<u>Introduction</u>

Background of the problem:

Training speed of the classifier without degrading its predictive capability is an important concern in text classification. **Feature selection** plays a key role in this context. It selects a subset of **most informative words (terms)** from the set of all words. The correlative association of words towards the classes increases an uncertainty for the words to represent a class. The representative words of a class are either of positive or negative nature. The standard feature selection methods, viz. **Mutual Information (MI), Information Gain (IG), Discriminating Feature Selection (DFS) and Chi Square (CHI),** do not consider positive and negative nature of the words that affects the performance of the classifiers. To address this issue, here a new feature selection method named **Correlative Association Score (CAS)** is represented. It combines the **strength, mutual information, and strong association of the words to determine their positive and negative nature for a class**. CAS selects a **few (k) informative words** from the set **of all words (m).** These informative words generate a set of **N-grams of length 1-3**. Finally the top most, **b informative N-grams**, where b is a number set by an empirical evaluation are selected based on **TF-IDF** values and then a **vocabulary of N-grams** is created to train the model. **Multinomial Naive Bayes (MNB) and Linear Support Vector Machine (LSVM) classifiers** evaluate the performance of the selected N-Grams. The training time of SVM is higher than MNB but the classification results of SVM are more accurate.

Literature:

The related preliminary concepts, i.e. **word representation, normalization, feature selection, and text document classification** are described in this section.

- **Word Representation**: The representation of the words in the form of vectors is the base to determine the computational informativeness of the words and plays a vital role in an automatic classification of the text documents. The most common models to represent the words as vectors are the ***Bag Of Words (BOW) and N-grams Language (NGL).*** In BOW model, the frequency of each word in the documents of the corpus represents a vector. In this model, the order of word occurrence is not important. The N-grams are the combination of 2–4 words that co-occurred together in the documents. In the NGL model, the set of N-grams represents a **vector space**. Thus, the order of word occurrence is maintained in the NGL model and improves the quality of word representation.

- **Normalization**: Normalization is a technique of scaling the data in a fixed range. The authors (***Dewang, R.K., & Singh, A.K. (2017). State-of-art approaches for review spammer detection: a survey .Journal of Intelligent Information Systems; Agnihotri, D., Verma, K., & Tripathi, P. (2014). Pattern and cluster mining on text data. In IEEE Computer Society, CSNT, Bhopal In Fourth International Conference on Communication Systems and Network Technologies 2014*** and ***Sebastiani, Machine learning in automated text classification. ACM Computing Surveys, (2002))*** addressed problems like **keyword spamming, scaling up frequent words and scaling down rare words**. The problem of keyword spamming occurred when a word appears repeatedly in a document with the purpose of improving its ranking on the Information Retrieval system or even to create a bias towards longer documents. The word frequency in a document of a vector space is usually normalized using the **Term Frequency-Inverse Document Frequency (TF-IDF)** method to overcome this problem. (***Agnihotri, D., Verma, K., & Tripathi, P. (2014). Pattern and cluster mining on text data*** , ***Manning, C.D., Raghavan, P., & Schutze, H. (2008). Introduction to information retrieval. NY: Cambridge University Press, Sebastiani 2002).***

$$Wi,j = tfi,j * \log\frac{Nd}{dfi}$$

Where Wi,j = weight for word t_i in document d_j ,
Nd = Total number of documents in the corpus,
tfi,j = frequency of word t_i in document d_j ,
dfi = document frequency of i^{th} *word in the corpus.*

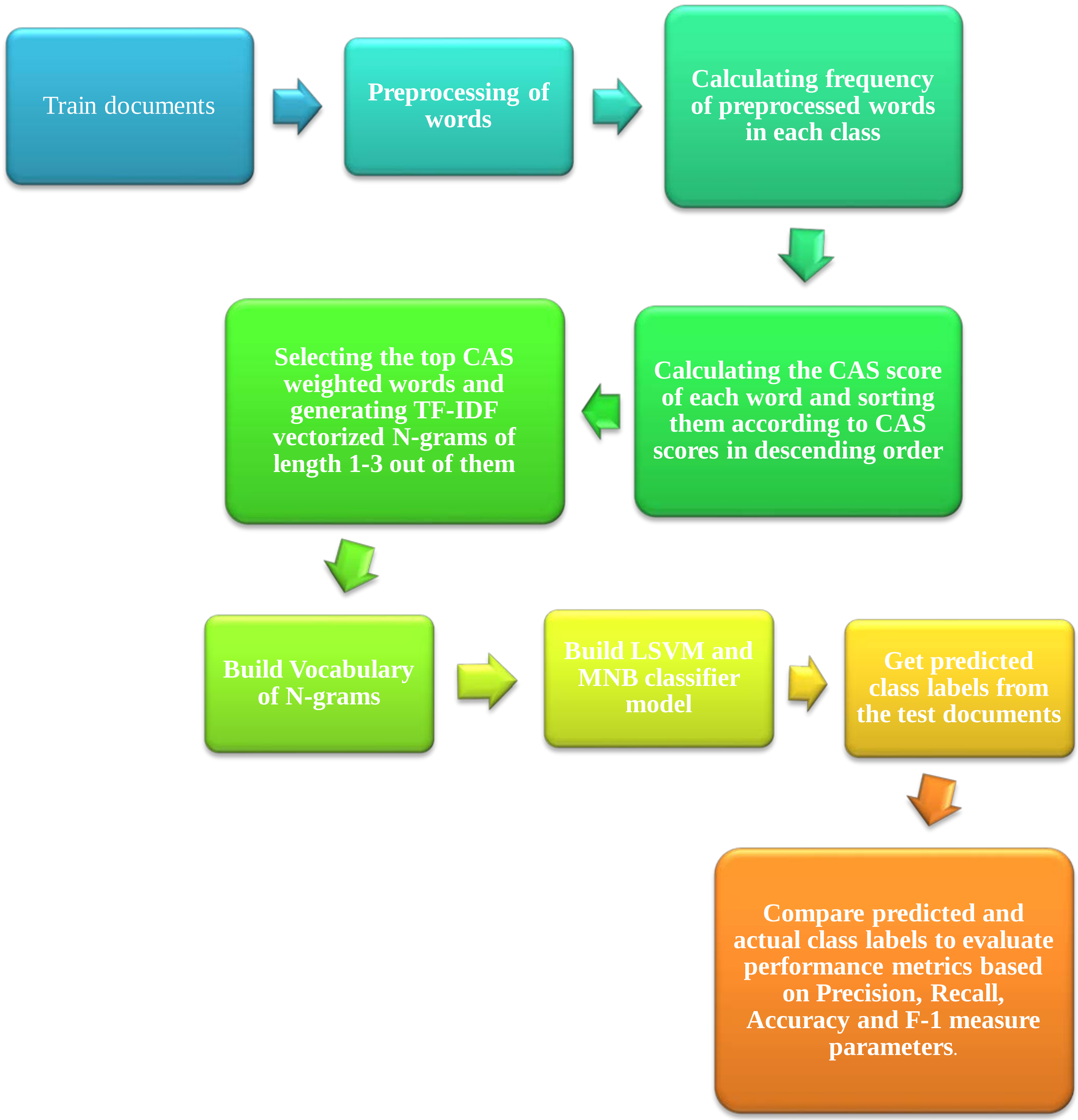
- **Feature extraction:** Feature selection **improves the performance** and **accelerate the training speed** of the classifiers. It reduces a huge feature space into a smaller subset. Let us define, p as the total number of words in the corpus, and n as the total number of documents. Subsequently, the text contents of the entire corpus D is extracted as tokens p and kept in a set t. Let $\boldsymbol{t} = [\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_p}]$**, where** $\boldsymbol{p > 0}$ and each word contains some information to discriminate the **class label of the documents**. The selection of words $\boldsymbol{t_q} \in \boldsymbol{t}$ that contain the **maximum information to discriminate a class label** which helps **in correct classification of the documents** is known as feature selection (***Agnihotri, D., Verma, K., & Tripathi, P. (2016). Computing symmetrical strength of n-grams: a two pass filtering approach in automatic classification of text documents. SpringerPlus,2016).***

- **Text document classification**: In text document classification, the documents set **(*y* = [*d*1, *d*2, …, *dj*])** of a *r* number of classes ***C* = [*C*1,*C*2, …,*Cr*]** is divided into two subsets, i.e. training (*training_doc*) and test (*test_doc*). The objective of the classification is to build a model based on the known class labels of training set documents which have the capability to predict the class labels of test documents with maximum accuracy ***(Manning et al. 2008; Sebastiani 2002; Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features, Springer Berlin).***

Problem Statement:

Automatic classification of text documents based on the correlative association score (CAS) of words: Substantial works were carried out in the area of feature selection to improve the prediction capability of the classifiers. The standard methods, viz. **Mutual Information (MI), Information Gain (IG), Discriminating Feature Selection (DFS) and Chi Square (CHI)**, did not consider positive and negative nature of the words that affects the performance of the classifiers. **To address this issue, a new feature selection method named Correlative Association Score (CAS) is proposed**. CAS **combines the strength, mutual information, and strong association of the words** to determine the positive and negative nature of the words for the class.

Processing flow of the proposed work:



Algorithm for computation of Correlative Association Score (CAS) of terms

Declaration:

- Input is a set 'data' of documents of each class C_k € C. y=[d_1,d_2,……..,d_n], where n>0, C=[C_1,C_2,…….,C_j], where j>0, j<=r
- The output of the algorithm is a set of most informative words t[k]⊂t[m]⊂t[p].

Procedure:

- y=training_doc+test_doc , where training_doc is the training set corpus and test_doc is the test corpus.
- First Preprocessing of text documents is done as follows:
  Function Preprocessing (data):
  T=**[t_1, t_2,………,t_p]** **←Tokenizer**
  T=stop words removal (T)
  T=punctuation marks removal (T)
  T= digits removal (T)
  T**← [t_1, t_2,………..,t_m]=**White Space Removal(T), where m<p
  return (T)
- t=Preprocessing(training_doc) ←To get the list of preprocessed words for all the training documents
- Calculation of the occurring frequency of the i^{th} word in the j^{th} class.
  tf_i is the set of occurring frequencies of words in the jth class, where i=1,2,……r
- Calculating the strength (weight) $\mathbf{W_{1j}}$ **,** likelihood $\mathbf{W_{2j}}$ and association of words $\mathbf{W_{3j}}$ of i^{th} word for j^{th} class.
- The resultant weight $\mathbf{W_{CAS}}$ of each word $\mathbf{t_i}$ is calculated as :
  $\mathbf{W_{CAS}=\log\left(\sum_{j=1}^{j=r}(W_{1j}+W_{2j})^3\right)+\sum_{j=1}^{j=r}(W_{3j}{}^4)}$
- Words are sorted in descending order according to the calculated CAS score of each word: **FS[m]→ Sort (t_i,W_{CAS})**

- **FS[p]→ Select (t_i,W_{CAS}), where p<m –**Top p CAS weighted words are selected.

Detailed Description of Steps:

- The dataset used for classification consists of **612** records and five attributes namely **'Incident Number','Brief Descripion','Hazardous Element','Initiaing mechanism'** and **'Accident/incident'.**

- First the entire excel dataset is read as a **dataframe** by importing pandas library. Information stored in the entire dataset is then converted to lowercase using lambda function. Then rows of the entire dataset has been **shuffled** for better distribution of the documents of training_doc and test_doc. After shuffling ,first **460** records has been placed in dataframe training_doc and next **152** records has been placed in dataframe test_doc . Text data contained under the attribute '**Brief Description'** has been used to invoke the function Preprocessing i.e. this data has been used to calculate the CAS scores of words. The data stored under the attribute of '**Hazardous Element'** has been used as target data for classification. Also, classification has been performed separately by taking the data stored under the attributes of '**Initiaing mechanism'** and **'Accident/incident'** as target data.

- Function Preprocessing has been defined which takes in data under the attribute 'Brief Description' and then **removes punctuations, stop words(**defined in python natural language tool kit**), digits, spaces** and **special characters** from the text data and returns a list of clean text words.

- The training_doc is converted to a matrix containing data of attributes **'Brief Descripion','Hazardous Element','Initiaing mechanism'** and **'Accident/incident'** for better accessibility of data. Now, first taking **'Hazardous Element'** as target data which contains **eight different categories (Classes)** of elements, **word frequency tf_i** for each class has been calculated. **tf_i** is a 2D list containing eight lists of word frequencies. A dataframe of words **(set(t)→list of sorted words after preprocessing)** and their frequencies in each class is created.

- The following probabilities and word counts are calculated as described:

Notations	Value	Meaning
a	*count (ti,Cj)*	count of word *ti* in the documents of class *Cj*
b	$count\ (ti, \overline{Cj})$	count of word *ti* in the documents of other classes $\overline{Cj}$
c	$count\ (\overline{ti}, Cj)$	count of other words $\overline{ti}$ in the documents of class *Cj*
d	$count\ (\overline{ti}, \overline{Cj})$	count of other words $\overline{ti}$ in the documents of other classes $\overline{Cj}$
N	*(a + b + c + d)*	total number of words in all *r* numbers of classes
df	*df (ti,Cj)*	document frequency of word *ti* in class *Cj*
maxf	*max(ti,Cj)*	maximum frequency of word *ti* in class *Cj*
avgf	*mean(ti,Cj)*	average frequency of word *ti* in class *Cj*
p(ti)	*(a + b)/N*	The probability of word *ti*
$p(\overline{ti})$	*(c + d)/N*	The probability of other words $\overline{ti}$
p(Cj)	*(a + c)/N*	The probability of class *Cj*
$p(\overline{Cj})$	*(b + d)/N*	The probability of other classes $\overline{Cj}$
p(ti,Cj)	*a/N*	The probability of word *ti* for being in class *Cj*
$p(ti, \overline{Cj})$	*b/N*	The probability of other words $\overline{ti}$ for being in class *Cj*
$p(\overline{ti}, Cj)$	*c/N*	The probability of word *ti* for being in other classes $\overline{Cj}$
$p(\overline{ti}, \overline{Cj})$	*d/N*	The probability of other words $\overline{ti}$ for being in other classes
p(ti |Cj)	*a/(a + c)*	The probability of word *ti* when class *Cj* is present
$p(\overline{ti} \mid Cj)$	*c/(a + c)*	The probability of word *ti* when other classes $\overline{Cj}$ are present
$p(ti \mid \overline{Cj})$	*b/(b + d)*	The probability of other words $\overline{ti}$ when class *Cj* is present
$p(\overline{ti} \mid \overline{Cj})$	*d/(b + d)*	The probability of other words $\overline{ti}$ when other classes $\overline{Cj}$ are present
p(Cj |ti)	*a/(a + b)*	The probability of class *Cj* when word *ti* is present
$p(\overline{Cj} \mid ti)$	*b/(a + b)*	The probability of other classes $\overline{Cj}$ when word *ti* is present
$p(Cj \mid \overline{ti})$	*c/(c + d)*	The probability of class *Cj* when other words $\overline{ti}$ are present
$p(\overline{Cj} \mid \overline{ti})$	*d/(c + d)*	The probability of other classes $\overline{Cj}$ when other words $\overline{ti}$ are present

- **Explanation of CAS**: The CAS extracts a set of *m* most discriminating words from the set of all words using a threshold value. It computes weight W_{CAS} of each word *ti* as follows:

  a. The CAS computes a unique weight of each word on the basis of three criteria, first criterion computes weight $\boldsymbol{W_{1j}}$ to measure the strength of *ith* word $\boldsymbol{t_i}$ for *jth* class C_j **(i.e.** $\boldsymbol{W_1(t_i, C_j)}$**),** second criterion computes weight W_{2j} **to measure the likelihood of class** C_j **when the word** t_i **is present (i.e.** $W_2(t_i|C_j)$**),** and third criterion computes **weight** W_{3j} **of the word** t_i **to measure the association of** t_i **with class** C_j **(i.e.** $W_3(t_i, C_j)$**).** The resultant weight (W_{CAS}) of the word *ti* is computed as,

$$W_{CAS} = \log\left(\sum_{j=1}^{j=r} (W_{1j}+W_{2j})^3\right) + \sum_{j=1}^{j=r} (W_{3j}^{\ 4})$$

Where, $$W_1(t_i, C_j) = \frac{\text{maxf}\ (ti, Cj)}{\text{avgf}\ (ti, Cj)} + \log_2\left[\frac{ad}{bc}\right]$$

$$W_2(t_i|C_j) = a \times \log\frac{p(ti, Cj)}{p(tj) \times p(Cj)} + b \times \log\frac{p(ti, \overline{Cj})}{p(tj) \times p(\overline{Cj})}$$

$$W_3(t_i, C_j) = \left|\frac{a}{a+c+df[ti,j]} - \frac{c}{a+c+df[ti,j]}\right|$$

b. The list of **words of set(t)** is now sorted **in descending order** according to their calculated **CAS (**W_{CAS}**).**

c. **The topmost p** of CAS weighted words from **set(t)** are selected based on a threshold value (2000) and a **new list of top_words** is made consisting of those p number of words.

d. A list of **N-grams(of length 1-3) →NG[g]** ,g>p is created using the list top_words .Unigrams, bigrams and trigrams are created separately and added to create the list **NG**.

e. Now,**TF-IDF** of each **N-gram** is calculated. The word frequency ina document of a vector space is usually normalized using the **Term Frequency-Inverse Document Frequency(TF-IDF)** method. First a dictionary **TFDict** is created whose **keys** are all the unique N-grams and whose **values** are their corresponding **term frequencies.** The term frequency of each N-gram is calculated as :

tf=(frequency of the term in the document/total number of terms in the document)
Here each N-gram is considered to be a document and and as all the N-grams are unique frequency of each N-gram is one and total number of terms in each document refers to he length of each N-gram.

Now, another dictionary **IDFdict is created** whose keys are all the unique N-grams and whose values are their corresponding **idf.** The idf (the values of IDFdict) is calculated as **:**
idf=log(total number of documents/number of documents with term in it)
Total number of documents refers to the length of the NG list and due to unique N-grams list **number of** documents with an N-gram always remains 1.

Lastly, the **TF-IDF** is calculated by **multiplying** the **tf and idf** for each **unique N-gram**.

f. The **NG list** is sorted in descending order according to the TF-IDF scores of each unique N-gram.

g. Now, **a new N-grams(of length 1-3) list (NG_l)** is created for each record of the training_doc dataframe under attribute **'Brief Description'** by using the lists of preprocessed words obtained by invoking the function Preprocessing. **NG_l is a 2D list** which contains lists of N-grams for each of the records. Then each of the unique **N-grams** contained **in the sorted list** of **NG**[b] (let b=5000) is checked with each list of N-grams **contained in the NG_l** list. If the N-grams match, then they are appended **(record wise)** into **another list called NG_new (2D list containing N-grams record wise).**

h. For training the model and for getting a better accuracy **N-grams (of length 1-3) are again made** out of the terms stored in **NG_new** list for each record and **stored in a list NG_l1 (2D list containing new lists of N-grams which will be used for training the model)**.

i. Now, the **LSVM and MNB** classifiers are used for training the model and predicting the output.

j. All the above steps are repeated using the categories under each of the attributes of **'Accident/incident' and 'Initiaing mechanism'** as target data.

Results and Evaluation

The **LSVM and MNB** classifiers of python **scikit-learn package** are used for classifying the model. **The pipeline class** of sklearn package is used **to sequentially apply** the list of transforms required and a final estimator. Intermediate steps of pipeline allow us to **implement fit and transform** methods and the final estimator only needs to implement fit.

The measures used to evaluate the performance of LSVM and MNB are **Precision, Accuracy, Recall, f1-score** (these are the parameters used for multiclass classification).

1. Using categories under the attribute **'Hazardous Element'** as classes for classification, we get the following results:

		Precision	**Recall**	**f1-score**	**Support**
LSVM	micro avg	0.79	0.79	0.79	152
	macro avg	0.38	0.40	0.38	152
	weighted avg	0.73	0.79	0.75	152

LSVM accuracy score=0.7894736842105

		Precision	Recall	f1-score	Support
MNB	micro avg	0.64	0.64	0.64	152
	macro avg	0.29	0.22	0.19	152
	weighted avg	0.64	0.64	0.54	152

MNB accuracy score=0.64473684210526

2. Using categories under the attribute **'Accident/incident'** as classes for classification, we get the following results:

		Precision	Recall	f1-score	Support
LSVM	micro avg	0.60	0.60	0.60	152
	macro avg	0.41	0.33	0.34	152
	weighted avg	0.57	0.60	0.56	152

LSVM accuracy score=0.5986842105263

		Precision	Recall	f1-score	Support
MNB	micro avg	0.45	0.45	0.45	152
	macro avg	0.13	0.15	0.12	152
	weighted avg	0.34	0.45	0.34	152

MNB accuracy score=0.4539473684210

3. Using categories under the attribute **'Initiating mechanism'** as classes for classification, we get the following results:

		Precision	**Recall**	**f1-score**	**Support**
LSVM	micro avg	0.37	0.37	0.37	152
	macro avg	0.13	0.16	0.14	152
	weighted avg	0.31	0.37	0.34	152

LSVM accuracy score= 0.3684210526315789

		Precision	**Recall**	**f1-score**	**Support**
MNB	micro avg	0.38	0.38	0.38	152
	macro avg	0.13	0.15	0.12	152
	weighted avg	0.30	0.38	0.31	152

MNB accuracy score= 0.3815789473684211