Advanced Text Analytics and Machine Learning Approach for Document Classification

Chaitanya Anne
chaitanya.anne@hotmail.com

Advanced Text Analytics and Machine Learning Approach for Document Classification

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
In partial fulfillment of the
Requirements for the degree of

Master of Science
in
Computer Science

by

Chaitanya Anne

Bachelor of Technology, JNTUK, 2014

May, 2017

Acknowledgement

I would like to express my sincere gratitude to my supervisor Dr. Md Tamjidul Hoque for his guidance and support throughout my dissertation research. His continual faith and encouragement made me finish my work with ease. He was always very patient and supportive throughout my endeavor.

I am very grateful to my co-supervisor Dr. Shengru Tu for his continued support and guidance. He steered me in the right direction whenever I needed his guidelines. Without their passionate participation and input, this work could not have been completed so easily.

My special thanks to Dr. Christopher M. Summa, for providing his expertise to serve as graduate committee for my thesis defense. I am gratefully indebted for his very valuable comments on this thesis.

I would also like to acknowledge Hoque-lab member Mr. Avdesh Mishra (PhD candidate) who continuously supported me throughout the tenure of this project. I am very thankful to lab member Ms. Sumaiya Iqbal (PhD candidate) for her guidance in my thesis writing.

My profound gratitude to my family for providing me with continuous encouragement and unfailing support throughout my years of study and through the process of research and write up of this thesis.

Table of Contents

List of Tables

List of Figures

Abstract

Text classification is used in information extraction and retrieval from a given text, and text classification has been considered as an important step to manage a vast number of records given in digital form that is far-reaching and expanding. This thesis addresses patent document classification problem into fifteen different categories or classes, where some classes overlap with other classes for practical reasons. For the development of the classification model using machine learning techniques, useful features have been extracted from the given documents. The features are used to classify patent document as well as to generate useful tag-words. The overall objective of this work is to systematize NASA's patent management, by developing a set of automated tools that can assist NASA to manage and market its portfolio of intellectual properties (IP), and to enable easier discovery of relevant IP by users. We have identified an array of methods that can be applied such as k-Nearest Neighbors (kNN), two variations of the Support Vector Machine (SVM) algorithms, and two tree based classification algorithms: Random Forest and J48. The major research steps in this work consist of filtering techniques for variable selection, information gain and feature correlation analysis, and training and testing potential models using effective classifiers. Further, the obstacles associated with the imbalanced data were mitigated by adding synthetic data wherever appropriate, which resulted in a superior SVM classifier based model.

KEY WORDS

Machine learning algorithms, Patent classification, imbalanced data, SVM, Multi-class classification.

1. Introduction

Document classification is the process of classifying a document into a predefined category. It plays a vital role in managing large number of text documents by automating the document classification task. The two main approaches in machine learning for document classification are supervised learning [1], where the model is trained with labeled documents, and unsupervised learning, where the predefined category or training data is not available for the classification task, rather clusters are made to observe the natural grouping of the documents. Further, the given document in supervised approach can be labeled as a single-class document as well as multi-class documents. Multi-class labeled document classification is relatively challenging compared to the single-class labeled document [2].

Supervised learning models are widely applicable because often the possible categories are predetermined based on business activities. Document classification mainly consists of document representation (vector form), feature selection, feature extraction, application of machine learning algorithm on the data for evaluation.

In this study we performed classification task on NASA's patent documents, available in html format and downloaded from the U.S. Patent Office [3]. We applied five machine learning methods k- Nearest Neighbors (kNN), two variations of Support Vector Machine (RBF and PolyKernel), and two tree- based classification algorithms (Random Forest and J48), to carry out the initial classification on NASA patent data. We have assessed the accuracies of the initial classification with 5 fold cross- validation. Based on the results, we conclude that the SVM performed well on the available patent data compared to other algorithms.

Along with the classification of top level categories, we have proposed a way to further proceed with second level of classification of a document in the same category i.e., providing a sub category to which top category the document belongs. For example, a document, which is classified as "Aeronautics", can be further classified into "Aircraft" or "Control" depending on its content. Thus, the sub category classification enhances user's visual discovery of intellectual property content of the documents.

Tag words can be generated from the training files as well as from the input files based on the information gain [4] of the useful word or term calculated from the given document. The generation of these tag words will provide further fine-grained details about patents combining the subcategories information, which will assist in search activities and document management.

To perform the entire document classification and management task, we have developed the proposed tool in Java. This tool uses the training data samples that are provided, builds model and then predicts the category of the input document. Along with the classification, probability distribution of the document could be viewed in pie chart by utilizing the probability values which is generated by the classifier. This helps user predict multi-labeled document and enable to place the document in the most appropriate category.

One of the challenges in this project was the existence of the imbalanced datasets. Imbalance in training dataset implies significant difference in the number of samples present in the classes. Imbalance dataset is problematic, because the total error of the majority-class biases the decision boundary in favor of the majority class by almost masking the minority-class [5]. In such cases the classification would tend to over adapt the classes with high number of samples ignoring the smaller classes [6]. To overcome the problem of document classification involving imbalanced and low training dataset, we have implemented a solution by adding synthetic data to the minority class.

The synthetic data has been collected from well- known [7] taxonomical repositories and articles, and fed into the respective classes by generating files having chunk of dataset within them. In this paper we analyze the accuracy of SVM classifier with data imbalance problem in classes and also describe how synthetic data addition to the training data set have improved the accuracy of the classifier.

2. Literature Review

In this section, a relevant literature related to the proposed research work has been discussed.

2.1 Review of multi-class document classification

In a pioneering approach, Fabrizio Sebastiani mechanized automated classification (or classification) of texts into predefined classes, this approach had seen an expanding enthusiasm in recent years, because of the expanded accessibility of documents in digital form and the subsequent need to classify them [8]. The predominant way to deal with these text-classification problems are popularly and manageably done using machine learning techniques, a general inductive process that naturally builds a text classifier by learning from a set of labeled text documents, the attributes of the categories. The benefits of this approach over the knowledge engineering approach (manual classification and annotation by domain specialists) are a decent viability, and direct movability to various domains. The success story of automated text classification is expanding to neighboring fields of application. For example, noisy text classification resulting from the optimal character recognition and speech transcripts are available and inspiring [8].

Text classification has advanced from a minor research field in late '80s, into a completely bloomed investigation field, which has produces proficient, powerful, and generally useful classifications with many application areas. This achievement has been based on two aspects of modernization: *first*, the continually expanding inclusion of the machine learning group in text order, which has of late brought about the utilization of the latest machine learning innovation in text classification applications, and *second*, the accessibility of standard benchmarks, (for example, Reuters-21578 and OHSUMED) [9], which has supported research by giving a setting in which distinct research endeavors could be contrasted with each other, and in which the best techniques and calculations could emerge.

2.2 Review of approaches in document classification

Currently, text classification has become one of the key methods to handle and organize text data [10]. Documents, which generally contains strings of characters, have to be transformed into a suitable representation that usable in learning algorithms and classification problems. Information retrieval research suggests that word stems work very well as representation units leading to attribute value representation of text, which is used in implementing this project. The word stem is derived from the form of a word by removing case and derived information [11]. For example “machine”, “machining” are all mapped to the same stem “machin”. This leads to an attribute value representation of the text data. For each unique word, its count ω_i in the data corresponds to a feature as term frequency (ω_i, d), in corresponding text document d. To overcome the unnecessarily large feature vectors, words are represented as features if they have occurred in the training data set usually at least 3 times. Based on this representation, scaling the dimensions of the feature vector with their respective *inverse document frequency* (IDF, which is applied as the log inverse of ω_i) leads to an improved performance. IDF is calculated from the total number of training documents (n) and the document frequency of the particular word ω_i as shown in equation (1):

$$IDF\ (\omega_i) = log\left(\frac{n}{DF(\omega_i)}\right) \tag{1}$$

Here, $DF(\omega_i)$ is the number of those documents in the collection, which contain the term ω_i. Based on the standard feature vector representation of the text data, it was argued in [3] that the support vector machines are more appropriate for this type of setting. Different classification methods such as Bayes, SVM, C4.5 and kNN were applied on the Reuters-21578 and Ohsumed corpus [2] among which, SVM was found to have superior prediction with considerable performance gains.

Rennie Jason [12] in his work indicated that, SVMs performed better than Naïve Bayes for the multiclass text classification problems. 20 Newsgroups is a data set collected and originally used for text classification by [13]. Datasets from 20 Newsgroups and Industry-Sector [14, 15] were used for the experimental setup. Empty documents that are generated after preprocessing (23 for 20 Newsgroups, 16 for Industry Sector) were filtered out. The code and pre-processed data used in experiments can be obtained from [16, 17]. It contained 19,997 documents evenly distributed across 20 classes. All headers, UU-encoded blocks, stop words and words that occur only once were removed from further consideration. The vocabulary resulted into 62,061 words. Randomly selected 80% of documents per class for training and the remaining 20% for testing. This is the same pre-processing and splitting as McCallum and Nigam used in their 20 Newsgroups experiments [14]. Later, the experiments were conducted on 10 fold cross-validation (FCV) [1, 18]. To gauge performance for different amounts of training documents, the authors created nested training sets of 800, 250, 100 and 30 documents per class for 20 Newsgroups and (up to) 52, 20, 10 and 3 documents/class for Industry Sector. Empirical calculation of error for different sizes of data sets and their classification using SVMs and Naïve Bayes are also provided. Based on the experimental results, the support vector's improved ability in performing the binary classification have given lower errors compared to Naïve Bayes.

As indicated in [19], machine learning for text classification is best available tool for document classification, news filtering, document routing, etc. In text classification, effective feature selection is very important to make the learner model effective and more precise. Commonly known metrics such as *Chi Squared*, *Bi-normal Separation*, *Information Gain*, *Probability Ratio*, etc. have been applied on the data to extract the features. *Information Gain* (IG) measures the decrement in entropy when the feature is given versus the feature is absent.

Information gain (IG): In Machine learning field Information gain is commonly employed as a term-goodness criterion [20]. By knowing the presence and absence of a term in a document, IG measures the number of bits of information obtained for the category prediction. Let $\{c_i\}ss_{i=1}^{m}$ denote the set of categories in target space, and the information gain of a term t is calculated as

$$G(t) = -\sum_{i=1}^{m} P_r(c_i) log P_r(c_i) + P_r(t) \sum_{i=1}^{m} P_r(c_i\,/t) \log P_r\left(\frac{c_i}{t}\right)$$
$$+ P_r(\bar{t}) \sum_{i=1}^{m} P_r(c_i\,/\bar{t}) \log P_r(c_i/\bar{t}) \quad (2)$$

Here, *m* is the target space dimension, and P_r is the probability of occurrence of term t in document corpus. In a given training corpus, the information gain is computed for each unique term and the terms were removed from the attribute space if the information gain is less than the predetermined threshold value. In [20] the authors performed evaluation of feature selection techniques like IG, Chi-squared, Document frequency thresholding, and Mutual information [20] using Reuters [21] and Ohsumed [9] data and found Chi-squared and IG were more effective in aggressive term removal without losing categorization accuracy.

The work [19] introduced an observational examination of twelve-element choice techniques (i.e., Information Gain) assessed on a benchmark of 229 text classification issue occasions that were assembled from Reuters, TREC, OHSUMED [9], and so forth. The outcomes are investigated from numerous objective point of view such as exactness, F-measure, accuracy, and review since each is suitable in various circumstances. The overall feature selection procedure is to measure each potential feature according to available metrics like Chi Squared, Bi-normal Separation, Information Gain, and Probability Ratio and to consider the best *k*-features. Scoring the features involves counting the occurrences of the feature in positive and negative class in a training datasets separately and to compute the feature selection metrics. In this proposed research of patent classification, we are motivated to investigate and implement the *IG* based approach because of its proven effectiveness [19].

Text classification has been considered a much needed technique to oversee and handle an nearly-unlimited amount of documents in digital forms that are far reaching and ceaselessly expanding [22]. A document can be represented as an array of words, however, not all words in a document can be used for training the classifier, and those words to be ignored are called *stop words*. For example auxiliary verbs, conjunctions, articles can be classified into stop words. In a preprocessing task, stop words are often removed from consideration.

Stemming is another common task in preprocessing step. A stemmer algorithm replaces the words with the same stem with their respective stem-word. The removal reduces unrealistic feature variations, increases frequency count for the important feature and reduces the input-features dimension. This reduction of the curse of dimensionality [1] yields in improved classification accuracy.

Feature transformation differs significantly from feature selection approaches however, like feature selection, its main purpose lies in reduction of feature set size. Principal component analysis (PCA) is a famous approach used for the feature transformation [23]. The aim of the usage of PCA is to generate a discriminative transformation matrix to reduce the initial feature space into a lower dimensional feature space which reduces the complexity of the classification without any (zero to a little) trade off in accuracy as a choice. Feature extraction or feature selection techniques like term frequency, inverse document frequency, information gain described in this paper are followed in the preparation of our data.

As indicated by John Platt [24], in his new algorithm for the training of support vector machines: Sequential Minimal Optimization, or SMO, training a support vector machine requires the classification of a huge quadratic programming (QP) improvement issue. SMO breaks this vast QP issue into a progression of smallest conceivable QP issues. These small QP issues are tackled analytically, which abstains from utilizing a tedious numerical QP enhancement as an inner loop. The measure of memory required for SMO is directly dependent on the preparation set size, which permits SMO to deal with large training sets. The data set used in the experiment to test SMO's speed was the UCI "adult" data set, which is available at [25]. The SVM was given 14 attributes of a census form of a household. The training times of SVM and chunking algorithm were compared on the data set. SMO's time scales for the training order is comparatively better than the chunking time. SMO's computation time is mainly dominated by evaluation of SVM subsequently, SMO is faster for linear SVMs and scanty information sets. On real world data, SMO can be more than 1000 times quicker than the chunking algorithm. This motivates us to apply SMO classifier setting to perform the classification task in our project.

Aurangzeb Khan *et al.* [26] reviewed a few machine learning approaches and text classification strategies. Data preprocessing is done using feature extraction and feature selection approaches. In feature extraction, stemming, tokenization, stop words removal is applied on the documents. After feature extraction, vector space is constructed using the Term Frequency/Inverse Document Frequency (TF-IDF) approach. TF-IDF captures relevancy among the words, text documents and particular categories. The statistics from information gain and chi square confirms that these are the mostly used and well performed techniques for feature selection. More than 100 variants of five major feature selection methods are tested using four popular classification algorithms: Naive Bayesian (NB) approach, Rocchio's-style classifier, kNN technique and Support Vector Machines. Among all the machine learning algorithms, Support Vector Machine, Naïve Bayes, kNN and their hybrid approaches with a combination of other algorithms have shown accuracy in the existing literature. NB algorithm performed well in spam filtering and email classification or categorization, which requires a small amount of data for training to calculate the parameters useful for classification. On the other hand, SVM method was able to effectively collect the inherent characteristics and embed the structural risk management principle which reduces the upper bound on the generalization error. However, the main disadvantage of SVM lies in difficulty of parameter tuning and selection of kernels.

Support vector machines (SVMs) were initially intended for binary classification of documents [27]. Step by step the algorithm has been extended for multiclass classification. A few strategies have been proposed where a multiclass classifier is built by combining many other binary classifiers [28]. Some researchers have proposed strategies that consider all the classes at once. The authors compared the methods performance and three techniques based on binary classifications: "one-against-all, “one-against-one," and coordinated non-cyclic diagram SVM (DAGSVM). Their tests show that the "one-against-one" and DAG techniques are more reasonable for pragmatic use than alternate methods. From UCI Repository [29] the following datasets: iris, wine, glass, and vowel were collected. From Statlog collection [29], all multiclass datasets: vehicle, segment, dna, satimage, letter, and shuttle were collected. The data sets were trained using RBF kernel $K(x_i, x_j) \equiv e^{-\gamma||x_i - x_j||^2}$ parameters. The generalized accuracy is measured using different kernel parameters γ and cost parameters C: $\gamma = [2^4, 2^3, 2^2, \ldots 2^{-10}]$ and $C=[2^{12}, 2^{11}, 2^{10}, \ldots 2^{-2}]$. From these experimental results, authors concluded that the SVMs accuracy could be increased by changing the kernel and cost parameters.

As indicated by Simon Tong [30], text classification have a vital role to play due to the recent expansion of readily available data. SVMs have gained remarkable success in solving number of real world problems. Usefulness of SVM based methods [31] has resulted in its widespread popularity.

In this work, authors represent the document as a stemmed, TD-IDF-weighted word frequency vector. Stop word consisting of list of common words were used to ignore the irrelevant features. Because of proven significance, stop words and TD-IDF preprocessing techniques are implemented in our project.

As indicated by Edda Leopold [32], the decision of the kernel function is critical to most applications of support vector machines. Representation of texts in input space is performed by frequency transformations and inverse document frequency. In frequency transformation the term frequencies $f(w_k, t_j)$ is used and multiplied with inverse document frequency. IDF of term ω in a text data collection consisting of N document is calculated suing equation (1). The vector of the IDF for all types in the document collection is given by

$$IDF = (IDF_1, \dots IDF_n) \tag{3}$$

Thus, the product of frequency transformation and the IDF gives the vector of the text to be represented in the input space. Test input data were collected from Reuters-21578 dataset, Frankfurter Rundschau newspaper and Die Tageszeitung (German) newspaper [21]. SVM classifier with linear, polynomial and RBF kernel were applied on the data. The experimental results have shown that the kernel functions did not affect the precision and recall performance very much. Compared to polynomial and linear kernels, RBF kernel generated more support vectors. The experimental results have also shown that SVMs are capable of efficiently processing very high dimensional feature vectors.

According to Kim [33], Support vector machines (SVMs) have been perceived as a standout amongst the best classification strategies for some applications including text categorization. The experimental data consists of a subset of MEDLINE database [34] with 5 categories. Each class has 500 documents and divided into 1250 documents each for training and test set. Using Centroid based algorithms for dimensionality reduction of clustered data: For term document matrix *A*, the reduced dimensional representation is achieved by transforming each document vector in the m dimensional space to a vector in the *l* dimensional space for some *l* < *m*. Dimensional reducing transformation $G^T \in R^{l\times m}$ is computed explicitly or the dimensionality can be achieved by formulating as a rank reducing approximation where the given matrix *A* is decomposed into two matrices *Y* and *B* i.e.

$$A \approx BY \tag{4}$$

Here, $B \in R^{l\times m}$ with rank (*B*) = l and $Y \in R^{l\times n}$ with rank (Y) = l.

The optimal separating hyperplane of one versus rest binary classifier could be obtained by conventional SVMs. The decision rule for SVM is

$$y(x,j) = sign\left(\sum_{x_i \in SV} \alpha_i\, y_i K(x, x_i) + b - \theta_j^{SVM}\right) \tag{5}$$

Here *y* (*x*,*j*) ∈ {+1, -1} is the classification for the document x with respect to the class j, SV is the support vectors set, θ_j^{SVM} is class specific threshold for the binary decision, $\sum_{i=1}^{n} \alpha_i\, y_i = 0, 0 \leq \alpha_i \leq C, i = 1, .. n$ and the kernel function is represented with $K(x, x_i) =< \emptyset(x_i), \emptyset(x_j) >$ where <, > represents the inner product of two vectors, is introduced to handle nonlinearly separable cases without any explicit knowledge of the feature mapping $\emptyset$. This threshold is set so that a new document *x* should not be classified to belong to class *j* when it is located very close to the optimal separating hyperplane. After performing classification using algorithms like SVM and kNN, it was shown that the classification accuracy of the SVM is higher and it achieved a significant reduction in the time and space complexity. Despite the fact that the learning capacity and computational complexity of training in support vector machines might be independent of the dimension of the feature space, diminishing computational complexity is a fundamental issue to proficiently deal with a large number of terms in solving the text classification problems. Novel dimension reduction strategies to lessen the dimension of the text vectors drastically were implemented. Decision functions for the classification algorithm and support vector classifiers to deal with the classification issue where a text document may have the probability of belonging to multiple classes were presented. The significant exploratory results in the paper demonstrate that several dimension reduction strategies that are composed particularly for clustered data, higher efficiency for both training and testing can be accomplished without giving up prediction accuracy of text categorization even when the dimension of the input space is essentially decreased [33].

2.3 Overcoming data imbalance issue

According to R. Longadge *et al.* [5], imbalance data set is the greatest problem in data mining [35]. The authors have proposed three methods for classification of imbalance data set which is divided into three categories namely algorithmic, data preprocessing and feature selection approach. The paper demonstrates systematic study of these approaches and the right direction for the class imbalance (class imbalance occurs when one of the two classes having more sample than other classes) problem. Sampling methods are used in solving the data imbalance problems, which is known as data preprocessing method. Under-sampling and oversampling are the two methods described in this paper. The authors provided the methods to perform the feature selection and algorithmic approaches and concluded that the data preprocessing method provides much better solution than other methods as it allows the addition of new data into the existing data or the deletion of the redundant data, which is of less importance. Under-sampling is an efficient strategy to deal with class imbalance but the drawback of under sampling is that it loses many potential data as some of the data is deleted. In order to achieve good prediction over minority class and avoid necessary information loss from the majority class, both k-means algorithm and random sampling approaches were implemented to generate balanced data sets.

$$Si = X_{Min} \times R_i, 1 \leq i \leq k \tag{6}$$

Here, $R_i = X_{Maj\_i}/X_{Maj}$ for $1 \leq i \leq k$, X_{Maj} is majority class samples, X_{Min} are minority class samples and X_{Maj_i} is the majority of samples in the *i*th cluster. The following algorithm 1 is proposed for implementing under-sampling technique.

```
**Algorithm 1**: Implementation of under-sampling technique.
INPUT: Majority class samples, minority class samples.
OUTPUT: Balanced training set
Step1: Cluster all the majority class samples into $k$ clusters using $k$-means clustering.
Step2: Compute the number of selected majority class samples in each cluster by using equation (6) and select $Si$ majority samples in $i$th cluster. This is done by selecting $Si$ majority class samples in $i$th cluster randomly. Thus, majority sample subsets C is obtained.
Step3: Combine $C$ with all the minority class samples respectively to obtain the training set $B$.
```

The proposed approaches are tested on UCI data sets [29] and found that kNN has improved performance compared to SVM. According to the experimental results, SVM's performance degraded as the class imbalance increased.

According to Rehan [36], SVM has been widely considered and has indicated noteworthy accomplishment in numerous applications. The accomplishment of SVM is extremely constrained when it is applied to the problem of learning from imbalanced datasets in which negative cases intensely dwarf the positive cases (e.g. in gene profiling and distinguishing credit card fraud). Undersampling the majority instances and oversampling the minority instances were the main approaches applied for imbalanced data sets. A new algorithm which needs to bias SVM in a way that will push the boundary away from the positive instances is introduced. The performance of proposed algorithm against the SMOTE algorithm by Chawla *et al.*, consolidated with Veropoulos *et al.*'s different error costs calculation, alongside under-sampling and general SVM, demonstrate that the proposed algorithm works well. The main problem with imbalanced data sets is that they skew the boundary towards the minority class instances. The classification function for the hard margin linear SVM is as follows:

$$Sign((w.b) + b) \tag{7}$$

Here, w is a vector that is normal to the separating hyperplane. The norm of w and variable b decide the distance of the hyperplane from the origin. This skew of the learning hyperplane is tested on UCI datasets [29]. Ideal boundary for this dataset is measured by testing balanced datasets which are linearly separable and noise free in the feature space. When SVM classifier is applied on the balanced UCI data [29], 100% accuracy is achieved. Upon repeating the experiment by keeping the positive instances same and reducing the negative instances, the results have shown a significant angles between ideal and learned hyperplane. Thus, this motivates us to reduce the data imbalance issue in our training set, while applying SVM with a good faith.

According to Yanling Li, Guoshe Sun, Yehang Zhu [6], imbalance in training data set means a significant difference in the number of samples present in the classes. In such cases the classification would tend to over adapt the classes with high number of samples ignoring the smaller classes. In most of the text classification problems this data imbalance is common. For example in monitoring the public opinion data, information security and supervision, most of the cases the quantity of texts which holds a negative view will be very low compared to the positive views. Classification algorithms like Support vector machines, kNN, and neural network are not adaptive in handling the imbalanced data sets. Their paper mainly focused on analyzing different forms of data imbalances including text distribution, class overlap, and class size and devised the conclusions based on experimental values.

3. Preliminary Work

3.1 Approaches to document classification using machine learning

In supervised approach, single-label documents are those which are classified into one class only, multi-label documents are those which are classified into more than one class [2]. In this project we perform multi class classification, in which a file is predicted into one of the predefined categories. Supervised learning models are widely applicable and can offer the insight about how the explanatory variables are related to the categorical response variable. Document classification mainly consists of document representation (vector form), feature selection, feature extraction, application of machine learning algorithm on the data for evaluation. There are different types of supervised text classification techniques available in machine learning platform, which we applied to perform our preliminary assessment discussed below:

kNN (*k Nearest Neighbors*): kNN algorithm is based on the assumption that the classification of a document is analogous to the classification of the other documents that are nearby in the vector space. "*k*" in kNN is a user-defined constant which is used to find the most frequent label of k training samples (documents) nearest to the unlabeled vector or test document.

Support Vector Machines: It is a supervised classification algorithm which is extensively implemented in text classification problems. SVM have the potential to handle large feature spaces as they use over fitting protection which does not depend on the number of attributes [37]. The task of SVM is to find the linearly separators for the available categories.

Random Forest: It is a robust and versatile algorithm, which is used for classification of documents. Random forest consists of many individual trees [38]. In this method, each tree votes on an overall classification for the given dataset and the random forest algorithm choses the most voted individual classification.

J48 Decision Trees: A decision tree is a predictive machine learning model which decides the target value of a new sample based on several attribute values of the available training dataset. A binary tree is created based on the training dataset and it is applied to target instance in the dataset for the classification.

We applied five machine learning methods on the experimental data [3] to carry out the initial classification. These five methods are k- Nearest Neighbors (kNN), two variations of Support Vector Machine (SVM), and two tree- based classification algorithms Random Forest and J48. We have assessed the accuracies of the initial classification with 5 fold cross- validation. The following Table shows training accuracies obtained from various methods using the full- text documents:

Table 1: Comparison of five machine learning methods' training accuracies for the given 15 categories.

kNN (w/ k=9)	**SVM** (w/RBF-kernel)	**J48**	**Random Forest**	**SVM** (w / PolyKernel exp (1.0))
54%	55.4%	57.07%	61.04%	**69.2%**

Based on the results from the above table, we have proceeded with SVM classifier algorithm to make the classification tool more robust for this experimental setup, because, in our preliminary assessment (see Table 1) SVM already has outperformed others.

3.2 A Dissection of Support Vector Machine in Classification

Support Vector Machines is a classifier formally defined by a separating hyperplane in other words, given labelled training data (supervised learning), the algorithm outputs categories. SVM is a relatively new class of machine learning techniques first introduced by Vapnik [37] and has been introduced in Text classification by Joachims. The Support vector machine classifier needs both positive and negative training set, which are uncommon for other classification problems. These positive and negative training set are needed for the SVM to seek for the decision boundary that separates the two classes in the *n* dimensional space, so called the hyperplane. The documents, which are closest to the decision surface, are called the support vectors. The accuracy or efficiency of the Support vector machine classifier classification does not alter if documents that do not belong to the support vectors are removed from the set of training data.

Usually the classification problem can be confined to thought of consideration of the two-class (binary) problem without loss of generality. The main goal of the problem is to separate the two classes by a function which is induced from available examples. The goal is to produce a classifier that will work well on unseen examples. Consider the example in Figure 1. In the figure there are many possible linear classifiers that can separate the data points, but there is only one classifier that maximizes the margin (maximizes the distance between it and the nearest data point of the two classes). Such linear classifier is defined as the optimal separating hyperplane. Intuitively, we would expect this hyperplane to generalize well as opposed to the other possible class boundaries.

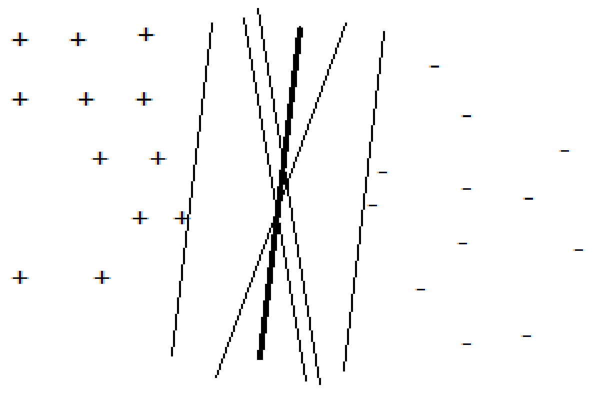


Figure 1: Hyperplane in SVM, possibility of many hyperplanes which divide the data points.

The goal of SVM is to maximize the margin in-between classes of the training data. It implies finding the biggest margin provides the optimal hyperplane for the classifier. A hyperplane separates the two classes based on the equation (8).

$$w.x + b = 0 \tag{8}$$

With $w.x_i + b \geq 1 \; if \; y_i = 1$ and $w.x_i + b \leq -1 \; if \; y_i = -1$ for $i = 1,2, \dots N$

The goal of SVM is to find the optimal separating hyperplane by maximizing the distance between the two classes, which can be calculated from equation (9).

$$min\left(\frac{1}{2}w.w\right) \tag{9}$$

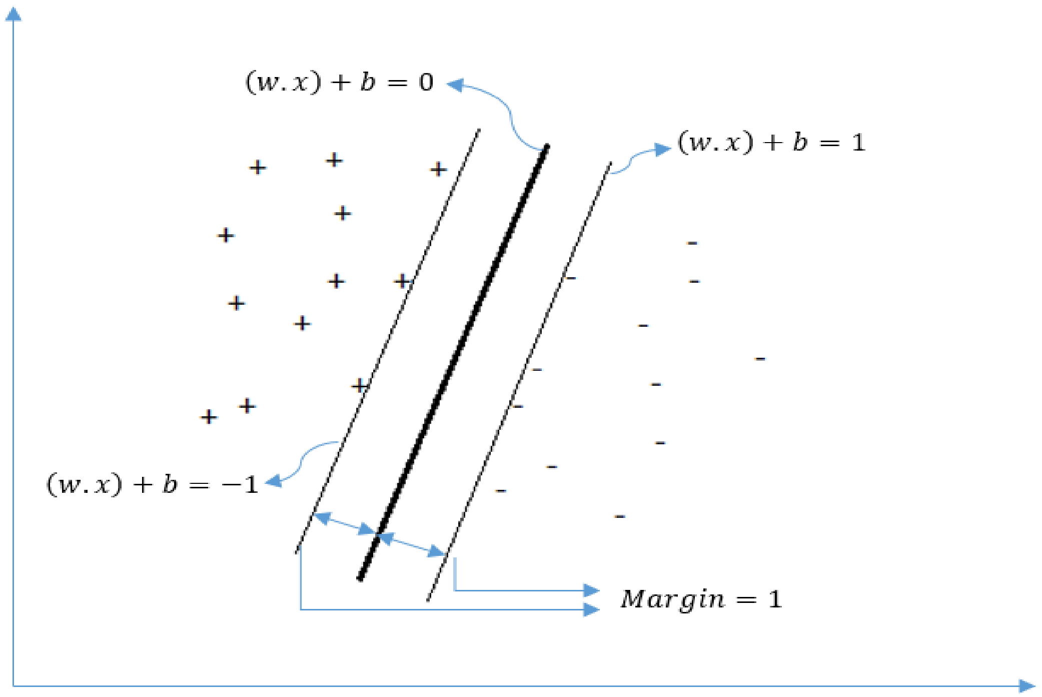


Figure 2: Hyperplane with support vectors in SVM, where the solid line represents the decision boundary while the thin lines represents the boundaries of maximal margin area. Data points lying on the thin lines are the support vectors.

3.3 Attribute Selection

Attribute selection plays a key role in data preparation for training the classifier model. It is a process in which the best set of attributes are selected from the dataset. It is used to create transforms of the dataset such as rescaled attribute values into their constituent parts to make more and useful structure for the classifier models. Keeping irrelevant attributes in the training model could cause overfitting problem.

Tokenization: Text data available is composed of terms grouped into sentences and paragraphs. One technique to represent text data using the vector space model breaks down the textual data into a set of terms. Each of these terms corresponds to an attribute of the input data and therefore becomes an axis in the vector space. The data is then represented as vectors, whose components correspond to the terms contained in the data collection and whose value indicates either a binary present/absent value or a weightage for the term for the data point. This representation is known as the Extracted Term' representation and is the most commonly used representation for text data patterns. The process of breaking the available stream of text into words is called tokenization.

Feature extraction using StringToWordVector: This filtering method transforms string attributes present in the text document into word vectors i.e., creates one attribute for each term that is present in the document. The filtering options for applying on the dataset are discussed in section 3.4.

Information Gain Attribute Evaluator (InfoGainAttributeEval): Once the feature extraction is completed, feature selection is implemented by using the Attribute Selection filter. Information Gain Attribute Evaluator present in the filter allows us to set the threshold of the weights of the features to be selected for classification. In our project we have set the threshold to 0.0 such that all the features with positive information gain are taken into consideration.

3.4 A Machine Learning Toolkit, WEKA

Weka tool is a collection of machine learning algorithms for the data mining/document classification tasks. The algorithms could be applied to the datasets directly or called from user developed java code. This application contains data pre-processing, classification, regression, clustering etc. and many other tools. It is well-suited for developing machine learning schemes and applications. [39]

The following options that are available in StringToWordVector filter are used for preprocessing the training data set. The method executes used to normalize the reiterations to get the multi-terms extraction.

The following options are applied on StringToWordVec filter:

wordsToKeep **-** The number of words (per class if there is a class attribute assigned) to attempt to keep.

outputWordCounts **-** Output word counts rather than Boolean 0 or 1 (indicating presence or absence of a word).

lowerCaseTokens - Set to make all the word tokens to converted to lower case before being added to the dictionary.

normalizeDocLength **-** Sets the word frequencies for a document (instance) to be normalized.

TFTransform – TF stands for *Term Frequency*. TFTransform sets the word frequencies to be transformed into:

$$log(1+f_{ij}) \tag{10}$$

Here, f_{ij} is the frequency of word *i* in document (instance) *j*.

IDFTransform – IDF stands for *Inverse Document Frequency*. IDFTransform sets the word frequencies in a document to into:

$$f_{ij} \times \log\,(num\ of\ Docs/num\ of\ Docs\ with\ word\ i), \tag{11}$$

Here *fij* is the frequency of word *i* in document (instance) *j*.

Term weighting plays vital part in the determination of the success or failure of classification problems. As every term has different level of weightage in a text document, the term weight is associated with every term as an important factor. Term frequency of each word in a document (TF) is a weight which relies on the distribution of each word in text documents. Term frequency can be calculated from the equation (10) and it describes the weightage of the word in the given text document. Inverse document frequency of each word in the document database (IDF) is a weight which relies on the distribution of each word in the document database. It is calculated using equation (11) and describes the weightage of each word in the given documents. TF-IDF, weights a given word to determine how well the word describes an individual document within a corpus. It does this by weighting the term positively for the number of times the term occurs within the given document, while also weighting the term negatively relative to the number of documents which contain the word. Consider term *t* and document *d,* where *t* occurs in *n* of *N* documents in *D*. The TF-IDF function is of the form:

$$TFIDF(t, d, n,N) = TF(t, d) \times IDF(n,N) \qquad (12)$$

TF-IDF approach is run against all terms in all documents in the document corpus to rank the words by their weights. Higher TF-IDF weight indicates that a word is both important to the document, as well as relatively uncommon across the document set. This is often interpreted to imply that the term is significant to the document, and could be used to accurately summarize the document. TF-IDF provides a good heuristic for determining likely candidate keywords. TF-IDF approach is very popular in text classification field and almost all the other weighting schemes are variants of this approach. In vector space model organization of document also affect the performance of system. In this experiment we use term frequency method and inverse document frequency for our experiment setup options.

Applying Snowball Stemmer:

This algorithm reduces inflected/derived words to their word stem, by trimming the root of its inflectional affixes; usually, only suffixes that have been added to the right-hand end of the root word are removed. It reduces the repetitions of words of the same form which gradually reduces the dimensionality of the word vectors.

Removing Stop words using Rainbow list:

Stop words, are the words, which are filtered out prior to the processing of natural language data (text). The presence of these words as attributes does not have any positive impact on the classification. They are usually defined as *functional-words*, which do not carry meaning and influence the classification positively. In our setup, we use rainbow stop words list and have appended them with extra words to make the list effective. Thus by the removal of these stop words, only terms with high information gain could be considered for the training purpose.

4. Methods and their Integrations

Our experimental setup consists of the following 4 stages and the information flow has been shown by Figure 3:

i) Experimental Data Collection

ii) Data Preprocessing

iii) Applying Machine Learning algorithms

iv) Synthetic data Injection

The Experimental data is collected and preprocessed using *StringToWordVec* and *AttributeSelection* filters. Once the data is preprocessed, the model is trained with the data. Depending on the accuracy of the model obtained using cross fold validation, synthetic data is added to the required classes and the process is repeated from stage *ii*.

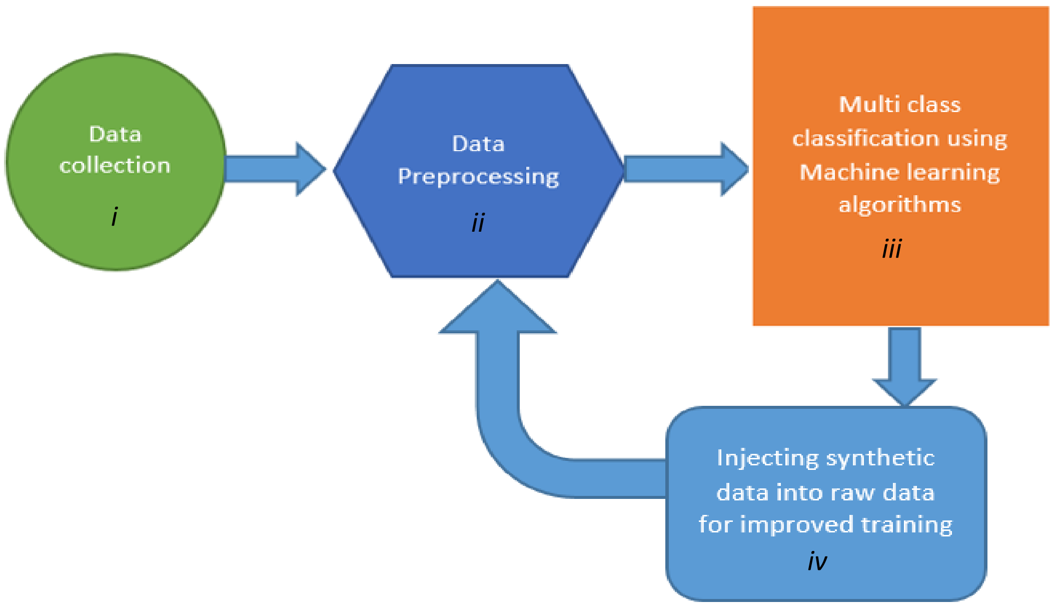

Data collection
i
Data Preprocessing
ii
Multi class classification using Machine learning algorithms
iii
Injecting synthetic data into raw data for improved training
iv

Figure 3: Steps involved in text classification process The Experimental data is preprocessed using unsupervised and supervised filtering techniques. Machine learning algorithms are applied on the preprocessed data. Based on the model accuracy and training samples available in each class, synthetic data is added and the process is repeated from stage *ii*.

Stage 1. Experimental Data Collection

By manual processing, a team of NASA experts identified 15 top-level categories for their patents. The information about these patents is available online. By selecting one of these categories, the NASA's (intelligent properties) portfolio is capable of restricting the results within the chosen category. However, improvement is needed. For example, the precision of the search results of "social network" is only 4.2%; out of 24 output items (descriptions of patents), 23 are irrelevant but included due to the word "network". On the other hand, the relevant item is categorized under "Health, Medicine and Biotechnology". Finding the relevant item through category-by-category search will take time because the logical relationship between "social network" and "Health, Medicine and Biotechnology" is not obvious. The problem of this example can be effectively solved by refined categorization and semantic tagging.

Table 2: Number of training documents available in the experimental data.

SL.	Classes	Patent File Count/Class
1.	Aeronautics	87
2.	Communications	29
3.	Electrical and Electronics	47
4.	Environment	28
5.	Health Medicine and Biotechnology	83
6.	Information Technology and Software	81
7.	Instrumentation	33
8.	Manufacturing	35
9.	Materials and Coatings	190
10.	Mechanical and Fluid Systems	73
11.	Optics	71
12.	Power Generation and Storage	25
13.	Propulsion	24
14.	Robotics Automation and Control	67
15.	Sensors	192

In the experimental data, each patent document were assigned category and were classified into 15 different categories. Table 2 show the available training document for each class. We assumed, that majority of the manually assigned classes to the patent files were correct; but some files could be misclassified. Further, some files could be qualified for more than one classes but only one class has been assigned.

Challenges faced in experimental data

Presence of less informative sections in patent files: Consider the "References" section in a patent document, which does not provide positive information about the document.

Solution: Every section of patent file in the experimental data is not useful for training. Hence, extracting features from the "References" section of a patent document merely increases the dimensionality of the space rather than providing useful information. Thus some sections like, "References", "See also" etc. were removed from the files, which deliberately provides useful words for training the classifier.

Data Imbalance issue: The number of training samples for each class in the experimental data set if not consistent. For example, there is huge difference in the samples count of Sensors class and Power Generation and Storage class.

Solution: The classifier model tends to bend towards the Sensors class due to more number of Support vectors generated by Sensors. To resolve this issue we implemented Synthetic data addition to the classes, which contain relatively less number of training samples. Also, when the data sets are very small, this approach helps in increasing the training material for the model i.e., the number of attributes/features helpful for the classification are increased and also the classifier will be able to predict the unseen patents correctly using synthetic data.

Stage 2: Data Preprocessing

Creating Attribute-Relation File Format (ARFF) file: Weka tool accepts ARFF file format to preprocess the data to perform classification. The text documents in the experimental data were converted into ARFF format. An ARFF file is an ASCII text file that describes the list of instances sharing a set of attributes. ARFF files have two different sections. First section consists Header information and the second section contains data information [40]. The following command has been used to convert the text directory into ARFF file:

```
java weka.core.converters.TextDirectoryLoader -dir <input_directory > ">" <file_name>.arff
```

Once the patent files are converted into ARFF file, it is preprocessed and used for preparing training file for the classifier. The data preprocessing techniques used in this project are discussed in the following sections.

Filtering options: This module processes the texts of the text database are extracted into a set of unigrams (words) that will be used to train the classifier model. It implements two preprocessing methods as follows, and it's the attribute extraction phase.

Applying filters on the data

i) Unsupervised filter

ii) Supervised filter

Here, unsupervised filter *StringToWordVector* is applied on the data, it converts String attributes into a set of attributes representing the word occurrence information from the text contained in the strings.

StringToWordVector filtering options applied for data preprocessing: The options shown in Table 3 can vary depending on the size of the data. The number of attributes generated after applying StringToWordVector filter is 4855 words. Once the *StringToWordVec* filter is applied on the data, the attributes are generated without considering any relationship with respect to the class information (hence, this is unsupervised filtering). To view the attributes in sorted order of information gain, Attribute selection filter is applied on the generated attributes from unsupervised filter.

Table 3: Unsupervised filtering options applied on data in the experimental setup.

Filter Option	**Value set**	**Explanation**
wordsToKeep	1000	This option selects 1000 words from each category as attributes.
outputWordCounts	True	Absence/Presence of word is calculated using 0/1 format.
normalizeDocLength	True	It normalizes the frequencies of word in a document.
TFTransform	True	TFTransform is set true for the calculation of word weight.
IDFTransform	True	IDFTransform is set true for the calculation of word weight.
Stemmer	Snowball stemmer	Snowball Algorithm is selected for this option.
Stop words	Words From File	File containing stop words list is selected for this option (It gives the advantage of adding stop words manually into the file).
tokenizer	. , ; : ' " ( ) ? !	WordTokenizer is selected, such that the document is converted into tokens based on the options.

The Table 4 shows the options to be set in *AttributeSelection* filter. The number of attributes generated after applying Attribute Selection filter is 1218. The Table 5 shows the weights and attributes after applying supervised filter on the data. The attributes are ranked in descending order of their TF-IDF weights.

Table 4: Supervised filtering options applied on data in the experimental setup.

Filter Option	**Value set**	**Explanation**
Evaluator	InfoGainAttributeEval	Attributes are selected based on the information gain ratio.
Search	Ranker	Threshold value is set to 0.0, such that all attributes with positive information gain is considered.

Table 5: Weight (TF-IDF) - Attributes after supervised filtering

Weight (TF-IDF)	**Attribute Name**
0.1800	light
0.1660	materials
0.1637	metal
0.1571	signal
0.1437	optical
…..	…..
0.0508	electromagnetic
0.0478	suspension
0.0476	controllers
0.0471	molecule
0.0465	reflective
…..	…..
0.0242	nanotechnology
0.0220	biomolecule
0.0215	alkali
0.0214	recycling

Thus, attributes that does not contribute much to the document classification are omitted using the above techniques. The threshold values in supervised filter can be changed based on the number of attributes required. The flowing figure shows the frequency distribution of "light" attribute in the documents. We can see from the figure that there is even distribution of the term "light" count in each class (see column wise, i.e. for a particular frequency range, the count of the term is displayed in each column per class), which makes clear that the term provides high information gain for the classification problem.

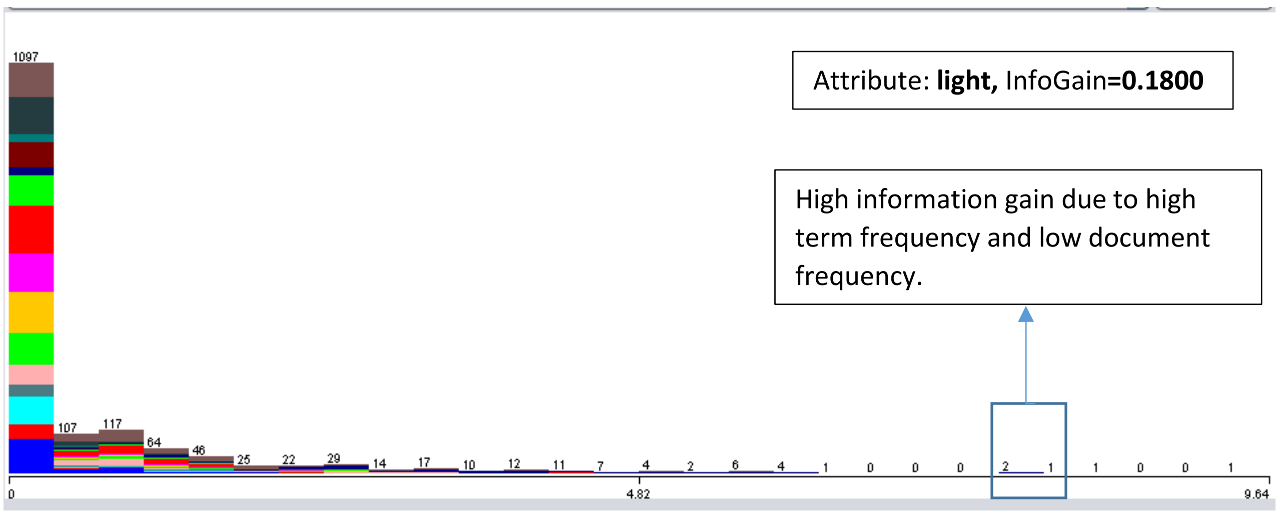


Figure 4: Frequency distribution of "light" attribute in the documents, horizontal axis is the TF-IDF value of the term "light", each color represents a class present in the training dataset and the value on the top of each vertical block represents the total count of the term in the respective frequency zone.

In the above figure we observe that the number of classes that have the term "light" are 2, with frequency approximately in between 6-7. With high term frequency and low document frequency, the TermFrequency-InverseDocumentFrequency (TF-IDF) is high which indicates high information gain for the word "light".

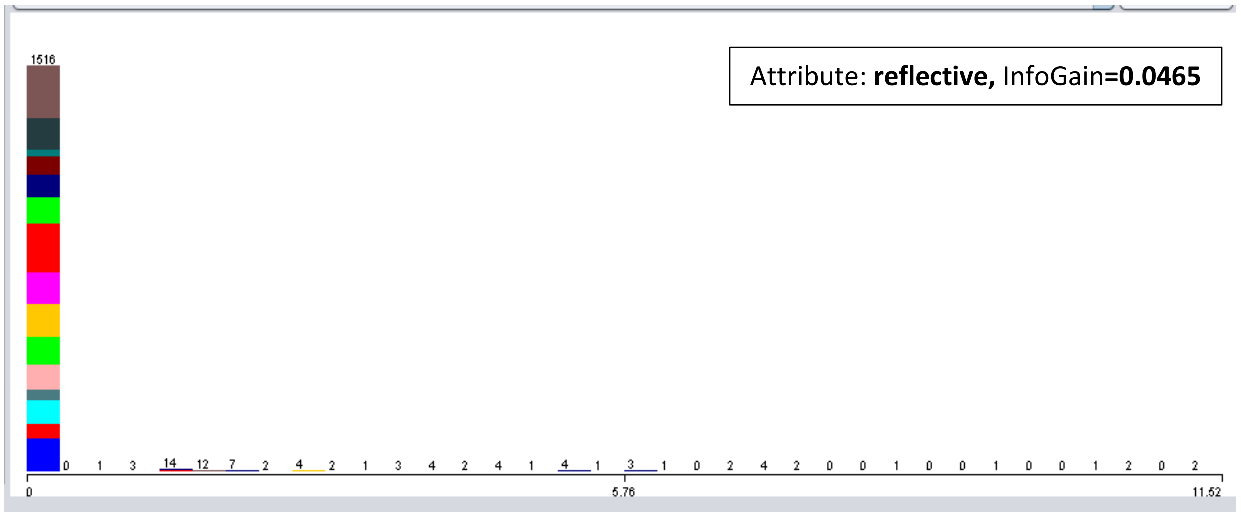


Figure 5: Frequency distribution of "reflective" attribute in the documents

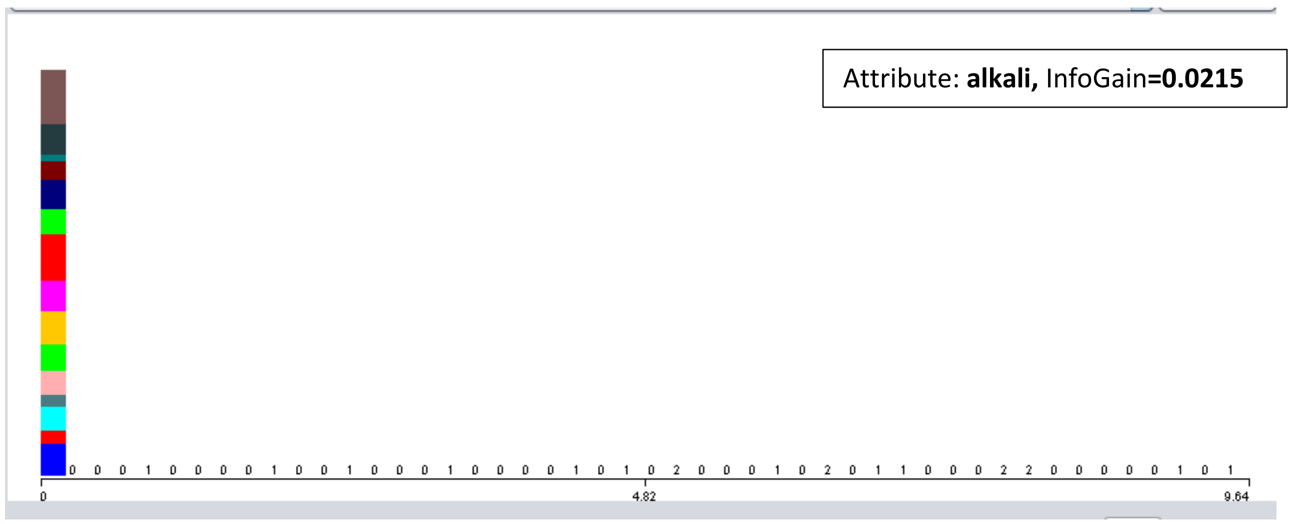


Figure 6: Frequency distribution of "alkali" attribute in the documents

Supervised filtering consists of applying Attribute selection filter on the data which extracts the attribute based on the threshold level that is set in the filter options. This reduces the number of attributes used for the classification. Hence words which provide more information required for the classification are extracted. Once the data preprocessing is completed, the ARFF file is saved and used in the preparation of model. From figures 4, 5, 6 we can see the variation of TF-IDF with the number of documents, the term is present. In figure 4, we see that the information about the class can be obtained more compared to figure 5, 6.

Stage 3. Applying Machine Learning algorithms for classification

Text classification is a task of deciding whether a document belongs to any of a set of pre -defined categories based on the probability. This module implements the Classification process to perform the classification based on the generated knowledge or training data using the Support Vector Machine Classification Approach, the generated output for the NASA patent classification. We used NASA's predefined patent documents as training data for performing classification. Implementation steps were discussed in the following section.

Applying algorithms experimental data: The results from Table 1 suggest that the current data, without refinement cannot be used to build up a robust classifier. For example, kNN is a simpler method and the poor accuracy (54%) indicates that there are many overlapped samples or, miss-labeled samples exist. On the other hand, SVM with various parameters, can very efficiently fit the (augmented) data- space, once the given sample are consider the ground- truth – however, which is not true in this case and the accuracy is not very high (~70% using SVM). Further, the tree based approaches (Random Forest, J48) which are usually good for noisy data are not even providing good accuracy either. Therefore, we planned to identify the good sample to determine the natural and effective class- boundaries using simpler method – the simpler method, such as kNN, will help us avoid over- fitting (SVM or, Tree based approach can over- fit easily).

Robust Training and Testing: To arrange robust and effective training using kNN approach, we implemented the following two steps:

(1) Identify the best *k* value in the kNN approach (see Table 6 and 7),

(2) and iteratively retrain and remove the miss- classified sample and obtain best training accuracy.

Table 6: Comparison of the training accuracies using 5 fold CV with different values of *k* from 1 to 10.

k =	1	2	3	4	5	6	7	8	9	10
accuracy%	60.11	54.98	55.63	55.82	54.70	54.98	55.35	54.14	53.96	55.07

Table 7: Comparison of the training accuracies using 5 fold CV with different values of k from 11 to 20.

k =	11	12	13	14	15	16	17	18	19	20
accuracy%	53.68	54.52	54.70	54.33	54.98	54.70	54.52	54.24	52.84	53.86

We exhaustively computed the classification accuracy, for *k* = 1 to 20, and there is lesser variations among the outcomes. We avoid selecting *k* = 1, because *k* = 1 by nature mostly over-fits, hence avoidable. Thus, we selected the most reasonable as well as optimal value, *k* = 7.

Iteratively retrain and remove the miss- classified sample and obtain best training accuracy: This is the most important step to retain the best samples and retrain using appropriate sample for higher accuracy. This is also a very effort intensive steps, as for each iterations, we had to remove the misclassified files applying semi- automatic approach. The confusion matrix and the accuracy in each steps are shown below. It is to be noted that the accuracy is usually expected to be increasing in each steps – however, it can degrade while there are very insufficient sample for a class or classes, or, the samples among classes become heavily imbalanced:

Iteration # 1: kNN (*k* = 7), accuracy 81.47%, this is achieved by removing miss- classified samples 1st time (shown in Table 8).

Table 8: Confusion matrix of the samples kNN Iteration 1.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	<-- classified as
50	0	2	0	0	1	0	0	0	3	0	0	1	0	0	a = Aeronautics
1	15	2	0	0	0	0	0	0	0	0	0	0	0	0	b = Communications
1	3	19	0	0	0	0	0	0	0	0	0	1	0	0	c = Electrical and Electronics
0	0	0	13	0	0	0	0	0	1	0	1	0	0	0	d = Environment
4	0	3	0	46	0	0	0	1	0	1	0	0	0	1	e = Health Medicine and Biotechnology
5	2	1	0	0	29	0	0	0	0	3	0	0	0	0	f = Information Technology and Software
1	1	0	0	0	0	7	0	0	0	0	0	0	0	0	g = Instrumentation
0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	h = Manufacturing
1	0	3	0	0	0	0	0	115	3	0	0	0	0	1	i = Materials and Coatings
5	0	0	0	0	0	0	0	0	48	0	1	0	0	1	j = Mechanical and Fluid Systems
1	1	2	0	0	0	0	0	0	3	21	0	0	0	2	k = Optics
0	1	0	0	0	0	0	0	0	0	0	4	0	0	0	l = Power Generation and Storage
1	0	0	0	0	0	0	0	0	1	0	0	17	0	0	m = Propulsion
3	0	0	0	0	0	0	0	0	2	0	0	0	39	0	n = Robotics Automation and Control
6	4	3	1	0	0	0	0	2	12	13	0	0	0	53	o = Sensors

Iteration # 2: kNN (k = 7), accuracy 89.73%, this is achieved by removing miss- classified samples 2nd time (shown in Table 9).

Table 9: Confusion matrix of the samples kNN Iteration 2.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	<-- classified as
47	2	0	0	0	0	0	0	0	0	0	0	1	0	0	a = Aeronautics
0	14	0	0	0	0	0	0	0	0	1	0	0	0	0	b = Communications
1	0	17	0	0	0	0	0	0	0	0	0	1	0	0	c = Electrical and Electronics
0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	d = Environment
6	3	1	0	33	0	0	0	0	1	2	0	0	0	0	e = Health Medicine and Biotechnology
3	2	0	0	0	23	0	0	0	0	1	0	0	0	0	f = Information Technology and Software
0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	g = Instrumentation
0	0	2	0	0	0	0	18	0	1	0	0	0	0	0	h = Manufacturing
1	0	3	0	0	0	0	2	109	0	0	0	0	0	0	i = Materials and Coatings
4	0	2	0	0	0	0	0	0	42	0	0	0	0	0	j = Mechanical and Fluid Systems
0	1	1	0	0	0	0	0	0	0	19	0	0	0	0	k = Optics
0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	l = Power Generation and Storage
0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	m = Propulsion
0	0	0	0	0	0	0	0	0	1	0	0	0	38	0	n = Robotics Automation and Control
0	1	2	0	0	0	0	0	1	0	2	0	0	0	47	o = Sensors

After removal of the misclassified samples from the training data, Support vector machine classifier is applied to measure the accuracy of the classification. The confusion matrix shown in Table 10, is achieved by performing SMO poly-kernel method on kNN's second iteration data. The accuracy achieved is, **96.0331**%. It is to be noted, that as the training samples further decreased in classes like "Communications", "Power Generation and Storage" etc. we restored the original samples into the data set to avoid poor training set. To improve the model accuracy further we have proposed to add synthetic data to the training files which is discussed in the following section.

Table 10: Confusion matrix of the samples after applying SMO on iteration 2 data.

```
  a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   <-- classified as

 46   1   0   0   0   0   0   0   1   2   0   0   0   0   0 |  a = Aeronautics
  0  28   0   0   0   0   0   0   0   0   0   0   0   0   2 |  b = Communications
  0   0  26   0   0   2   0   0   2   0   0   0   0   0   0 |  c = Electrical and Electronics
  0   0   0  26   0   0   0   0   0   0   0   0   0   0   0 |  d = Environment
  0   0   0   0  45   0   0   0   0   0   0   0   0   0   1 |  e = Health Medicine and Biotechnology
  1   0   0   0   0  27   0   0   0   0   1   0   0   0   0 |  f = Information Technology and Software
  0   0   0   0   0   0  28   0   0   0   0   0   0   0   0 |  g = Instrumentation
  0   0   0   0   0   0   0  30   0   0   0   0   0   0   0 |  h = Manufacturing
  0   0   0   0   0   0   0   0 114   1   0   0   0   0   0 |  i = Materials and Coatings
  1   0   1   0   0   0   0   0   1  45   0   0   0   0   0 |  j = Mechanical and Fluid Systems
  0   0   1   0   0   0   0   0   0   0  26   0   0   0   0 |  k = Optics
  0   0   0   0   0   0   0   0   0   0   0  28   0   0   0 |  l = Power Generation and Storage
  0   0   0   0   0   0   0   0   0   0   0   0  26   0   0 |  m = Propulsion
  0   0   0   0   0   0   0   0   0   1   0   0   0  38   0 |  n = Robotics Automation and Control
  1   1   0   0   0   0   0   0   3   0   0   0   0   0  48 |  o = Sensors
```

Stage 4. Addition of Synthetic in training data

Reducing the difference in number of training samples between classes: We first tried to increase the number of samples in the less dense classes (i.e., classes having fewer number of training samples) by copy pasting the available files in their respective classes. Doing so will increase the count of files in the classes but could not achieve much accurate results upon testing. As the samples in small classes are copy-pasted, it increases the number of training data and increases the weight of the training attributes. However, as we tested the data, which is not seen by the training model, it resulted in poor accuracy. And the *accuracy* was 46.2366 % and total 93 files tested.

To overcome the problem of text classification involving imbalanced and low training dataset, we have implemented a solution by adding synthetic data in the classes that have less number of samples. The synthetic data has been collected from well- known taxonomical repositories and articles, and fed into the respective classes by generating files having chunk of dataset within them. Keywords related to each class has also been collected in this procedure and are added into text files and are fed into the classes for training purpose. The purpose of the addition of the keywords is to have higher information gain for the keywords/attributes, which are very closely related to a particular class. In future when a new patent document is given for prediction, it is expected that due to feeding relevant in the training synthetically, the prediction will be more accurate. Synthetic data is collected based on the major areas that are sub domains for the top class. For example, the top class, "Aeronautics", we have collect data that is suitable for Sub categories such as:

- Aerodynamics and Fluid Mechanics
- Structures and Materials
- Propulsion and Power

We trained the model with the relevant data and tested it with several other related input files. Based on the data provided by NASA and the synthetic data that is collected, sub categories are classified. We read the documents thoroughly and cross- validated against standard resources like IEEE for the sub category classification and fit the data in the respective sub classes for training.

Table 11: Accuracy of the model, after multiplying the training samples in dataset

```
a b c d e f g h i j k l m n o <- -  classified as
5 0 0 0 0 1 0 0 0 0 0 0 0 0 3 | a = Aeronautics
0 1 0 0 0 0 0 0 0 0 0 0 0 0 3 | b = Communications
0 0 1 0 0 0 0 0 2 1 0 0 0 0 1 | c = Electrical and Electronics
0 0 0 3 0 0 0 0 0 0 0 0 0 0 1 | d = Environment
0 0 0 0 5 1 0 0 1 0 0 0 0 0 1 | e = Health_Medicine and Biotechnology
0 0 0 0 0 4 0 0 0 0 0 0 0 3 0 | f = Information Technology and Software
1 0 0 0 0 0 0 0 1 0 0 0 0 0 3 | g = Instrumentation
0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 | h = Manufacturing
0 0 0 0 0 0 0 0 9 0 0 0 0 0 2 | i = Materials and Coatings
0 0 0 0 0 0 0 0 0 3 0 0 0 0 3 | j = Mechanical and Fluid Systems
0 0 0 0 0 1 0 0 1 1 0 0 0 0 4 | k = Optics
0 0 0 0 0 0 0 0 1 0 0 0 0 0 3 | l = Power Generation and Storage
0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 | m = Propulsion
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | n = Robotics Automation and Control
0 0 0 0 0 1 0 0 2 1 1 0 0 0 9 | o = Sensors
```

Data- collection: Collecting the data from MediaWiki database: The main advantage of adding synthetic data is that the model can handle new/unseen data that comes for testing hence can result in better prediction. By adding synthetic data classes, the ill impact of imbalance has been subsided and the domination of the higher sample classes reduces gradually, which ultimately improves the model accuracy for testing cases, i.e., true prediction in classification enhanced.

Table 12: Training sample File count before and after insertion of synthetic data

SL.	Classes	Old File Count/Class	New File Count/Class
1.	Aeronautics	87	125
2.	Communications	29	52
3.	Electrical and Electronics	47	93
4.	Environment	28	44
5.	Health Medicine and Biotechnology	83	98
6.	Information Technology and Software	81	105
7.	Instrumentation	33	130
8.	Manufacturing	35	121
9.	Materials and Coatings	190	192
10.	Mechanical and Fluid Systems	73	99
11.	Optics	71	112
12.	Power Generation and Storage	25	73
13.	Propulsion	24	24
14.	Robotics Automation and Control	67	118
15.	Sensors	192	214
	Total =	**1065**	**1600**

Using web crawler program to collect data: Collecting data from each source is a tedious task. This issue can be resolved by using web crawling technique. A web crawler is a simple program to collect data from web, which uses web links as inputs. We built a web crawler to collect data from internet (MediaWiki) and use it in training data set. We have collected sub category list for each top category from IEEE taxonomy. We provide web crawler the URL of MediaWiki and a taxonomy word to search for. The crawler will go to the web page and download as text file and if the link doesn’t work, it will go to the next page and repeat. Visiting the pages that is already visited should be taken care of while extracting data. Thus the data that is required to reduce the class imbalance issue is collected. The collected data is added to the training samples and used to build a document classifier (the distribution is shown in Table 12).

Table 13 Confusion matrix of the test samples before addition of synthetic data.

```
a b c d e f g h i j k l m n o <- -  classified as
9 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | a = Aeronautics
0 3 0 0 0 0 0 0 0 0 1 0 0 0 1 | b = Communications
0 0 4 0 0 0 0 0 0 0 1 0 0 0 1 | c = Electrical and Electronics
0 0 1 3 0 0 0 0 1 0 0 0 0 0 0 | d = Environment
0 0 0 0 7 0 0 0 0 0 0 1 0 0 1 | e = Health Medicine and Biotechnology
0 0 0 0 0 5 0 0 0 0 1 0 0 1 2 | f = Information Technology and Software
1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 | g = Instrumentation
0 0 0 0 0 0 0 2 2 0 0 0 0 0 1 | h = Manufacturing
1 0 0 0 0 0 0 3 12 0 0 0 0 0 0 | i = Materials and Coatings
0 0 0 0 0 0 0 0 0 7 0 0 0 0 1 | j = Mechanical and Fluid Systems
0 0 0 0 0 0 1 0 1 0 5 0 0 0 0 | k = Optics
0 0 1 0 0 0 0 0 1 0 0 2 0 0 0 | l = Power Generation and Storage
0 0 0 0 0 0 0 0 0 1 0 0 3 0 0 | m = Propulsion
0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 | n = Robotics Automation and Control
0 0 2 0 1 2 3 0 0 1 2 0 0 0 6 | o = Sensors
```

Table 12 shows the increase in file count for each category using web crawling technique. The training samples increased from 1065 to 1600, which reduced the data imbalance issue in the training dataset. The confusion matrix in Table 13 shows the accuracy (which is 64.6552%) before and Table 14 shows the accuracy (which is 69.8276%) of the model after the addition of synthetic data that is tested on 116 randomly selected files from each class using the training data that is generated after kNN second iteration (see Table 6).

Table 14: Confusion matrix of the test samples after addition of synthetic data.

```
a b c d e f g h i j k l m n o <- -  classified as
9 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | a = Aeronautics
0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 | b = Communications
0 0 4 0 1 0 0 0 0 0 0 0 0 0 1 | c = Electrical and Electronics
0 0 0 3 1 0 0 0 0 0 1 0 0 0 0 | d = Environment
0 0 0 0 7 0 0 0 0 0 0 1 0 0 1 | e = Health Medicine and Biotechnology
0 0 1 0 0 4 0 0 0 0 1 0 0 1 2 | f = Information Technology and Software
0 0 0 0 0 0 1 0 1 0 1 0 0 0 2 | g = Instrumentation
0 0 0 0 0 0 0 2 2 0 0 0 0 0 1 | h = Manufacturing
0 0 0 0 0 0 0 2 13 1 0 0 0 0 0 | i = Materials and Coatings
0 0 0 0 0 0 0 1 0 7 0 0 0 0 0 | j = Mechanical and Fluid Systems
0 0 0 0 0 0 1 0 1 0 4 0 0 0 1 | k = Optics
0 0 0 0 0 0 0 0 1 0 1 2 0 0 0 | l = Power Generation and Storage
0 0 0 0 0 0 1 0 0 1 0 0 2 0 0 | m = Propulsion
0 0 0 0 0 0 0 0 0 1 0 0 0 5 0 | n = Robotics Automation and Control
0 0 0 0 0 0 1 0 1 1 0 0 0 1 13 | o = Sensors
```

Thus we see, there is a clear improvement about 5% in test accuracy of the same model. Adding synthetic data has proven that the model efficiency could be increased for this kind of data. The reason behind the improvement of the accuracy of the model is the weights (TF-IDF) of the attributes which provide more information gain is increased by the addition of the terms. Hence, it confirms that this approach improves overall prediction accuracy of the classifier when the synthetic data injection idea is fully applied.

5. Sub categories within top categories

Along with the classification of top 15 categories for NASA, we started developing classifier model to perform Sub category classification for the 15 NASA categories. The NASA patent document data set we were experimenting is not big enough to train a model for the classification of sub categories. Thus we have used the synthetic data approach to collect relevant data from internet sources to train the classifier model.

The main advantage of adding synthetic data is to reduce the class imbalance between the classes. But when data sets are very small, this approach helps in increasing the training material for the model i.e., the number of attributes/features helpful for the classification are increased.

Based on the IEEE taxonomy [7], we have collected data for the classification of sub categories. We used web crawling technique to harvest the useful data from internet. This web crawler uses the taxonomy and searches for the links in Wikipedia pages and visit pages according to the category tree in the MediaWiki content repository and collects the data. Once the data mining is finished, it is cleaned and added into the sub category folders and preprocessing is done to train the model.

Possible sub- categories: We have selected sub categories for the top classes based on the IEEE taxonomy and mediawiki category tree. The following list could be further increased based on the data available.

Table 15: Sub categories suggestions: Based on IEEE taxonomy and MediaWiki category tree, each top category is sub classified into different sub categories.

Sub- categories for the top category: Aerospace Engineering
- Aerodynamics and Fluid Mechanics
- Structures and Materials
- Propulsion and Power
Sub- categories for the top category: Communication
- Communication Technology and New Media
- Digital Communications and Networking
- Systems and Communications
Sub- categories for the top category: Electrical and Electronics
- Circuits
- Electro Magnetics
Sub- categories for the top category: Environment:
- Geo Science
- Oceanography
- Social Factors
Sub- categories for the top category: Health, Medicine and Biotechnology:
- Bio Informatics
- Computational biology
Sub- categories for the top category: Information Technology and Software:

- Databases/Information Systems
- Software Engineering
Sub- categories for the top category: Instrumentation:
- Mechatronics
- Biomedical devices and instrumentation
Sub- categories for the top category: Manufacturing:
- Agile Manufacturing
- Electronic Devices and Semiconductor Manufacturing
Sub- categories for the top category: Materials and Coatings:
- Structural Materials
- Materials Chemistry
- Biology and Biomimetic Materials
Sub- categories for the top category: Mechanical and Fluid Systems:
- Acoustics, Dynamics, and Controls
- Applied Mechanics
- Biomechanical Engineering
- Computer- Aided Engineering and Design
- Electro- Mechanical Systems
- Energy Systems
- Heat Transfer, Combustion
Sub- categories for the top category: Optics:
- Bio imaging and biomedical optics
Sub- categories for the top category: Power Generation and Storage:
- Propulsion and Power
- Power and Energy
Sub- categories for the top category: Propulsion:
- Spacecraft propulsion
- Jet engines
Sub- categories for the top category: Robotics, Automation and Control:
- Artificial Intelligence
- Robotics
Sub- categories for the top category: Sensors:
- Wireless sensor networks
- Image Sensors

6. Design and development of application program

We have built a simple software to perform the document classification automatically by selecting the test file. Training documents are collected from NASA patent repository, data preprocessing is performed and used as training data set for the model. After the input file is selected, the model performs evaluation using SMO classifier using training dataset and the predicted ouput is displayed.

We used Java platform, Weka to import classifier methods, and Eclipse IDE to develop the software architecture and the interface to classify or categorize the given document by user. In brief, for a given document, the software will provide user suggested class or category visually using a pie chart (to indicate the probable portion) – so the user immediately get an idea of document sharing by the best as well as other categories.The following screenshots of the tool will desecibe the whole procedure of category prediction as well as will provide an idea of how to use the software.

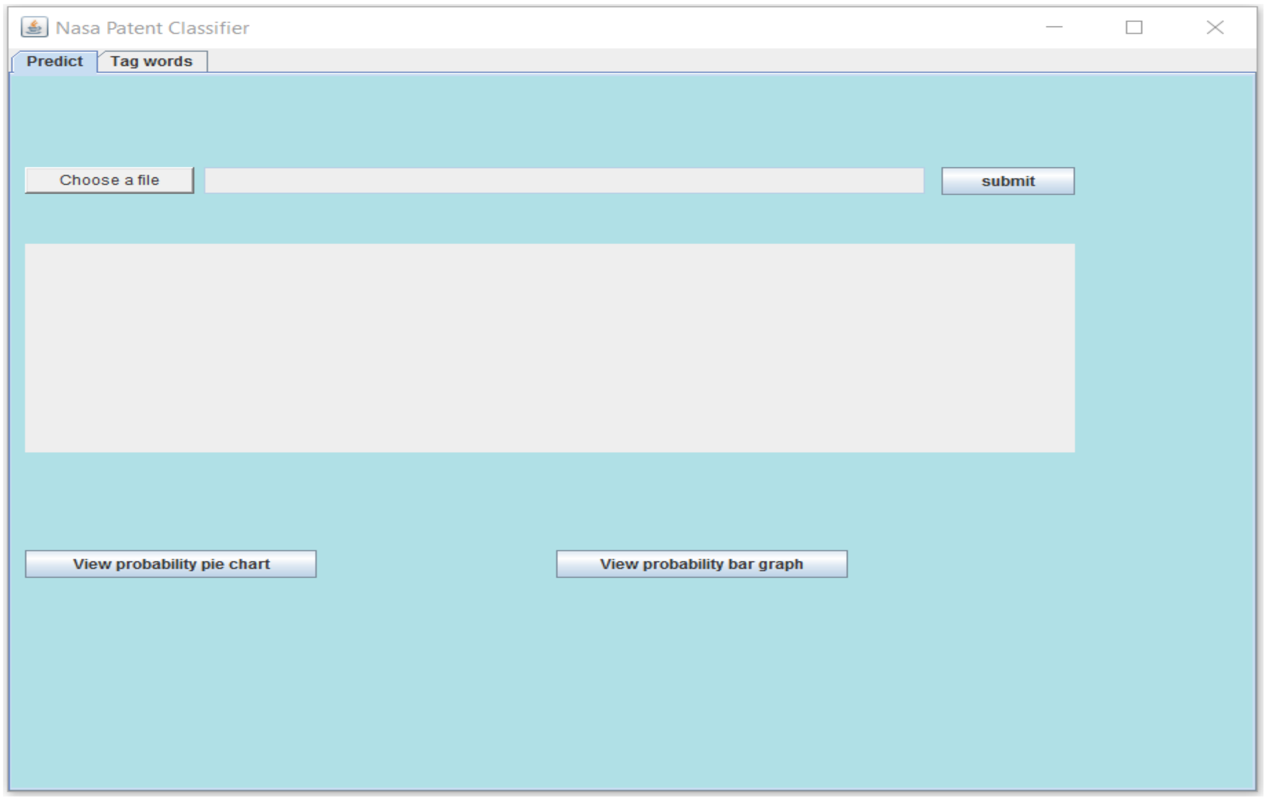
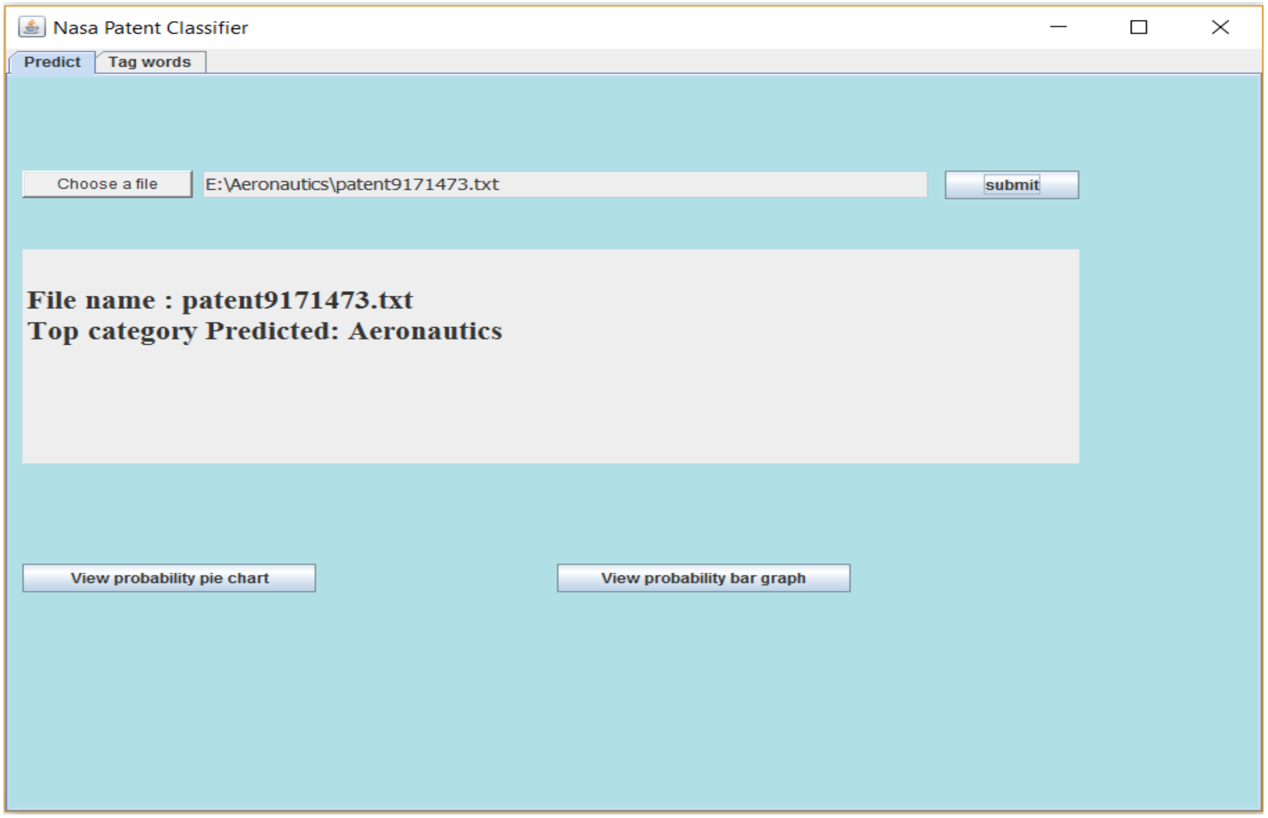


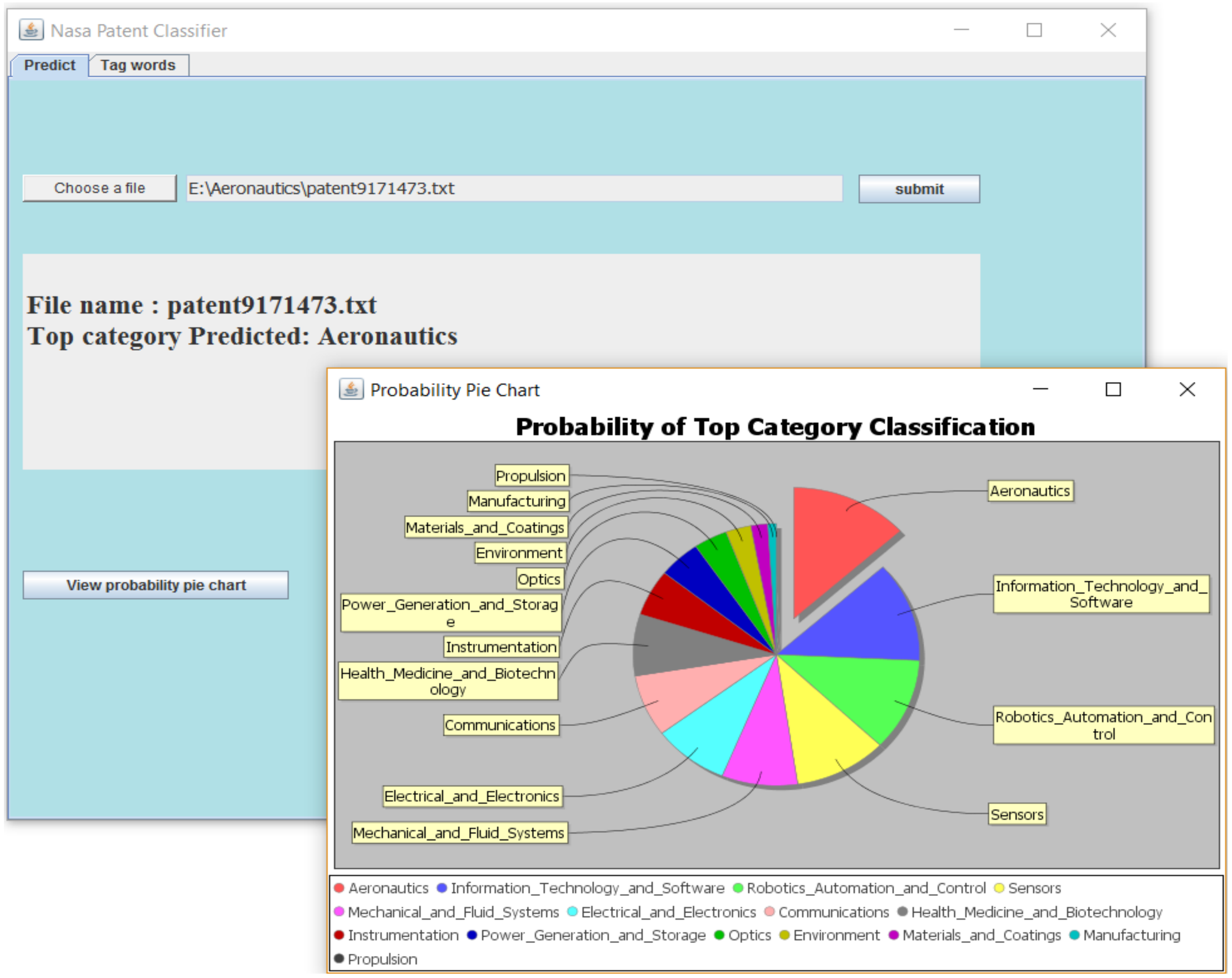
Figure 7: Home page of Text classification tool. After selecting file from a location, click "Submit" button for the output. Once the prediction is completed, the output is shown in the text area present in the window.


Nasa Patent Classifier
Predict
Tag words
Choose a file
E:\Aeronautics\patent9171473.txt
submit
View probability pie chart
View probability bar graph

File name : patent9171473.txt
Top category Predicted: Aeronautics

Figure 8: Output prediction window for Text classification tool. Once an input file is selected, it is evaluated by the classifier model and output is predicted.

User can view the probability distribution over different categories for the file chosen. “View probability pie chart” provides the probability distribution of the prediction category in the form of a pie chart. By scrolling the mouse pointer on to a particular category will show the percentage distribution of that particular category. Figure 6 shows the probability distribution of the test document instance.


Nasa Patent Classifier
Predict
Tag words
Choose a file
submit
View probability pie chart
Probability Pie Chart
Probability of Top Category Classification
Propulsion
Manufacturing
Materials_and_Coatings
Environment
Optics
Power_Generation_and_Storage
Instrumentation
Health_Medicine_and_Biotechnology
Communications
Electrical_and_Electronics
Mechanical_and_Fluid_Systems
Aeronautics
Information_Technology_and_Software
Robotics_Automation_and_Control
Sensors
● Aeronautics ● Information_Technology_and_Software ● Robotics_Automation_and_Control ● Sensors
● Mechanical_and_Fluid_Systems ● Electrical_and_Electronics ● Communications ● Health_Medicine_and_Biotechnology
● Instrumentation ● Power_Generation_and_Storage ● Optics ● Environment ● Materials_and_Coatings ● Manufacturing
● Propulsion

```
E:\Aeronautics\patent9171473.txt
```

```
File name : patent9171473.txt
Top category Predicted: Aeronautics
```

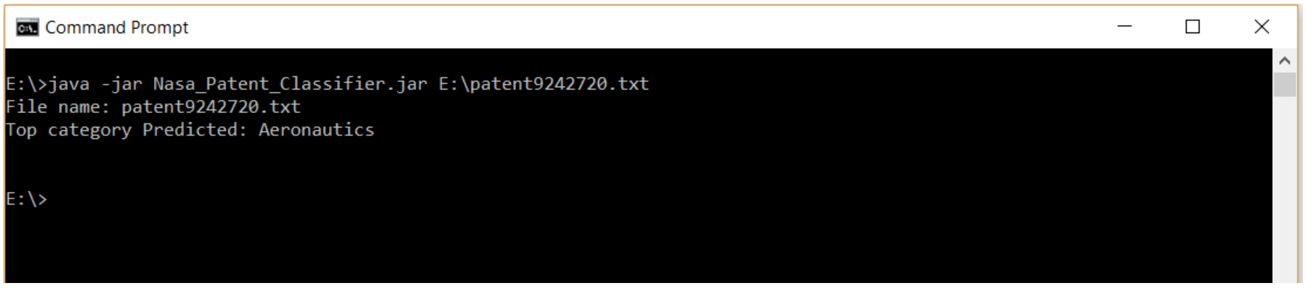
Figure 9: Probability distribution pie chart for the test instance: input file probability distribution over all the labeled classes is shown based on the classifier evaluation.

Operating tool through command line: In order to perform batch processing the input files, command line version provides great platform for the user. By providing the location of the tool and the input file location, document classification can be performed. This version also provides the probability distribution of the input file along with the category prediction. Command line options were implemented in the code to help user to view output in multiple ways i.e., tool performs only category prediction with no options, category prediction with probability distribution with “p” option etc.

The following images shows the operation of tool through command line version. By using the syntax shown in the following figure, user is able to view the predicted category of the document.

Example syntax:

```
java -jar Nasa_Patent_Classifier.jar E:\ patent9242720.txt
```



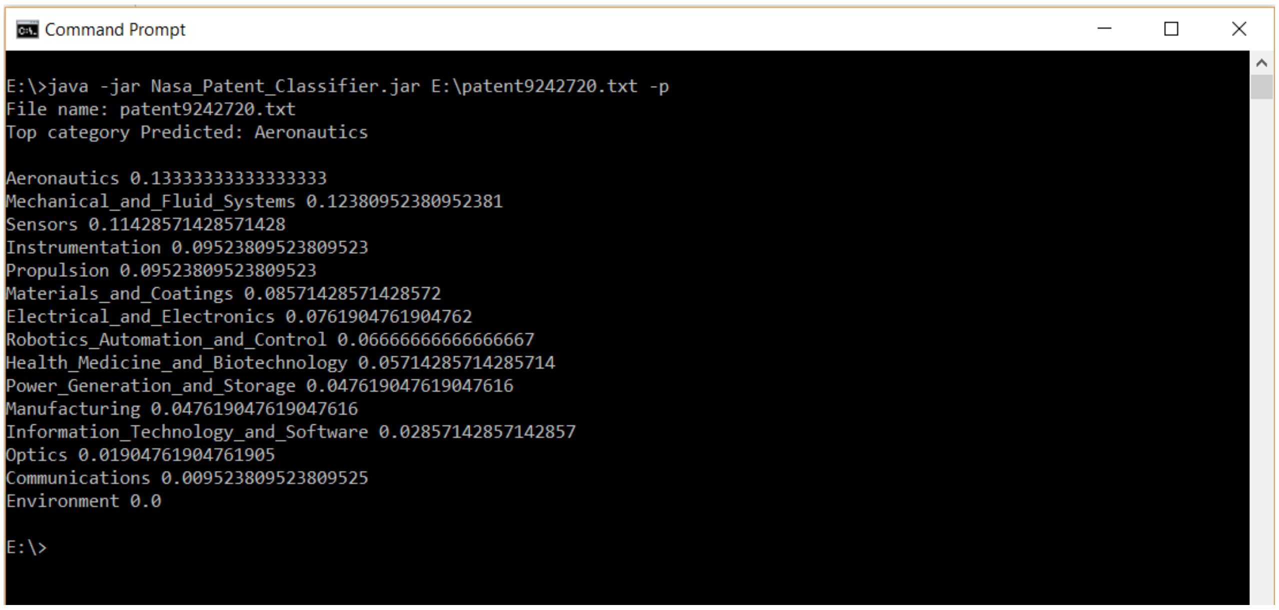
```
E:\>java -jar Nasa_Patent_Classifier.jar E:\patent9242720.txt
File name: patent9242720.txt
Top category Predicted: Aeronautics

E:\>
```

Figure 10: Operation of tool through command line version, inputs: Application's runnable jar file location and a test file.

The syntax shown in the following figure shows the operation of tool to display the probabilities of the top level category.

Example syntax:

```
java -jar Nasa_Patent_Classifier.jar E:\ patent9242720.txt -p
```



```
E:\>java -jar Nasa_Patent_Classifier.jar E:\patent9242720.txt -p
File name: patent9242720.txt
Top category Predicted: Aeronautics

Aeronautics 0.13333333333333333
Mechanical_and_Fluid_Systems 0.12380952380952381
Sensors 0.11428571428571428
Instrumentation 0.09523809523809523
Propulsion 0.09523809523809523
Materials_and_Coatings 0.08571428571428572
Electrical_and_Electronics 0.0761904761904762
Robotics_Automation_and_Control 0.06666666666666667
Health_Medicine_and_Biotechnology 0.05714285714285714
Power_Generation_and_Storage 0.047619047619047616
Manufacturing 0.047619047619047616
Information_Technology_and_Software 0.02857142857142857
Optics 0.01904761904761905
Communications 0.009523809523809525
Environment 0.0

E:\>
```

Figure 11: Operation of tool through command line version with options. For this application, we have developed the code to provide the probability distribution of the input file when option "p" is passed thorugh command line. Developers can add more options based on the experimental requirements.

7. Conclusions and Future Works

In this work, we designed a novel method for patent document classification and sub classification using machine learning techniques. Our method involves data preprocessing techniques necessary for document classification and handling overlapped classes. In this study, we found that the accuracy of the document classifier (5 fold cross validation results) improves from 69.20 % to 96.03% with the removal of misclassified files from the training dataset. In addition, data imbalance issue causes a drop in the accuracies of the classifier models. To mitigate the problem, we applied several techniques to reduce the data imbalance issue in training dataset, but addition of synthetic data worked best in improving the classifier accuracy. This process might help the future researchers to proceed further in the usage of synthetic data for the reduction of data imbalance problem in training sets. We have developed two models for document classification, first, excludes the misclassified files from the training data, second, and does not exclude misclassified files, where the former model achieved more accuracy percentage. It is to be noted that both the models are equally important as the class overlap helps us acquire the probabilities with which the input file fall into multiple classes. Furthermore, we suggest the method of obtaining the tag words which provide fine-grained level of details about the input file or categories generated based on the information gain of the terms. The terms with highest frequency gain from the data were considered as the tag words. We would like to draw the attention of researchers to perform in detail analysis and implementation of sub category classification, as it needs more training data for building a robust model, which can be achieved by collecting more synthetic data.

References

1. Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. 2009: Springer.
2. Wang, T.-Y. and H.-M. Chiang, *Solving multi-label text categorization problem using support vector machine approach with membership function.* Neurocomputing, 2011. **74**(17): p. 3682-3689.
3. *Patft*. [cited 2017 March 17th]; Available from: http://patft.uspto.gov/netahtml/PTO/index.html.
4. Kullback, S., *Information Theory and Statistics*. 1959, NY: Wiley.
5. Longadge, M.R., M.S.S. Dongre, and L. Malik, *Multi-cluster based approach for skewed data in data mining.* Journal of Computer Engineering (IOSR-JCE), 2013. **12**(6): p. 66-73.
6. Li, Y., G. Sun, and Y. Zhu. *Data imbalance problem in text classification*. in *Information Processing (ISIP), 2010 Third International Symposium on*. 2010. IEEE.
7. *IEEE Taxonomy*.
8. F., S., *Machine learning in automated text categorization.* Acm Computing Surveys, 2002. **34**(1): p. 1-47.
9. *Ohsumed and Reuters text classification datasets – download*. [cited 2017 March 18th]; Available from: https://www.mat.unical.it/OlexSuite/Datasets/SampleDataSets-about.htm.
10. Joachims, T., *Text categorization with support vector machines: Learning with many relevant features.* European conference on machine learning. Springer Berlin Heidelberg,, 1998.
11. Porter, M.F., *An algorithm for suffix stripping.* Program, 1980. **14**(3): p. 130-137.
12. Rennie, J.D. and R. Rifkin, *Improving multiclass text classification with the support vector machine.* 2001.
13. Lang, K. *Newsweeder: Learning to filter netnews*. in *Proceedings of the 12th international conference on machine learning*. 1995.
14. McCallum, A. and K. Nigam. *A comparison of event models for naive bayes text classification*. in *AAAI-98 workshop on learning for text categorization*. 1998. Citeseer.
15. Tishby, N.S.N.F.N., *Agglomerative multivariate information bottleneck.* 2001.
16. *20 Newsgroups*. [cited 2017 March 20th]; Available from: http://www.ai.mit.edu/people/jrennie/ecoc-svm/.
17. *Improving Multiclass Text Classification with the Support Vector Machine*. [cited 2000 Feb 25]; Available from: http://www.ai.mit.edu/people/jrennie/ecoc-svm/.
18. Bishop, C.M., *Pattern Recognition and Machine Learning*. 2009: Springer.
19. Forman, G., *An extensive empirical study of feature selection metrics for text classification.* Journal of machine learning research, 2003. **3**(Mar): p. 1289-1305.
20. Yang, Y. and J.O. Pedersen. *A comparative study on feature selection in text categorization*. in *Icml*. 1997.
21. *Reuters-21578 Text Categorization Collection Data Set*. [cited 2017 March 20th]; Available from: https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection.
22. Ikonomakis, M., S. Kotsiantis, and V. Tampakas, *Text classification using machine learning techniques.* WSEAS transactions on computers, 2005. **4**(8): p. 966-974.
23. Zu, G., et al. *Accuracy improvement of automatic text classification based on feature transformation*. in *Proceedings of the 2003 ACM symposium on Document engineering*. 2003. ACM.
24. Platt, J., *Sequential minimal optimization: A fast algorithm for training support vector machines.* 1998.
25. *Index of /pub/machine-learning-databases/adult, .* [cited 2017 March 20th]; Available from: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult.
26. Khan, A., et al., *A review of machine learning algorithms for text-documents classification.* Journal of advances in information technology, 2010. **1**(1): p. 4-20.
27. Hsu, C.-W. and C.-J. Lin, *A comparison of methods for multiclass support vector machines.* IEEE transactions on Neural Networks, 2002. **13**(2): p. 415-425.
28. Islam, M.N., et al., *A balanced secondary structure predictor*. Journal of theoretical biology, 2016. **389**: p. 60-71.

29. *Machine Learning Repository*. [cited 2017 March 19th]; Available from: https://archive.ics.uci.edu/ml/datasets.html.
30. Tong, S. and D. Koller, *Support vector machine active learning with applications to text classification.* Journal of machine learning research, 2001. **2**(Nov): p. 45-66.
31. Iqbal, S., A. Mishra, and M.T. Hoque, *Improved prediction of accessible surface area results in efficient energy function application.* Journal of theoretical biology, 2015. **380**: p. 380-391.
32. Leopold, E. and J. Kindermann, *Text categorization with support vector machines. How to represent texts in input space?* Machine Learning, 2002. **46**(1-3): p. 423-444.
33. Kim, H., P. Howland, and H. Park, *Dimension reduction in text classification with support vector machines.* Journal of Machine Learning Research, 2005. **6**(Jan): p. 37-53.
34. *U.S. National library of Medicine MedLine dataset*. [cited 2017 March 19th]; Available from: https://www.nlm.nih.gov/databases/download/pubmed_medline.html.
35. Islam, M.N., et al., *A balanced secondary structure predictor* Journal of Theoretical Biology, 2016. **389**: p. 60–71.
36. Akbani, R., S. Kwek, and N. Japkowicz. *Applying support vector machines to imbalanced datasets*. in *European conference on machine learning*. 2004. Springer.
37. Cortes, C. and V. Vapnik, *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.
38. Liaw, A. and M. Wiener, *Classification and regression by randomForest.* R news, 2002. **2**(3): p. 18-22.
39. *Weka*. [cited 2017 March 25th]; Available from: http://www.cs.waikato.ac.nz/ml/weka/.
40. *Attribute-Relation File Format (ARFF)*. [cited 2017 March 26th]; Available from: http://www.cs.waikato.ac.nz/ml/weka/arff.html.

Vita

The author was born in Andhra Pradesh, India on Sept 14th 1992. After obtaining his Bachelor's degree in 2014, he joined University of New Orleans's Computer Graduate Program in 2015 and worked as a research assistant under Dr. Md Tamjidul Hoque of computer science department of UNO. He worked on text classification project with the objective to develop a set of automated tools that can assist NASA to manage and market NASA portfolio of intellectual properties (IP), and to enable easier discovery of relevant IP by users.