Chapter 5: Text Summarization with Naïve Bayes

This chapter discusses a Text summarization model which has been designed and implemented using the Naïve Bayes Classifier. The initial section delibrates on classification in general and the Bayes theorem. It is followed by the variants of the Naïve Bayes clasification. Then the actual model design, its implementation and the results are discussed in detail.

5.1 Introduction

The internet is flooded with huge amount of information. Therefore, it is always difficult to find relevant information from this vast sea of data. With the help of Text Summarization, relevant information can be derived in the form of a summary. In the past, researchers have worked with models which were graph based or statistical based to accomplish Text Summarization. In recent times, researchers have started to explore other than traditional ways and have started working on machine learning and deep learning techniques for generating summaries. These methods can build summaries automatically without human intervention.

Initially a number of machine learning techniques were used to summarize the essential information in human-readable form, thus helping people to analyze the documents from which the data originates (Cristianini & Taylor 2000, Mani et al 2002, Chen et al 2005). A large portion of these documents must be retrieved for analysis using good information retrieval models. Technical domain and information technology experts must be closely involved with every step of the information retrieval and extraction processes, thus enabling users to get what they want quickly and correctly.

Text summarization extracts sentences based on the calculation of the score and rank from the input document. In the proposed work of this research, the model that has been developed uses Latent Semantic Analysis technique and chooses sentences based on specific threshold given by the system. Further, using Naïve Bayes approach of Machine Learning, the model trains the classifier

and predicts the summary that is built on the basis of calculation of Singular Value Decomposition (SVD). Before training the model, it selects two important concepts of SVD - feature ranking and recursive feature elimination. Before describing the proposed model it is important to discuss classification and some of its different approaches which has been done in the next few sections.

5.2 Classification in Machine Learning

Classification is a supervised learning approach, where a classifier model is learnt through a set of training data (Mitchell 1997). Classification, also popularly known as Categorization, takes as input data objects which can be structured or unstructured and based on predefined knowledge of the objects is a data mining and knowledge management technique, used in grouping similar data objects together (Han & Kamber 2001). The reason why it is called supervised learning is because it assigns class labels to data objects which do not have class labels, based on the association between the data items which have pre-defined class labels. Figure 5.1 shows a typical classification framework.

Given a collection of training sets, each of which contains a set of attributes and one of the attributes is the class, classification is the task of finding a model for a class attribute as a function of the values of other attributes. The main goal is to assign a class as accurately as possible to previously unseen sets of data.

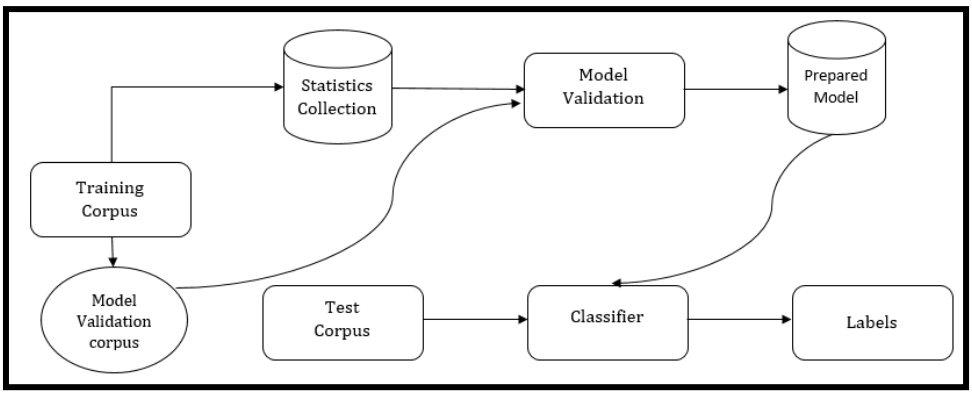

Figure 5-1 Typical supervised learning framework

The training data sets have number of features i.e. attributes out of which one is considered to be the class label. Once the classifier is developed, it is used to predict the class of data sets which do not have pre-defined classes.

Designing a classifier first is very important and there are a number of metrics which describe the goodness or the efficiency of the classifier.

Machine Learning techniques can be used for summarizing data. Conceptual clustering try to group data with similar features in same clusters and this generalized data may be both understandable by people and usable by the computer for answering high-level questions (Kolodner 1984). These clusters are many times used as the training data sets for Classification techniques where we need to predict the class of data sets which do not have pre-defined classes i.e. the test datasets. The datasets in this work are the sentences of a document whose summary we are trying to generate.

There are a number of classification methods each with its own set of modeling capability. Amongst the popular ones are decision tree classifiers, rule-based classifiers, neural networks, support vector machines and Naïve Bayes classifiers. Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by the learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before.
While designing the classifier model and evaluating its efficiency, it is important that certain techniques are utilized to appraise the model using the training dataset in such a way that it is first divided into two parts – training and testing. The testing dataset over here now has two sets of classes – the actual and the predicted. Using these metrics the correctness of the classifier can be evaluated. There are a number of methods like – the hold-out and random subsampling method, cross-validation and bootstrap methods are the most popular ones. In each one of them the training set is initially divided into training and testing and then the model is evaluated. In each case the division manner and method is different.

In this proposed model, latent semantic analysis has been used to analyze the relationship between sentences and the words, which includes the generated concepts that is related to words and sentences. After the concepts are generated, the Naïve Bayes classifier has been used to predict the summary. Instead of considering all the features, initially before using the classifier, recursive feature elimination has been used to eliminate features as per the

importance of the words and sentences in the given document. By doing this, it can help the model to avoid overfitting and increase the performance in terms of accuracy and time.

5.2.1 Classification - Text Handling

Till the last decade it has been observed that classification techniques were mainly used for structured data. It is also perceived that the incorporation of linkage information into the classification process, can significantly improve the quality of the underlying results (Shilpa & Neeraj 2012). As compared to other sophisticated models, classification models are simpler, can be quickly generated, and are extremely useful tools for many practical data mining applications, where both predictive accuracy and the ability to analyze the model are important. Almost all the known techniques for classification such as decision trees, rule based, Bayesian methods, nearest neighbour classifiers, SVM classifiers, and neural networks have been extended to handle text data.

It has been noticed that some authors (Lertnattee & Theeramunkong 2004) have also proposed to parallelize and distribute the process of text classification in increase the speed of execution. Due to this approach on classification, the performance of classifiers can be improved in both accuracy and time complexity. It is now a lucrative area of research by including Machine Learning along with the traditional methods of data mining and moving towards a new direction for the improvement of the performance of individual classifiers (Bao & Ishii 2002).

5.2.2 Classifying Summary Sentences

Classification techniques have been found to be successful in systems that capture knowledge, and this makes it easier to mechanize tasks that are already successfully performed by humans, although genetic algorithms, neural networks, fuzzy logic, etc, can perform well. We have looked at the perspective of modeling the task of summarization as a classification problem, where sentences in a document are classified into one of the two classes, ‘summarize’ or ‘not summarize’. In the proposed work we employ a classification based view of text summarization, in a way we can exploit the prediction capabilities of the Naïve

Bayes Classifier. The performance of this classifier model in classifying a given sentence as a summary sentence or not, is evaluated in terms of accuracy, precision, recall and F-Score. Before actually discussing the model and the output it is important to understand the Naïve Bayes Classifier and its base which is the Bayes Theorem.

5.3 The Naïve Bayes Classifier

Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes' probability theorem, are known for creating simple yet well performing models, especially in the fields of document classification and document summarization. Naive Bayes classifiers are linear classifiers that are known for being simple yet very efficient. The adjective Naïve in the probabilistic model of Naïve Bayes classifiers comes from the assumption that the features in a dataset are mutually independent. In practice, the independence assumption is often violated, but Naïve Bayes classifiers still tend to perform very well under this unrealistic assumption.

Being relatively robust, easy to implement, fast, and accurate, this classifier is used in many different fields. Apart from Text Mining, the applications also include the diagnosis of diseases and making decisions about treatment processes, the classification of RNA sequences in taxonomic studies], and spam filtering in e-mail clients. It is a graphical model where it shows the relationship between features.

5.3.1 The Bayes Theorem

In order to understand how Naïve Bayes classifiers work, we have to briefly recapitulate the concept of Bayes' rule. The probability model that was formulated by Thomas Bayes (1701-1761) is quite simple yet powerful; it can be written down in simple words as follows:

$$\text{posterior probability} = \frac{\text{conditional probability}\,.\ \text{prior probability}}{\text{evidence}} \qquad (5.1)$$

In simple terms, a Naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3

inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naïve'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods. The Bayes theorem as shown in Equation 5.1 can be written as:

$$P(\frac{A}{B}) = \frac{P(\frac{B}{A}) \, . \, P(A)}{P(B)} \tag{5.2}$$

Where,

P(A|B) : It is known as posterior probability, the conditional probability that event A occurs , given that B has occurred.Over here, A is our hypothesis and B is the test data. It means we are trying to find the probability that the given object B belongs to class A.

P (A) and P (B) are the probabilities of events A and B.

P (B|A): It is the conditional probability of event B occurring given that event A has occurred.

The right hand side of the equation is based on the training data, which already has pre-defined classes. In addition, using this information we are trying to find the class of the test data given on the left hand side of the equation.

In a simple machine learning approach, the model learns from attributes that is provided as independent random variables. To be precise, B denotes the attribute set and A denotes the class variable. If the class variable has a non-deterministic relationship with the attributes, then we can treat A and B as random variables and capture their relationship probabilistically using P(A/B).

There are two versions of Naïve Bayes algorithm. One is the Bernoulli model that only takes into account the presence or absence of a particular term, so it does not capture the number of occurrence of each word. The other model is the multinomial model that captures the word frequency information in documents.

The detailed study and comparisons are given below. Both are based on the Bayes theorem, where a document or a sentence B is placed in the class A as per the probability given in Equation 5.2.

For classes the probability P(B) will remain same as it does not depend on any class and hence can be ignored. So, the posterior probability now becomes,

$$\mathrm{P}\left(\frac{\mathrm{A}}{\mathrm{B}}\right) = \left(\mathrm{P}\left(\frac{\mathrm{B}}{\mathrm{A}}\right).\ \mathrm{P}(\mathrm{A})\right) \tag{5.3}$$

Since we are dealing with terms (which are the sentences) within a document the posterior probability P(B/A), is actually $P((t^1,t^2, \ldots t^n)/A)$ where each t_i is a sentence or a feature of the document. In the Naïve Bayes approach we assume each term occurrence is conditionally independent and that is why the name Naïve. It is a simple assumption because of which of posterior probability $P((t^1,t^2, \ldots t^n)/A)$now becomes,

$$P(B/A) = P((t^1, t^2, \ldots\ldots, t^{nd})/A) = P(t^1/A)P(t^2/A)\ldots.P(t^{nd}/A) \tag{5.4}$$

$$P(B/A) = \prod_{1 \le k \le n} P(t^k/A) \tag{5.5}$$

In the above formula,

$P(t^k/A)$ - is the conditional probability of term t^k occurring in class A,

P(A) - is the prior probability of a sentence occurring in class A.

t^1, t^2, … ,t^{nd} - are the terms(tokens) of document d that are part of the vocabulary.

n^d – total number of tokens in the document.

$\prod_{1 \le k \le n} P(t^k/A)$- is the multiplication of conditional probabilities of all terms in the document.

5.3.2 The Multinomial Model

This model depends on the term counts in a document i.e. the number of times a term occurs in a document. The position of the term is not considered.

As per (5.3) and (5.5), the posterior probability becomes,

$$P(A/B) = \mathrm{P}(\mathrm{A}) \prod_{1 \le \mathrm{k} \le \mathrm{n}} \mathrm{P}(\mathrm{tk}/\mathrm{A}) \tag{5.6}$$

$\prod_{1\leq k\leq n} \mathrm{P}\left(\frac{\mathrm{tk}}{A}\right)$can have many terms and the multiplication of the probabilities can result in floating point underflow. Therefore, a better option is to use logarithms. After applying logs, (5.6) becomes,

$$P(A/B) = \log \mathrm{P(A)} + \sum_{1\leq k\leq nd} \log \mathrm{P(tk/A)} \tag{5.7}$$

In the multinomial model to find the probabilities on the RHS of (5.7) we use the maximum likelihood estimate i.e. the relative frequencies of occurrences. As per this estimate, the probability P(A/B) can be calculated by using the following frequencies:

$$\mathrm{P(A)= N_A /N} \tag{5.8}$$

Where,

N_A – total number of sentences/documents in class A

N – total number of sentences/documents

$$P(t/A) = \frac{\mathrm{TAt}}{\sum_{t'\epsilon V} \mathrm{TAt'}} \tag{5.9}$$

Where,

TAt – total number of occurrences of term t in sentences/documents of class A

$\sum_{t'\epsilon V} \mathrm{TAt'}$ - total number of terms in all sentences/documents of class A

In (5.9) the probability becomes zero if a term does not occur in a class and if it is applied to (5.6) it will become multiplication by zero. To eliminate this problem, Laplace's smoothing is used as follows:

$$P(t/A) = \frac{\mathrm{TAt}+1}{\sum_{t'\epsilon V}(\mathrm{TAt'}+1)} = \frac{\mathrm{TAt}+1}{\sum_{t'\epsilon V} \mathrm{TAt'} + \mathrm{V}} \tag{5.10}$$

$\sum_{t'\epsilon V} (1)$ - this is equal to V the size of the vocabulary.

The formula in (5.6) using (5.8) and (5.9) becomes,

$$P(A/B) \propto \frac{\mathrm{NA}}{\mathrm{N}} . \prod \frac{\mathrm{TAt} + 1}{\sum_{t' \epsilon V} \mathrm{TAt}' + \mathrm{V}} \tag{56.11}$$

We can now find P(A/B) the probabilities for the document for each class and the document will belong to the class where this probability is maximum. Therefore, our aim is to find max. P(A/B).

5.3.3 The Bernoulli Model

In this model, unlike the multinomial model instead of counting the number of times terms/tokens occur in a document, the presence or absence of a term (0 or 1) is noted. The P(t/A) estimates the fraction of documents of class A that contain term t.

The implementation and analysis shows that the Bernoulli model works well for short documents only. The main drawback being that since just the presence of a term is noted, a document which contains a certain term only once may get classified in a class to which it actually does not depend.

This model is also based on the formula mentioned in (5.7) and (5.11). The difference lies in the estimation strategies. Unlike Multinomial in the Bernoulli model the absence of a term is also modeled while computing the probabilities. The estimates for priors P(A) are similar to (5.8), whereas there is a little difference in estimates for (5.10) which is given below:

TAt – total number of documents of class a where term t occurs

$\sum_{t' \epsilon V} \mathrm{TAt}'$- total number of documents of class A

V – two cases to be considered for each term i.e. occurrence or non-occurrence

The final equation of (5.11) now becomes,

$$P\left(\frac{A}{B}\right) \propto \frac{\mathrm{NA}}{\mathrm{N}} . \left[P\left(\frac{t1}{A}\right) . P\left(\frac{t2}{A}\right) \dots P\left(\frac{tx}{A}\right)\right] . \left[1 - P\left(\frac{ty}{A}\right) . 1 - P\left(\frac{ty+1}{A}\right) \dots 1 - P\left(\frac{tm}{A}\right)\right] \tag{5.12}$$

In (5.12), the terms t1… tx occur in the document whereas the terms ty … tm do not occur in the document. The document B is placed in the class which has the highest probability using (5.12).

5.3.4 Comparison of the Multinomial and Bernoulli Models

The comparison between both the models considering different aspects is shown in Table 5.1.

Table 5-1 Comparison between Multinomial model and Bernoulli model

Parameters	**Multinomial Model**	**Bernoulli Model**
Variable used	t – number of times a term occurs in a document	t – 1 if term occurs, 0 if it does not in a document
Document representation	d = {t1, t2, … , tn} where t1, t2,…tn terms are occurring in document d and are also part of V(vocabulary) – terms in d but not part of V are not considered	d = {t1, t2, … , tn} where each tk ϵ {0,1} – indicates the presence or absence of a term in the document d
Multiple term occurrences consideration	Yes	No
Efficiency (Document size)	Both short and long	Only short
No. of features handled	Efficient with more	Works best with few
Estimate for the term ‘the’	P(X = the / c) 0.5	$P(U_{the} = 1 / c) = 1.0$

Naïve Bayes classifier because of its simple assumption that the features in a dataset are mutually independent, works very well for different problems such as medical and machine learning applications. The Naïve Bayes requires a small number of training set for estimating the outcome for classification. In fact, for text classification, Naïve Bayes classifiers have shown tremendous performance.

5.4 The Proposed Model

The proposed model has been developed for single document summarization and uses Latent Semantic Analysis technique choosing sentences with weights based on specific threshold given by the system. Further, using Naïve Bayes approach of Machine Learning, the model trains the classifier and predicts the summary that is built on the basis of calculation of Singular Value Decomposition (SVD). Before training the model, there are two major steps involving the concepts of SVD - feature ranking and recursive feature elimination.

This model focuses on extractive text summarization using Machine Learning, statistical techniques and Latent Semantic Analysis. The model is depicted in Figure5.2.

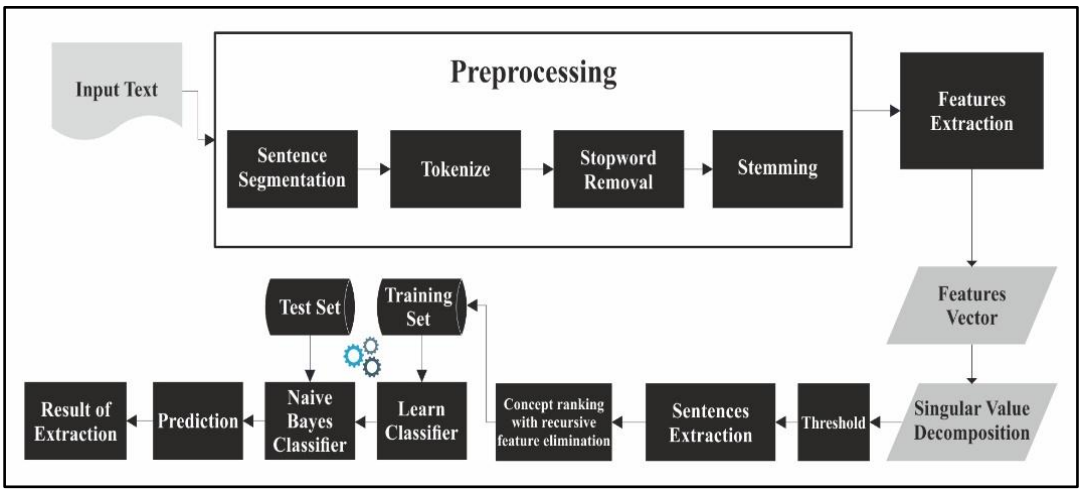

Figure 5-2 Proposed Model

5.4.1 The Dataset description

For implementation of the above mentioned model, datasets which are generally used in literature for empirical evaluation of summarization are selected from the DUC 2004 repository. There is currently much interest and activity aimed at building powerful multi-purpose information systems. The agencies involved include DARPA, ARDA and NIST. Their programmes, for example DARPA's TIDES (Translingual Information Detection Extraction and Summarization) programme, ARDA's Advanced Question & Answering Program and NIST's TREC (Text Retrieval Conferences) programme cover a range of sub-programmes. These focus on different tasks requiring their own evaluation designs.

Out of the initial workshop and the road mapping effort has grown a continuing evaluation in the area of text summarization called the Document Understanding Conferences (DUC). Sponsored by the Advanced Research and Development Activity (ARDA), the conference series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments. DUC is an evaluation series, organized by NIST and supported by Defense Advanced Research Projects Agency (DARPA), in the area of automatic text summarization. NIST selected 50 English document clusters from the TDT collection, 25 Arabic document clusters from the TDT collection and 50 English document clusters from the TREC collection. Each TDT or TREC cluster selected contains on an average 10 documents. Also, NIST provided questions for the 50 TREC clusters. There were an average of 20 numbers of sentences as minimum number of sentences; and 56 as maximum number of sentences per document.

The whole model was developed using Python Version 3.6.5. Python is an interpreted, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability notably using significant white space. It provides constructs that enable clear programming on both small and large scales. Further the Machine Learning Repository : scikit-learn which is a simple and an efficient tool for data mining and data analysis was also used. It is accessible to everyone and reusable in various contexts. It is built on Numpy, Scipy and matplotlib. Moreover, it is an open source which comes under BSD license.

The steps of the implementation of the model have been explained in detail in subsequent sections.

5.4.2 Pre-processing the Data

As can be seen in the model diagram of Figure 5.2, once a document is taken as input the next step is pre-processing which in itself contains a number of sub-processes. Each step in pre-processing is explained as given below.

i. Read as input the text file for which summary is to be generated.

ii. Pre-process the file:

   a. Remove all unnecessary characters. In this step, all unnecessary characters like punctuations, symbols will be removed.

b. Split each word by sentence – segmentation. In segmentation, it is the task where text is divided into word, unit, or topic.
c. Remove all stop words.
d. Convert all word into lower case.
e. Perform stemming on each word using Porter Stemmer.

The detail about stop words removal and stemming has already been discussed in Chapter-3.

5.4.3 Feature Extraction

The next important step is feature extraction.

i. Occurrence of a word in a file.

It is known as term-sentence matrix. If it is for multiple documents it is called a term-document matrix. A mathematical matrix explains the occurrence of term in a collection of sentences. Word (or n-gram) frequencies are typical units of analysis when working with text collections. Over here the sentences are converted to a matrix of token counts. During processing, if the system or the model does not provide a-prior dictionary and analyzer then system can use feature selection as equal to vocabulary size for analyzing the data. When preparing a matrix, rows represent the document/sentences and columns represent terms.

ii. From occurrence to frequencies (tf-idf).

Term frequency-inverse document frequency, is a numerical method to understand importance of a word in a corpus. The tf-idf value depends on the number of times a term appears in the document/sentence, but is often compensated for by the frequency of the word in the corpus, which helps to adjust the fact that some words appear more frequently in general. Different types of tf-idf weighting methods are used for scoring and ranking a document. The formula for generating the tf-idf scores has been discussed in detail before in the pre-processing chapter.

5.4.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is one of the statistical techniques to analyze relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. It is assumed that words having similar meaning will occur in similar pieces of text. Detailed discussion of LSA has been covered in Chapter – 4.

5.4.5 Threshold Generation

After performing SVD (This is with reference to the explanation on SVD as given in Chapter – 4), the V^T matrix, the matrix of extracting sentences X concepts is used for picking up the significant sentences. In V^T matrix, row represents the importance of concepts. The cell values demonstrate the relationship between the sentences and the concepts. A higher value indicates that the sentence is more relevant to that concept.

The sentences are marked with 1 or 0 with a specific threshold for prediction of extracting sentences. The threshold has been decided by comparing the importance of sentences as given in the human and pre-defined summary.

```
For each generated concept
        Begin
                Check whether cell valuehas the minimum
                threshold
                If yes
                        Set the sentence as 1 for summarization
                Else
                        Set the sentence as 0
        End
Endfor
```

Figure 5-3 Threshold selection method

As per Figure 5, for each concept the process is performed till we reach a threshold which generates the best summary sentences.

5.5 Recursive Feature Elimination

The textual dataset is very high dimensional in nature. It contains many attributes. When we use such dataset for classification or regression, not all attributes are important and have equal weightage. The vocabulary list is also

quite large. Moreover, not all attributes contribute in predicting the outcome. Therefore, removing a few attributes can boost the model's performance and accuracy. In addition, it also avoids the chance of overfitting and training time can be reduced for training of large datasets. Attributes are the features or in plain language the important terms in sentences which would be selected as part of the summary.

As the name suggests, recursive feature elimination, it removes the feature recursively and builds the model with remaining features. With the help of Recursive Feature Elimination (RFE), a group of attributes are selected, which contribute for prediction of the target variable which is the summary.

In Recursive Feature Elimination algorithm, the algorithm tries to fit all the features. Each feature is assigned the scoring according to its importance to the model. Let D is the order number of sequence of the features to be retained (where, D1 > D2,…). At every iteration of the feature selection, the top most Di ranked features are retained and again the model iterates after refitting. After every iteration, the model measures the performance. The score of D_i of each feature is determined and as per requirement, top most features are used to train the model. The features as explained before are the important sentences.

1. Train the model on the training set using all features
2. Calculate model performance
3. Calculate variable importance or rankings
4. for *Each subset size* $D_i = 1 \ldots D$ do
5. Keep the D_i most important variable
6. Pre-process the data [optional]
7. Train the model on the training set using D_i predictor
8. Calculate model performance
9. Recalculate the rankings for each predictor
10. end
11. Calculate the performance profile over the D_i
12. Determine the appropriate number of predictor
13. Use the model corresponding to the optimal D_i

Figure 5-4 Algorithm of the Recursive Feature Elimination

5.6 Classification Using Naïve Bayes

This is the next important step in the model after the recursive elimination has been completed. The working of the Naïve Bayes classifier has been discussed in detail in the previous section. Using the sentences which have been selected in the previous step, and considering them as the training data the classifier is generated.

The dataset, DUC 2004, as explained in detail before contains 50 document clusters and each cluster having atleast 10 documents each. Before classifying the sentences for summary it is important a model is selected to divided the training and the test data to evaluate the performance of the classifier. After that the classifier is evaluated using the standard parameters to decide how well it is working. There are a number of methods for the performance evaluation along with the different metrics.

5.7 Model Evaluation

Once constructed, the model is used to classify unseen examples. There are different methods available for the same like – holdout and random sampling, cross-validation and the bootstrap methods.

In the holdout method the training dataset is randomly partitioned into two independent sets where the bigger set is the training and the smaller is the testing. This partitioning could be 80% training and 20% testing or 2/3rd training and 1/3rd testing. The training set is used to derive the model and the testing is used to check the accuracy. The selection is sometimes done using random sampling where the same is repeated a number of times by randomly selecting the training and the testind datasets.

Pros of the hold-out strategy: Fully independent data; only needs to be run once so has lower computational costs.

Cons of the hold-out strategy: Performance evaluation is subject to higher variance given the smaller size of the data.

For assessing the classifier accuracy k-fold cross validation technique is used. In k-fold cross validation, the initial data are randomly partitioned into k mutually exclusive subsets, and training and testing are performed k times. In

each of this iteration, k-1 blocks are used for training and the remaining block is used to test the performance of the algorithm. The accuracy estimate of the classifier is the average of the measures from the k iterations. It is common to choose k=10 or any other size depending mainly on the size of the original dataset.

Pros of the K-fold strategy: Prone to less variation because it uses the entire training set. Cons of the K-fold strategy: Higher computational costs; the model needs to be trained K times at the validation step (plus one more at the test step).

In the bootstrap method the the training datasets are selected uniformly with replacement. In this research work the holdout method has been used.

5.7.1 The Evaluation Measures

Text classification rules are typically evaluated using performance measures from information retrieval. Common metrics for text categorization evaluation include recall, precision, accuracy and error rate and F1 (F-score). Given a test set of N documents, a two-by-two contingency table with four cells can be constructed for each binary classification problem.

The cells contain the counts for true positive (TP), false positive (FP), true negative (TN) and false negative (FN), respectively. Clearly, N = TP + FP + TN + FN. These parameters have been used for the summary evaluation.

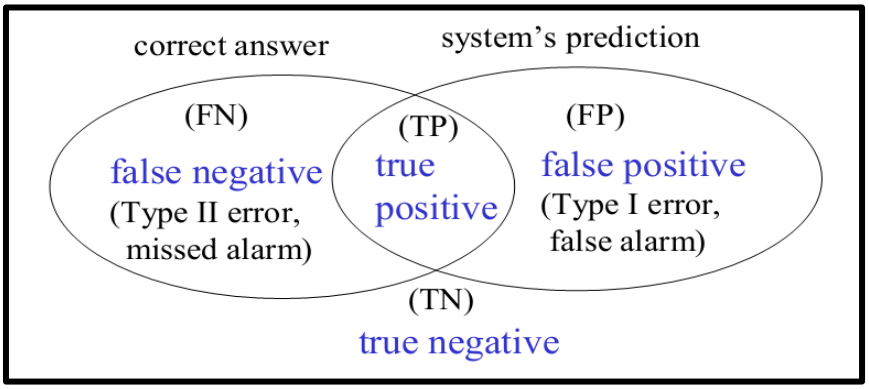

Figure 5-5 Common evaluation metrics

The metrics for binary-decisions are defined as:

- Accuracy
- Error
- Precision

- Recall
- F, F1 or F-score

Each measure is briefly described below.

Accuracy

Accuracy is used to find the correct classification rate. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. It is also known as the overall recognition rate of the classifier. Accuracy is effective when the class distribution is balanced. The equation of accuracy is given below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{N} \tag{5.13}$$

Error Rate

Error rate is also known as the misclassification rate. It gives the percentage of classifiers that were misclassified. While finding the error rate if only the training set is used and not the actual test set, the error estimate becomes optimistic and is also called the resubstitution error.

$$Error\ Rate = \frac{FP + FN}{TP + TN + FP + FN} = \frac{FP + FN}{N} \tag{5.14}$$

$$Error\ Rate = 1 - Accuracy \tag{5.15}$$

Precision

It is known as the measure of exactness. It considers that how many sentences are correctly predicted. It tells us how many sentences were classified as “Summarize” from the given set of sentences i.e. what percentage of tuples labeled as positive are actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{57.16}$$

Recall

Recall is a measure of completeness. It tells us what percentage from the actually positive tuples the classifier could classify them as positive. It is also called sensitivity or the true positive rate.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{5.17}$$

The way we have a true positive rate we also have a true negative rate which is called the specificity and is given by:

$$Specificity = \frac{TN}{TN + FP} = \frac{TN}{N} \tag{5.18}$$

Both precision and recall are incomplete measures as far as actually finding the accuracy and effectiveness of a classifier is considered. A perfect precision score of 1 for a class C means every tuple belonging to this class has been correctly classified as belonging to C but it does not tell us how many tuples were incorrectly labeled (i.e. it does not tell us about false negatives). Similarly, a perfect recall score of 1 for a class C means every tuple of C was labeled as such but it does not tell us about false positives.

This shows that there is an inverse relation between precision and recall and therefore a better measure becomes the F measure which is a harmonic mean of the two.(Han and Kamber 2013).

F-Score

It is a harmonic mean of precision and recall (Kohavi 1995). It is combining recall and precision score to calculate F-score. The equation of f-score as per below

$$F - Score = \frac{2 * precision * recall}{recall + precision} \tag{5.19}$$

- Harmonic mean of recall and precision
- Sometimes instead of multiplying by the numerator by 2, other parameters like ∝ or β are also used where each one of them has some integer value.

Micro-average F1

- global calculation of F1 regardless of topics

Macro-average F1

- average on F1 scores of all the topics

The formulas to find the different F measures are given in Table 5.3.

Table 5.3 The F measures for Microaveraging and Macroaveraging

	Microaveraging	**Macroaveraging**
Precision(π)	$\pi = \frac{\sum_{i=1}^{\|c\|} TP_i}{\sum_{i=1}^{\|c\|} TP_i + FP_i}$	$\pi = \frac{\sum_{i=1}^{\|c\|} \pi_i}{\|c\|} = \frac{\sum_{i=1}^{\|c\|} \frac{TP_i}{TP_i+FP_i}}{\|c\|}$
Recall(ρ)	$\rho = \frac{\sum_{i=1}^{\|c\|} TP_i}{\sum_{i=1}^{\|c\|} TP_i + FN_i}$	$\rho = \frac{\sum_{i=1}^{\|c\|} \rho_i}{\|c\|} = \frac{\sum_{i=1}^{\|c\|} \frac{TP_i}{TP_i+FN_i}}{\|c\|}$

Where c represents the class.

These scores can be computed for the binary decisions on each individual category first and then be averaged over categories (macro-averaging). They can also be computed globally over all the n x m binary decisions where n is the total number of test documents and m is the number of categories under consideration (micro-averaging).

The micro-averaged F1 tend to be dominated by the classifier's performance on common categories whereas the macro-averaged F1 are more influenced by the performance of rare categories.

Due to the often highly unbalance number of positive vs. negative examples, note that TN often dominates the accuracy and error of a system, leading to miss-interpretation of the results. For example, when the positive examples of a category constitute only 1% of the entire test set, a trivial classifier that makes negative predictions for all documents has an accuracy of 99%, or an error of 1%. However, such a system is useless. For this reason, recall, precision and F1 are more commonly used instead of accuracy and error in text categorization evaluations.

In multi-label classification, the simplest method for computing an aggregate score across categories is to average the scores of all binary task. The resulted scores are called **macro-averaged** recall, precision, F1, etc. Another way of averaging is to sum over TP, FP, TN, FN and N over all the categories first, and then compute each of the above metrics. The resulted scores are called **micro-averaged**. Macro-averaging gives an equal weight to each category, and

is often dominated by the system's performance on rare categories (the majority) in a power-law like distribution. Micro-averaging gives an equal weight to each document, and is often dominated by the system's performance on most common categories. The two ways of measuring performance are complementary to each other, and both are informative.

A free-text document is typically represented as a feature vector x=(x(1),…,x(p)) , where feature values x(i) typically encode the presence of words, word n-grams, syntactically or semantically tagged phrases, Named Entities (e.g., people or organization names), etc. in the document. A standard method for computing the feature values x(i) for a particular document d is called the bag of words approach as discussed before.

It is useful to differentiate text classification problems by the number of classes a document can belong to. If there are exactly two classes (e.g. spam / non-spam), this is called a 'binary' text classification problem. In this model we have instead of documents sentences which have two classes 'summarize' or 'not summarize' depending on whether that sentence should be part of the summary or not. If there are more than two classes (e.g. positive / neutral / negative) and each document falls into exactly one class, this is a 'multi-class' problem. In many cases, however, a document may have more than one associated category in a classification scheme, e.g., a journal article could belong to computational biology, machine learning and some sub-domains in both categories. This type of text classification task is called a 'multi-label' categorization problem. Multi-label and multi-class tasks are often handled by reducing them to k binary classification tasks, one for each category. For each such binary classification tasks, members of the respective category are designated as positive examples, while all others are designated as negative examples. We will therefore focus more on binary classification.

The summaries generated using the proposed model were compared to the ones generated by the standard summary generators like Edmundson, Lex_Rank, Text_Rank and SumBasic.

5.8 Experimental Results

The comparison between the summaries generated by the proposed model and its comparison with the standard summarizers in terms of the classification metrics of recall, precision and F-score is given below. All the pre-processing was initially done on the documents followed by the feature extraction, LSA, concept generation and finally applying the Naïve Bayes classification

To make the comparisons, the ROUGE kit has been used which has inbuilt functions related to the different summaries as mentioned before

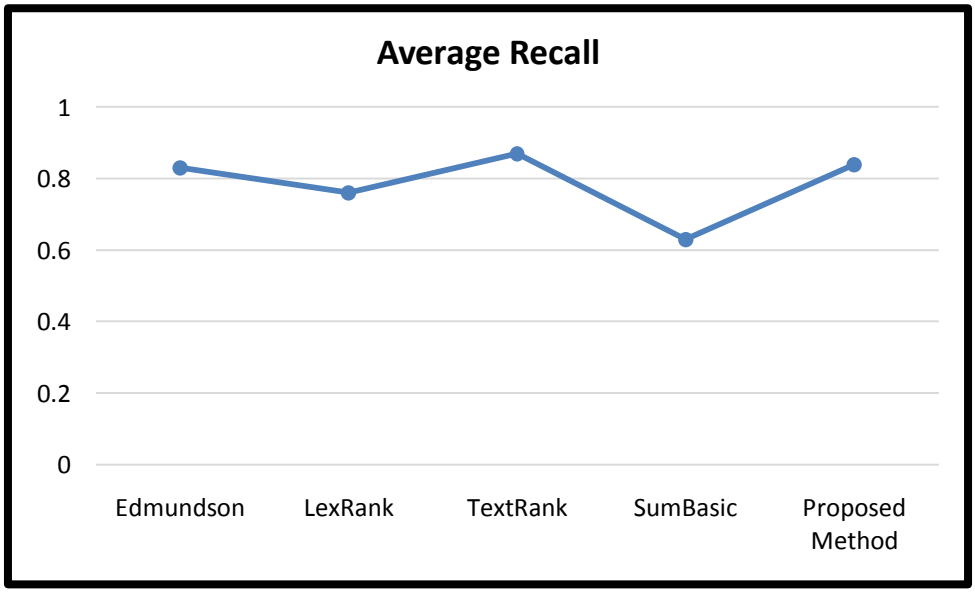

Figure 5-6 Comparison of chart for average recall of different summarization techniques

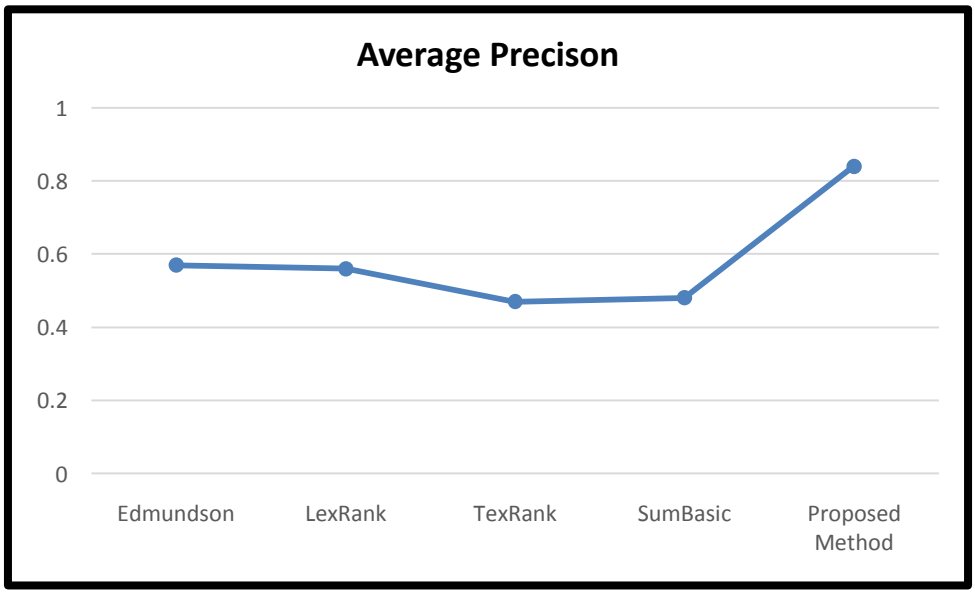

Figure 5-7 Comparison of chart for average precision of different summarization techniques

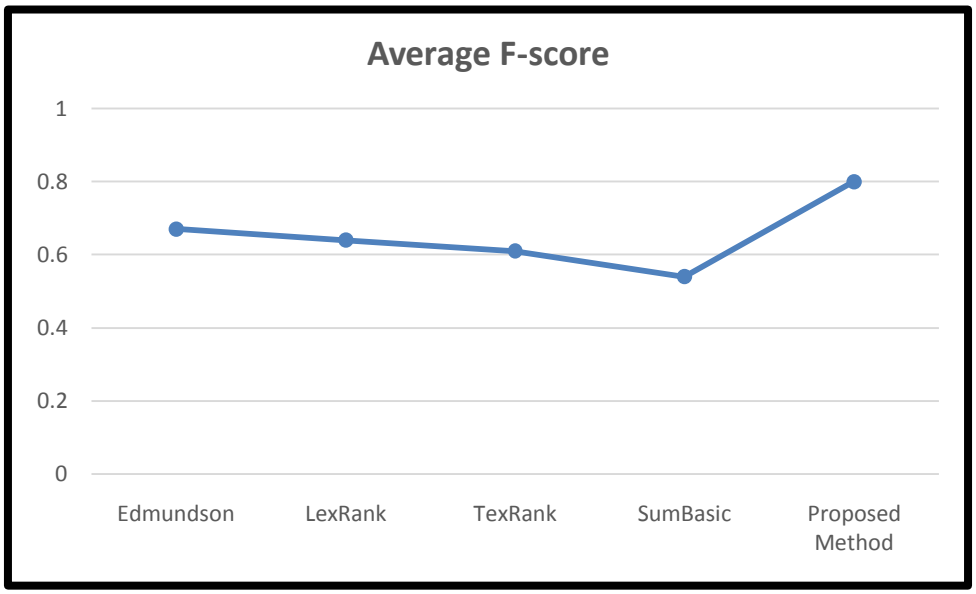

Figure 5-8Comparison of chart of average F-score of different summarization techniques

As per result obtained from various summarization tools. Figure 5.6 shows TextRank algorithms perform excellent compared to all other summarizers for average recall, and the proposed method also performs equally well for recall. Another obervation shows that SumBasic shows worst performance in comparison to rest in all three metrics.

LexRank and Edmundson both have almost equal performance for average recall. In brief, our proposed model and TextRank have almost the same result in an average recall score.

As per Figure 5.7, which is the average precision for all summarization tools of all documents. Our proposed model has shown remarkable performance in comparison of all other summarization techniques. The TextRank and SumBasic both shown weak performance for average precision in comparison to the proposed model. However, except the proposed model, Edmundson, LexRank, TextRank and SumBasic have almost same score of average precision for all documents.

The F-score results which is supposed to be the best metric for deciding the performance of a classifier, as shown in Figure 5.8 shows the highest value for the proposed model and demonstrates a value of 0.8 which is extremely good.

The rest of the summarizers have values between 4.7 and 6.3. The SumBasic has the lowest F-score as compared to other models. Edmundson, LexRank, and TexRank have almost same scores for average F-score.

Table 5-2 Proposed model's evaluation for sample documents

Document	Precision	Recall	F-Score
1	0.83	0.84	0.8
2	0.69	0.83	0.76
3	0.94	0.92	0.92
4	0.95	0.91	0.86
5	0.87	0.8	0.8
6	0.69	0.83	0.76
7	0.8	0.67	0.62
8	0.94	0.9	0.88
9	0.85	0.89	0.82
10	0.88	0.86	0.79

Table 5.2 shows the scores of sample documents for recall, precision and F-score. As per the table, almost all documents have achieved above 80% score except document number 2, 7 and 10.

Table 5-3 Average of recall, precision and F-score of all methods

Methods	Recall	Precison	F-score
Edmundson	0.83	0.57	0.67
LexRank	0.76	0.56	0.64
TexRank	0.87	0.47	0.61
SumBasic	0.63	0.48	0.54
Proposed Method	0.84	0.84	0.8

Table 5.3 shows averages recall, precision and recall for the sample documents. It can be observed that compared to other techniques, our proposed model has shown better result.

5.8.1 Visualization of Training and Testing Data

The proposed model was trained with Naïve Bayes classifier by using recursive feature elimination after which the dataset was divide into the training set and test set using the holdout method. In the following figures, sample training and testing datasets are shown in the form of a diagram. The data in this case are the sentences which are getting selected as part of the summary.

In all the figures given below, 1 (yellow dots) denotes that sentence is part of the summary, whereas 0 (blue dots) indicates that it is not part of the summary. As we can clearly see that few sentences fall into the category of summarized portion, even if they actually are not part of the summary. They indicate the False Positives / False Negatives.

Training & Test Visualization of Document – 1

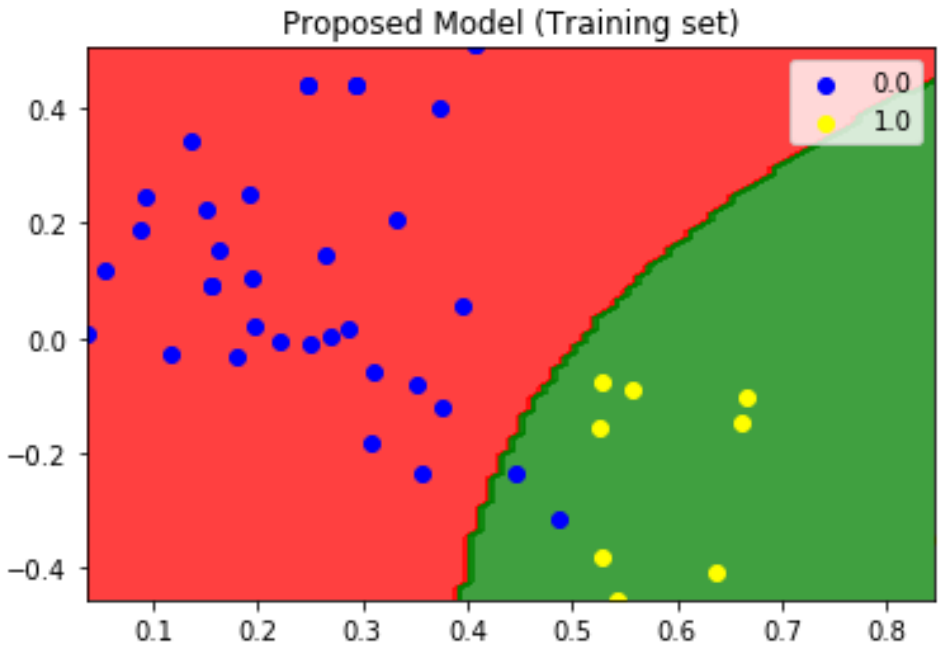

Figure 5-9 Training sets document visualization for Document 1

Once the classifier is trained, in the next step, we it is evaluated on the test set. Here also, 1 denotes that sentence is part of the summary and 0 indicates it is not part of the summary.

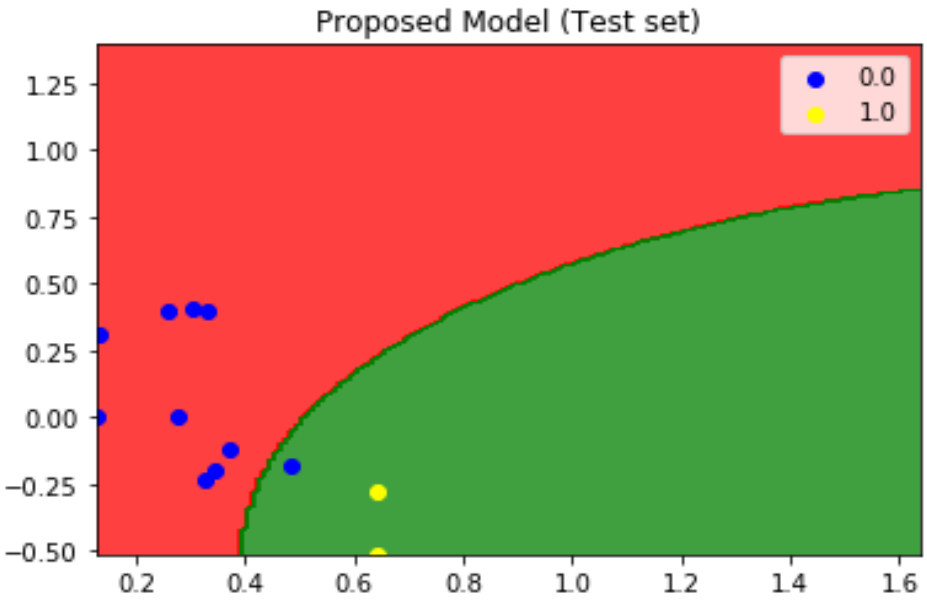

Figure 5-10 Test sets document visualization for Document 1

Training & Test Visualization of Document – 2

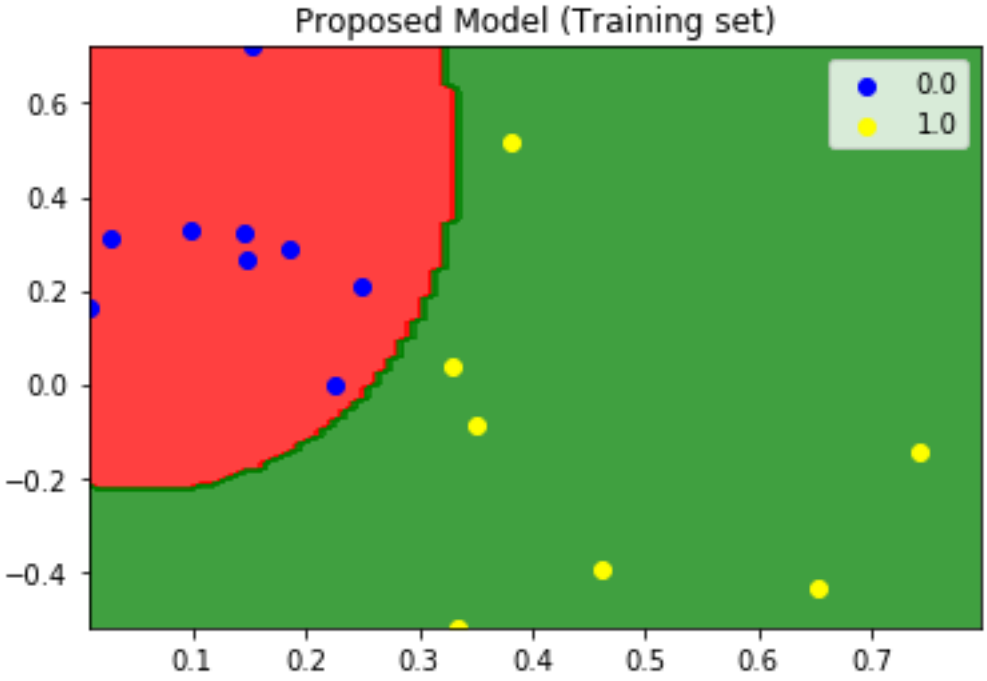

Figure 5-11 Training sets document visualization for Document 2

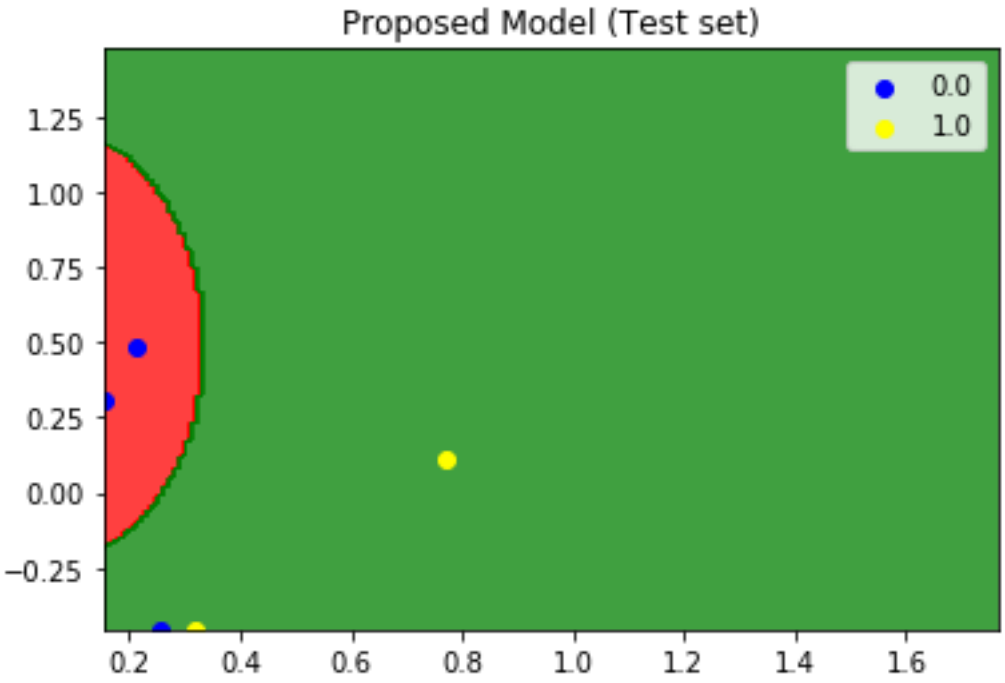

Figure 5-12 Test sets document visualization for Document 2

Training & Test Visualization of Document – 3

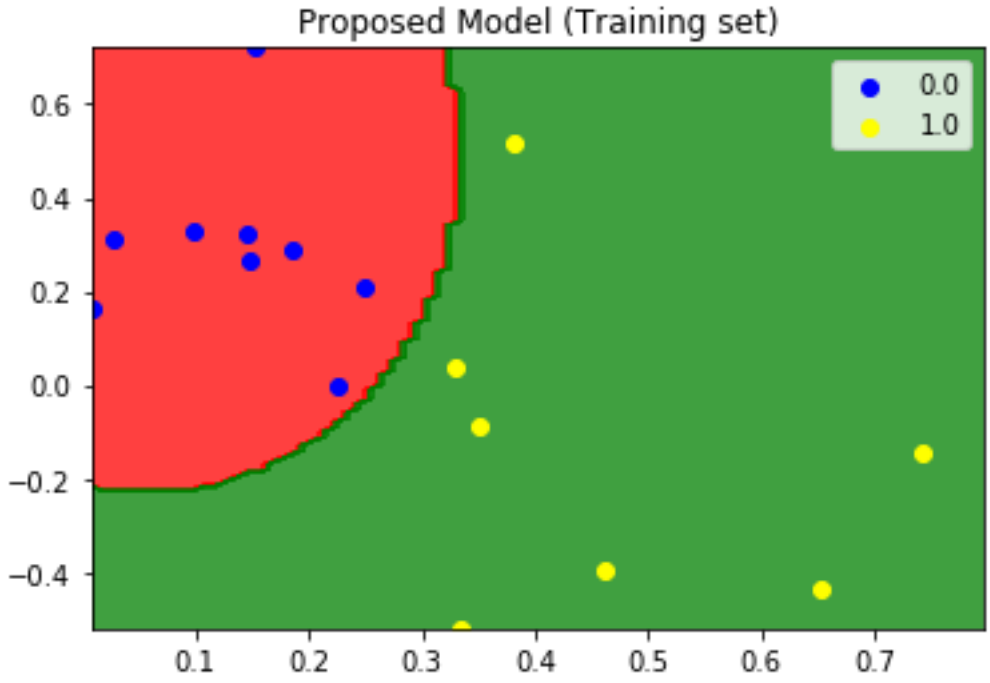

Figure 5-13 Training sets document visualization for Document 3

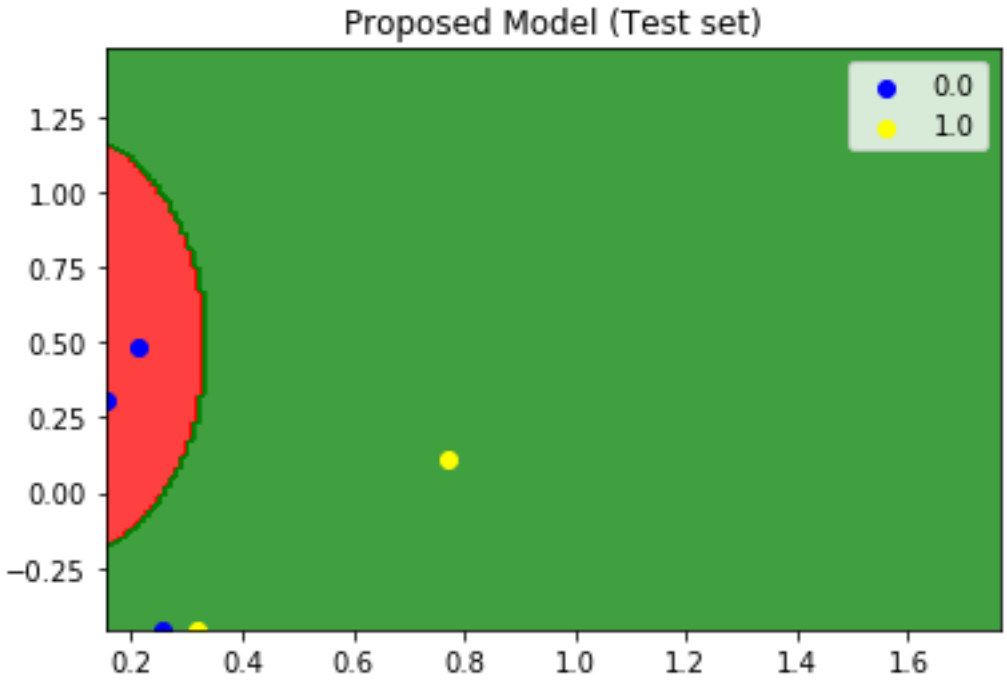

Figure 5-14 Test sets document visualization for Document 3

Summary

As discussed in the above section, it can be clearly seen that the proposed model generates a better summary as compared to the existing ones in terms of Accuracy, Precision and the harmonic mean – F-score. The datasets considered for the evaluation were the standard DUC 2004 datasets which had about 50 clusters (labels), each having at least 10 documents. The model has been evaluated on the DUC 2006 and the DUC 2007 also.