the LLIS, and our method assumes that the original contributors of lessons in the LLIS interpreted the intent of the section similarly. In many of the sections, such as the Recommendation(s) section, there is apparently little room for misinterpretation. Other sections, such as Evidence of Recurrence Control Effectiveness, are frequently left blank by the contributors and/or are filled in with inconsistent or low-quality information. In contrast, three sections are frequently filled in and tend to contain high quality information: Lessons Learned, Driving Event, and Recommendation(s). These observations influenced the decision to use these three sections in the taxonomy. Content in the Lessons Learned section is interpreted as information about the cause of the failure, content in the Driving Event section is interpreted as information about the failure itself, and content in the Recommendation(s) section is interpreted as information about recommendations.

Cause, failure, and recommendation, or similar variants, are often referenced in other failure taxonomies and in FMEAs. For instance, FMEAs often include columns for "failure mode", "potential causes", and "action recommended". Failure modes are similarly enumerated in methods such as FFDM [44]. Many failure classification schemes include a cause in addition to a failure mode [45]. Moreover, the cause of failure is important for this taxonomy's utility in early design failure analysis because it is important to be able to identify vulnerabilities in the system being designed and link them to potential consequences. Similarly, the inclusion of recommendations in the taxonomy facilitates the correction of issues during design time, as possible. Therefore, these three aspects are chosen to balance the information available in the LLIS with conventions for failure taxonomies in the literature, along with the potential utility of the extracted taxonomy for early design failure assessment.

For a sample of the extracted taxonomy, an interpretation

step is added to improve its human readability and usefulness. Raw topics are extracted in the form of lists of words, which may or may not appear in a logical order. Within the scope of this research, it is infeasible to interpret each topic within the entire taxonomy due to its size, and will be left to future work. Interpretation is performed by first selecting the most representative document for a particular row of the failure taxonomy. This is performed automatically according to the topic distribution of the document. Then, a human reader reviews the best matching document alongside the extracted taxonomy entry and associated lessons. A brief topic description is then generated by the human reader. This topic description is used in the final taxonomy.

4 RESULTS

The performance of the document selection procedure is presented before the taxonomy results. The two document selection models are combined in order to select the final documents for the taxonomy extraction. Topics identified using the partitioned hLDA approach are reported as an assembled taxonomy.

4.1 Document Selection

As seen in Table 2, both models exhibit satisfactory performance. Model 1 performs slightly better than Model 2 on 5-fold cross-validation performed on 298 documents, but the two models perform equally across all scores on the 100-document validation set. Due to the imbalance in the classes of usable and unusable, F-1, precision, and recall scores provide a better indication of model performance than accuracy. Since accuracy is the ratio of correct classifications to total classifications, accuracy is inflated when class imbalances exist and a classifier is better at identifying the majority class. Instead, precision, recall, and F-1 take into account the proportion of false positives and false negatives to true positives and true negatives in each class individually, and thus provide a more accurate representation of performance across classes with imbalances. These measures are calculated using macro average.

TABLE 2: DOCUMENT CLASSIFICATION MODEL PERFORMANCE METRICS.

Model	Method	Accuracy	F-1	Precision	Recall
Model 1	5-Fold CV	0.876102	0.777789	0.839764	0.745833
	Validation Set	0.919192	0.833613	0.855546	0.815476
Model 2	5-Fold CV	0.852881	0.708227	0.798149	0.67977
	Validation Set	0.919192	0.833613	0.855546	0.815476

The divergence in performance between the models is most evident in the F-1 and recall scores, with Model 1 obtaining scores of approximately 0.778 and 0.746 and Model 2 obtaining scores of approximately 0.708 and 0.680 respectively. The precision scores are more similar, with Model 1 scoring approximately 0.840 and Model 2 scoring 0.798. The smallest difference between the two is with the accuracy scores, where Model 1 is 0.876 and Model 2 is approximately 0.853. The two models have substantial or moderate agreement on the validation set similar to the agreement in human coders, with Cohen's $\kappa = 0.646$ and Cronbach's $\alpha = 0.785$ between models. The two models agree on 92% of the validation set.

TABLE 3: CONFUSION MATRIX FOR BOTH MODELS. THREE FALSE NEGATIVES INDICATE THE MODELS RARELY REJECT USABLE DOCUMENTS.

	Predicted Unusable	Predicted Usable
Actual Unusable	10	5
Actual Usable	3	81

The confusion matrix from the 100-document validation set, which is the same for both models, is displayed in Table 3. As evident in the matrix, there are five false positives and three false negatives, indicating both models are less accurate when identifying unusable documents. Both models wrongly classify one-third of unusable documents, which may influence the results for taxonomy extraction by including non-failure related information. For improved quality of results, only those documents that are classified as usable by both models are included in the final set. This results in a set of 1709 lessons learned documents. Of those documents, 97.5% of them have the "lessons learned", "recommendation(s)", and "driving event" sections filled out, which are used to extract failure related information. After the removal of the documents with missing or uninformative sections, such as "see lesson(s) learned", the final set contains 1639 usable documents, representing 78% of the LLIS.

Overall the models perform very similarly, and either method or a combination of the two can be used for acceptable performance when classifying documents. Both models have benefits and drawbacks, but Model 2 requires less effort to implement since it does not require a lexicon. Model 1 can be used as a confirmation of the quality of the NLP algorithm used in Model 2. The creation of lexicons of relevant words from the specific context for Model 1, and the similar performance between the two models implies Model 2 bases its classification on criteria and words that are relevant to the domain. The implementation of document selection models increase quality of the extracted taxonomy.

4.2 Taxonomy Extraction

Coherence scores, which measure the quality of topics, are reported in Table 4. The coherence metric reported is C_V, which

falls between 0 and 1 with 1 being perfect coherence. C_V has been found to be an effective metric of coherence compared to other options [46]. Coherence is measured for each topic individually. Both the mean of all topics and the standard deviation are reported for all topic levels and all aspects. As evident in Table 4, topics from all sections have high coherence, and specifically the failure aspect has the highest with $C_V = 0.886576$, followed by recommendations at $C_V = 0.820014$, and causes at $C_V = 0.815733$. The standard deviations for the sections are relatively low and consistent across aspects, indicating there is low variation in topic quality. Since recommendations have the greatest standard deviation, topics from this section may have a greater range in quality than in other sections. This finding matches the observation that the overall quality of the recommendations section in the LLIS is more inconsistent than the other two sections. Level 2 topics in all sections have a higher coherence and lower standard deviation in some sections, suggesting that more granular topics tend to be of higher and more consistent quality.

TABLE 4: TOPIC COHERENCE MEANS AND STANDARD DEVIATIONS FOR EACH LEVEL, MEASURED WITH C_V.

Coherence C_V	Topic Level	Cause	Failure	Recommendation
Mean	Level 1	0.787823	0.864312	0.809440
	Level 2	0.812625	0.899817	0.826216
	All Levels	0.804666	0.891331	0.819179
Standard Deviation	Level 1	0.050215	0.055635	0.048762
	Level 2	0.050385	0.028244	0.048996
	All Levels	0.056530	0.045469	0.054666

Figure 2 illustrates a small segment of the recommendations hLDA tree structure. The root node corresponds to a level 0 topic, the two nodes branching from the root node correspond to level 1 topics, and the remaining nodes correspond to level 2 topics. The increasing degree of specificity is apparent by comparing the top words from each topic level. For instance, the top level topic includes general words such as "system" whereas the level 2 topics include significantly more specific terms such as "high pressure". These nodes contain raw topic outputs, and have not yet been human-interpreted for readability. As discussed in Section 3.3.1, each node is not simply a summary of its branches; instead, as is convention in hLDA, the tree structure follows a more semantically coherent formalism. For a full understanding of a node, the reader must review all subsequent nodes in its path as well. For instance, to understand a level 2 node, knowledge of the level 1 and level 0 nodes along its path is also required. This property ensures the hierarchy captures abstraction in a similar manner as it is captured in natural language. For instance, in the

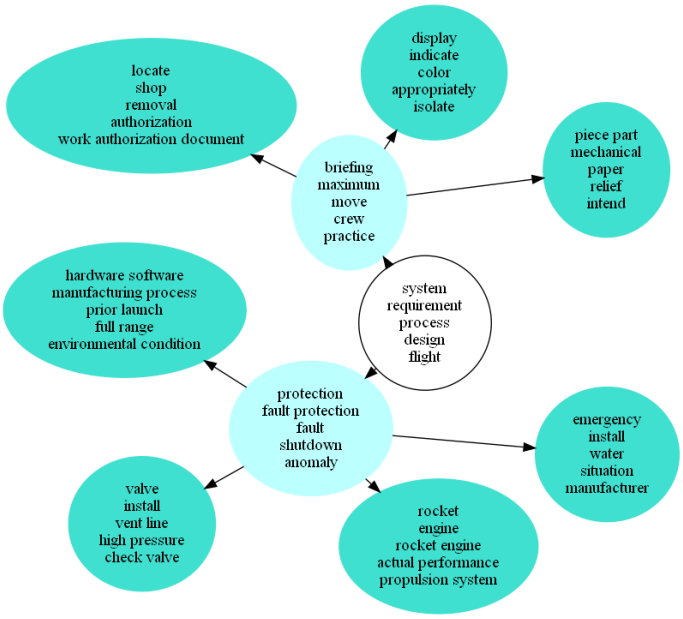

FIGURE 2: A SMALL SEGMENT OF THE RECOMMENDATIONS HLDA TREE STRUCTURE. NODES CONTAIN RAW TOPICS.

selected segment, the path from the top level node, to the level 1 node with "protection", to the level 2 node with "rocket" implies a recommendation relating to improved fault protection for a propulsion system. A more precise recommendation can be formed by analyzing this path alongside the best matching lesson for each topic. For instance, along this same path, which describes lesson number 14201 in the LLIS, the recommendation specifies ensuring that subcontractors can "meet all requirements", corresponding to the level 0 topic. Consistent with the level 1 topic, the lesson recommends determining the root cause of failures and applying controls to prevent those failures. Finally, consistent with the level 2 topic, the lesson specifies that the issue lies with the propulsion system design.

The partitioned hLDA approach yields ninety-six cause topics, ninety-five failure topics, and ninety-three recommendation topics. These figures include all three hierarchical levels, including the top level topic. A subset of the extracted taxonomy is human-interpreted and presented in Table 5. Taxonomy rows containing more than one lesson are selected in order to showcase the most generalizable results. Additionally, level 1 causes associated with large numbers of taxonomy rows are excluded due to space limitations. For failures with multiple causes, only the first listed cause is selected, again due to space limitations. The selected taxonomy rows are not selected based on whether the represented topics are best matching documents and therefore are likely to be representative of the degree of matching between listed lessons and their corresponding taxonomy entries. While interpreting the taxonomy, it is observed that there is some vari-

TABLE 5: A SELECTION OF THE FAILURE TAXONOMY EXTRACTED FROM NASA LLIS.

Cause		Failure		Recommendation		Lessons
Difficult to detect hazards	Defects are not detected until late in design process	Not following reliability best practices for design and test	Failures exposed through sinusoidal vibration	Simulate worst case performance	Voltage and Temperature Margin Testing (VTMT)	894, 779
	Potential hazards not evaluated prior to storage or test	Safety mechanism not secured	Engine shutdown mechanism	Detect anomalies and prevent hard shutdowns	Validate behavior using existing tools	1622, 1395
		Problems with parts in an assembly	Telemetry parts	Improved communication of procedures	Controls for interfaces and interactions	18701, 119
Unexpected operating conditions	Lack of real time monitoring or complete life cycle assessment	Unable to operate or access airlock, hatch, etc.	External effect leads to loss of functionality	Simulate worst case performance	Voltage and Temperature Margin Testing (VTMT)	1217, 269
		Lack of management oversight or approval	Interface or configuration issues	Improved communication of procedures	Controls for interfaces and interactions	26303, 4977, 1713
			Team scheduling or coordination	Improved communication of procedures	Controls for interfaces and interactions	2216, 1490
			Lack of precautions for planned operating environment	Redesign mechanism for actual operating conditions	Confirm appropriate usage	3339, 11
		Problems with parts in an assembly	Telemetry parts	Verify operating conditions	Assure adequate protection from environment	2298, 404
		Operation outside of operating or testing envelope	Off-nominal operating conditions	Improved communication of procedures	Implement audio warning systems	12101, 10401
			Lack of real-time instructions	Improved communication of procedures	Controls for interfaces and interactions	1264, 270
		Understanding of system state hindered	Visibly hidden hazard	Detect anomalies and prevent hard shutdowns	Validate behavior using existing hardware and software tools	1290, 1206
Hardware parts associated with problems	High current transients from switching of power components	Not following reliability best practices for design and test	Redesign necessitates changes in power supply	Simulate worst case performance	Redesign of power supply	1617, 1609

ation in the degree of matching between extracted topic and associated lessons. For instance, the first cause, related to defects that are difficult to detect, is clearly identified in the associated lessons, whereas the relation to failure of telemetry parts is not as clear in the associated lessons. This is expected behavior as the topic model is an abstraction.

5 DISCUSSION

Both models used for document selection are consistent with expected performance for these algorithms, especially when both models are used. There is, however, potential for improvement in recognizing unusable documents in particular. Such a method is highly desirable as a precursor to extracting a failure taxonomy because the quality of the information extracted for the taxonomy can only be as informative as the corpus allows, and the time-saving benefit of automating document classification outweighs the inclusion of documents incorrectly labeled as usable. The use of the machine learning classification models allows for document selection at a much larger scale than would be possible if human tagging of the entire database were required. Without the document classification step, the extracted topics would likely include irrelevant topics which would require removal.

The taxonomy extraction method assumes that the original contributors interpreted the intent of each of these sections similarly such that the resultant topics match the expected aspects. Qualitatively, the extracted taxonomy exhibits topics that align with the expected aspect reasonably well. Thus, it is likely that even if a portion of the contributors did not interpret a specific section similarly to others, this effect is insignificant. There are a few additional aspects that would be useful to include in the failure taxonomy, including the effect of failure. The effect of the failure is not consistently differentiated from the failure itself since it is not described in its own section. A different database is required to extract this information using the proposed approach. Finally, the data-based taxonomy extraction method assumes no particular conceptual model describing failures generally. This means that the discovered taxonomy is dependent on the dataset

and may or may not be "complete". However, this means that failures that do not fit a certain conceptual model can also be captured – a key benefit of a data-based approach and one that is well-suited to capturing unanticipated behaviors.

6 CONCLUSIONS AND FUTURE WORK

In this research, we present an approach to (1) identifying historical data in natural language format that is usable for a custom application and, (2) extracting a failure taxonomy from the identified usable documents. The overall framework can be used for identifying and extracting knowledge for other design problems in addition to the failure taxonomy presented here. The method of extracting various aspects from a semi-structured natural language source is extensible to other databases. Further, the extracted failure taxonomy itself provides a classification of causes, failures, and recommendations from lessons learned in NASA projects. This information will be useful to designers attempting to prevent the recurrence of these failures and can be incorporated into early design failure analysis techniques.

Future research will focus on extending the extracted failure taxonomy to create an assistive design tool for prompting designers building, for instance, bow-tie diagrams or performing early design failure analysis. Many such techniques are inherently limited by the imagination of the designer; an assistive tool can be implemented to suggest likely failure modes based on design characteristics. This will be accomplished using a machine learning approach such as a neural network to predict likely failure modes based on the failure taxonomy extracted in this research. This front-end work of extracting the taxonomy is necessary to the development of such a tool. In addition, the overall approach for identifying and extracting relevant information from a repository of text-based data will be applied to other databases to discover knowledge in other domains.

ACKNOWLEDGMENT

This research is supported by the System-Wide Safety (SWS) project in the Airspace Operations & Safety program (AOSP) in the NASA Aeronautics Research & Mission Directorate (ARMD). Any opinions or findings of this work are the responsibility of the authors and do not necessarily reflect the views of the sponsors or collaborators. The authors would also like to thank Guillaume Brat for his input.

REFERENCES

[1] Oregon State University, 2020. Design Repository. `https://design.engr.oregonstate.edu/repo`.

[2] NASA, 2020. NASA Public Lessons Learned Information System. `https://llis.nasa.gov/`.

[3] O'Halloran, B., Stone, R., and Tumer, I., 2012. "A failure modes and mechanisms naming taxonomy". In Reliability and Maintainability Symposium (RAMS), 2012 Proceedings, pp. 1–6.

[4] Jordan, T., Bender, B., Herzog, M., and Song, Y.-W., 2019. "A model-based approach to identify barriers in design knowledge reuse". In Proceedings of the Design Society: International Conference on Engineering Design, Vol. 1, pp. 2427–2436.

[5] Fan, I.-S., Li, G., Lagos-Hernandez, M., Bermell-Garcia, P., and Twelves, M., 2002. "A rule level knowledge management system for knowledge based engineering applications". In Proceedings of the ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 1: 22nd Computers and Information in Engineering Conference, Vol. 1.

[6] Begoli, E., and Horey, J., 2012. "Design principles for effective knowledge discovery from big data". In 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 215–218.

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

[8] Tumer, I., Stone, R., and Bell, D., 2003. "Requirements for a failure mode taxonomy for use in conceptual design". In International Conference on Engineering Design (ICED 03).

[9] Van Wie, M., Bohm, M., Barrientos, F., Turner, I., and Stone, R., 2005. Learning from failures: Archiving and designing with failure and risk. Tech. Rep. NASA-TM20050157895.

[10] Stone, R., and Wood, K., 2000. "Development of a functional basis for design". *Journal of Mechanical Design,* ***122***, 12.

[11] Chiarello, F., Melluso, N., Bonaccorsi, A., and Fantoni, G., 2019. "A text mining based map of engineering design: Topics and their trajectories over time". *Proceedings of the Design Society: International Conference on Engineering Design,* ***1***, 07, pp. 2765–2774.

[12] Dong, A., Hill, A., and Agogino, A., 2004. "Document analysis as a means for predicting design team performance". *Journal of Mechanical Design,* ***126***, 01.

[13] Bang, H., and Selva, D., 2016. "iFEED: Interactive feature extraction for engineering design". In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. V007T06A037.

[14] Hitomi, N., Bang, H., and Selva, D., 2017. "Extracting and applying knowledge with adaptive knowledge-driven opti-

mization to architect an earth observing satellite system". In AIAA Infotech@Aerospace Conference.

[15] Cheong, H., Li, W., Cheung, A., Nogueira, A., and Iorio, F., 2015. "Automatic extraction of function knowledge from text". In ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

[16] Cheong, H., Li, W., Cheung, A., Nogueira, A., and Iorio, F., 2017. "Automated extraction of function knowledge from text". *Journal of Mechanical Design,* ***139***, 09.

[17] Shi, F., 2018. "A data-driven text mining and semantic network analysis for design information retrieval". PhD thesis, Imperial College of London, Dyson School of Design Engineering, July.

[18] Pan, X., Wang, H., You, W., Zhang, M., and Yang, Y., 2020. "Assessing the reliability of electronic products using customer knowledge discovery". *Reliability Engineering & System Safety,* ***199***, p. 106925.

[19] Zhang, F., Fleyeh, H., Wang, X., and Lu, M., 2019. "Construction site accident analysis using text mining and natural language processing techniques". *Automation in Construction,* ***99***, 01, pp. 238–248.

[20] Sarkar, S., Vinay, S., and Maiti, J., 2016. "Text mining based safety risk assessment and prediction of occupational accidents in a steel plant". In 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), pp. 439–444.

[21] Vijayanarayanan, A., and Li, H., 2013. An evaluation of AIRES and STATISTICA text mining tools as applied to general aviation accidents. Tech. Rep. DOT/FAA/TC-TN13/7, June.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011. "Scikit-learn: Machine learning in Python". *Journal of Machine Learning Research,* ***12***, pp. 2825–2830.

[23] Palanisamy, P., Yadav, V., and Elchuri, H., 2013. "Serendio: Simple and practical lexicon based approach to sentiment analysis". In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), Association for Computational Linguistics, pp. 543–548.

[24] dictionary.com, LLC, Accessed: November-2020. thesaurus.com. `https://www.thesaurus.com/`.

[25] Oxford University Press, Accessed: November-2020. Lexico. `https://www.lexico.com`.

[26] Reverse Dictionary, Accessed: November-2020. `https://reversedictionary.org/`.

[27] Wikipedia contributors, 2021. Glossary of engineering — Wikipedia, the free encyclopedia. [Online; accessed November-2020].

[28] Iowa State University of Science and Technology, Department of Aerospace Engineering, Accessed: November-2020. Design engineering glossary.

[29] Design 1st, Accessed: November-2020. Glossary of product design terms.

[30] Zalaco, Accessed: November-2020. Guide to mechanical design and engineering terminology.

[31] Baccianella, S., Esuli, A., and Sebastiani, F., 2010. "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.". Vol. 10.

[32] Jadhav, S. D., and Channe, H. P., 2016. "Comparative study of k-nn naive bayes and decision tree classification techniques". *International Journal of Science and Research (IJSR),* ***5***(1), pp. 1842–1845.

[33] Chen, T., and Guestrin, C., 2016. "XGBoost: A scalable tree boosting system". In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, p. 785–794.

[34] A. J. Wyner, M. Olson, J. B., and Mease, D., 2017. "Explaining the success of adaboost and random forests as interpolating classifiers". *Journal of Machine Learning Research,* ***18***(1), pp. 1558–1590.

[35] Breiman, L., 2001. "Random forests". *Machine Learning,* ***45***, p. 5–32.

[36] Oza, N., Castle, J. P., and Stutz, J., 2009. "Classification of aeronautics system health and safety documents". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* ***39***(6), pp. 670–680.

[37] Aninditya, A., Hasibuan, M. A., and Sutoyo, E., 2019. "Text mining approach using TF-IDF and naive bayes for classification of exam questions based on cognitive level of bloom's taxonomy". In 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), pp. 112–117.

[38] Goswami, S., and Raychaudhuri, D., 2020. "Identification of disaster-related tweets using natural language processing". In International Conference on Recent Trends in Artificial Intelligence, IOT, Smart Cities & Applications (ICAISC-2020).

[39] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A., 2020. spaCy: Industrial-strength Natural Language Processing in Python.

[40] Loper, E., and Bird, S., 2002. "Nltk: The natural language toolkit". In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02, Association for Computational Linguistics, p. 63–70.

[41] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., 2002. "Smote: Synthetic minority over-sampling

technique". *Journal of Artificial Intelligence Research,* ***16***(1), June, p. 321–357.

[42] Blei, D., Griffiths, T., Jordan, M., and Tenenbaum, J., 2004. "Hierarchical topic models and the nested chinese restaurant process". *Advances in Neural Information Processing Systems,* ***16***, 05.

[43] Liu, J. S., 1994. "The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem". *Journal of the American Statistical Association,* ***89***(427), pp. 958–966.

[44] Stone, R., Tumer, I., and Wie, M., 2005. "The function-failure design method". *Journal of Mechanical Design,* ***127***, pp. 397–407.

[45] Uder, S., Stone, R., and Tumer, I., 2004. "Failure analysis in subsystem design for space missions". In Proceedings of DETC '04 2004 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 3.

[46] Röder, M., Both, A., and Hinneburg, A., 2015. "Exploring the space of topic coherence measures". In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM '15, Association for Computing Machinery, p. 399–408.