# Improved Algorithms for Document Classification & Query-based Multi-Document Summarization

Suzanne D'Silva, Neha Joshi, Sudha Rao, Sangeetha Venkatraman, and Seema Shrawne

**Abstract—With an excess of information in recent times, sound information retrieval is the need of the hour. Document Classification, where a document is classified as being under one of a number of predefined categories, is the foundation of an efficient and effective Information Retrieval system. Once information has been retrieved, the next step is unearthing the relevant and essential information. Query-based Multi-Document Summarization will do just that. In this paper, we analyze the different variants of the *k Nearest Neighbors* (kNN) Classification Algorithm and from them design the CAST Algorithm for Classification, which, as precision and recall results will show, performs better in most cases. For document summarization, we analyze and improvise on a *Hypergraph based* algorithm. Further, we design and describe the CAST Algorithm for Summarization and show that it performs well for Query-based Multi-Document Summarization.**

***Index Terms*—Document Classification, Multi-Document Summarization, Query-based summary.**

## I. Introduction

The 20th century and beyond, with the advent of the personal computer and the internet, has rightly been called the *Information Age*. The amount of information available is vast and is only growing exponentially. It has now become critical to search through this huge amount of information with speed and accuracy and finally find the required information. *Document Classification* is categorizing a document as being under one of a number of predefined classes. Categorizing helps, because it is now easier to find information and this has been put to a number of uses like spam filtering, topic filtering, language guessing, organizing documents, etc.

There are broadly two approaches to classification- the *supervised approach* where predefined category labels are given to documents based on the likelihood suggested by a training set of labeled documents [1] and the *unsupervised approach* where no training set or human intervention is required at any point of the process. The unsupervised approach includes *k-means clustering* and other *clustering algorithms* which simply try to organize data into clusters such that similarity within a cluster is maximized and similarity between clusters is minimized. Such an approach is well suited for systems like the World Wide Web where classes are dynamic and cannot be defined beforehand. In other scenarios the supervised approach has been found to be adequate.

Among the supervised approaches are Support Vector Machines (SVM), k Nearest Neighbors (kNN), Naive Bayes Classifier, Neural Networks (NN) and Decision Trees. Among these, kNN is one of the most popular and extensive [2], but it still has many defects, such as great calculation complexity and bias towards classes which have more training documents. In this paper, we mention methods to reduce the bias of kNN and design an algorithm using these methods.

While searching for information on a particular topic, it is often the case that we cannot decide whether an article is relevant or not without having read it entirely. Instead of having to read the entire article, if we could simply specify our search area of interest and have a system summarize the contents of the document that matched that interest we could then conclude whether the article was relevant. Thus, much time and effort could be saved. *Document Summarization* is converting a large text into a brief explanation that conveys all the important information present in the text, but omits trivial or redundant data.

Summarization approaches can be divided into two types – *Extractive summarization* and *Abstractive summarization*. Extractive summarization deals with identifying the most relevant sentences or passages in one or more documents and combining them together to form a non-redundant summary that is shorter than the original set of documents with as little information loss as possible. Abstractive summarization, on the other hand, involves parsing the original text in a deep linguistic way, interpreting it semantically and converting it into a formal representation, finding new more *concise* concepts to describe the text and then generating a new shorter text- an abstract, with the same basic informative content. While there has been quite a bit of work in using extractive summarization, there has been limited study in abstractive summarization as this is much harder to achieve. The algorithm presented in this paper makes use of the extractive summarization technique.

*Query-based Text Summarization* goes a step further as it summarizes data of a document based on a query entered by a user. *Query-based Multi-Document Text Summarization* takes a query, summarizes the content of a number of documents with respect to the query and returns a single consolidated summary. While there are now a number of tools that provide summaries of documents, work on

Manuscript received April 22, 2011 and revised June 28, 2011.

Suzanne D'Silva is with the Veermata Jijabai Technological Institute, Matunga, Mumbai – 400019, India (email: suzanne.d1411@gmail.com).

Neha Joshi is with the Veermata Jijabai Technological Institute, Matunga, Mumbai – 400019, India (email: nehaj_26@yahoo.co.in).

Sudha Rao is with the Veermata Jijabai Technological Institute, Matunga, Mumbai – 400019, India (email: raosudha89@gmail.com).

Sangeetha Venkatraman is with the Veermata Jijabai Technological Institute, Matunga, Mumbai, India (email: sangeetha2089@gmail.com).

Seema Shrawne is with the Veermata Jijabai Technological Institute, Matunga, Mumbai – 400019, India (email: scshravane@vjti.org.in).

query-based text summarization is still in progress, and it is this type of summary that we attempt to produce through the methods that we will describe in this paper.

## II. Related Work

### A. Classification: Problem Description

Zhou [3] described the process of text classification as follows: given a document collection $\{D_1, D_2, \ldots, D_n\}$, a category $C_i$ from the predefined category collection $\{C_1, C_2, \ldots, C_m\}$ has to be assigned to a test document $D_j$. Thus we aim to estimate a mapping function f: D x C -> {0, 1} (that describes how documents ought to be classified). The mapping function between collection D and collection C is : D x C $\{d_jc_i \mid j = 1,2,.. ,n , i = 1,2, ,m\}$ and the value of element $d_jc_i$ is 1 means that we classify document $d_j$ under category $c_i$, otherwise, the value of element $d_jc_i$ is 0 means we do not classify document $d_i$ under category $c_i$ .

### B. Preprocessing for Text Classification

For every document, *Stopword filtering* (where frequently occurring words of little importance are filtered out from the document) and *Stemming* (where different morphological variants of a word are reduced to a common root called a *stem*} are applied on each of the training data text files. The final set of words forms the vocabulary of the training set.

The documents are represented in the *Vector Space Model*. The core idea is to make the document become a numeral vector in multi-dimension space. Every dimension represents a word and the corresponding numerical value represents the *term frequency*, *term frequency- inverse sentence frequency* (tf-isf) or *term frequency-inverse document frequency* (tf-idf).

### C. k Nearest Neighbors(kNN) Algorithm

kNN is a supervised learning classification algorithm where the classification is performed using the samples in the training set without any additional data. kNN classification algorithm predicts the test sample's category based on the k training samples which are found to be the nearest neighbors to the test sample, assigns scores to the different categories using a decision rule and finally classifies it under the category which has the largest score.

The kNN algorithm for classifying a document X is:

Suppose there are a number of categories $\{C_1, C_2, ..C_m\}$ and the total number of training documents is n. After pre-processing each document, they all become j-dimension feature vectors.

1) Represent document *X* in the same text feature vector form $(X_1, X_2, \ldots, X_j)$ as the training samples.
2) Calculate the similarities between all training samples and document *X* using *Cosine Similarity*, refer to (1). Consider a training document $D_l$ $\{d_{l1}, d_{l2}, .., d_{lj}\}$.

$$\mathrm{Sim}(X, D_l) = \frac{\sum_{h=1}^{j} X_h \times d_{lh}}{\sqrt{\sum_{h=1}^{j} X_h^2} \times \sqrt{\sum_{h=1}^{j} d_{lh}^2}} \qquad (1)$$

3) Let knn be the k documents which have the highest similarity with X. These documents form the set of k nearest neighbours.
4) Use a decision rule to decide the scores of each class of X using knn.
5) Select the class with the highest score to be the category of X.

A number of decision rules exist to give scores to classes.

1) Decision rule of Classical kNN

$$Score(X, Ci) = \sum_{Dj \in knn} Sim(X, Dj) * y(Dj, Ci) \qquad (2)$$

where $y(D_j, C_i) = 1$ if document $D_j$ belongs to category $C_i$; = 0 otherwise.

2) Decision rule of Adaptive kNN

$$Score(X, Ci) = \frac{\sum_{Dj \in top\_nm\_knn} Sim(X, Dj) * y(Dj, Ci)}{\sum_{Dj \in top\_nm\_knn(Ci)} Sim(X, Dj)} \qquad (3)$$

top-nm-knn($C_i$) = {top nm documents of knn |

$$\min\left(\alpha + \left\lceil k * \frac{N(Ci)}{\max[N(Cg)]} \right\rceil, k, N(Ci)\right)\}$$

where $N(C_i)$ is the size of class $C_i$, $\max[N(C_g)]$ is the maximum size of any class, & α is a non-negative integer [4].

3) Decision Rule of Neighbour Weighted kNN

$$Score(X, Ci) = \sum_{Dj \in knn} wt(Ci) * Sim(X, Dj) * y(Dj, Ci) \qquad (4)$$

Here let the k neighbours among the training documents be contained in k* categories: $C_1, C_2, \ldots, C_{k*}$. Then weight of a class $C_i$ is given as [5],

$$\mathrm{wt}(C_i) = \frac{1}{\left(\frac{\mathrm{Num}(C_i^d)}{\min_{1 \le l \le k^*} \mathrm{Num}(C_l^d)}\right)^{1/\mathrm{exponent}}}$$

4) Decision Rule of Yang's Variant[6]

$$\mathrm{Score}(X, C_i) = \frac{\sum_{D_j \in knn} \mathrm{Sim}(X, D_j) \times \mathrm{y}(D_j, C_i)}{\sum_{D_j \in knn} \mathrm{y}(D_j, C_i)} \qquad (5)$$

### D. Improved knn using top k-buffer

This algorithm [7] reduces the number of *cosine similarity* operations to be carried out using a buffer of size k, called the *top-k buffer*. All term frequencies are normalized and cosine similarity value between two documents is only calculated between terms that are common to both documents (*keywords*). Using the top-K buffer, a set of the k closest documents at a certain point of time is always maintained.

While considering a new document, if its sum of term frequencies is less than the value of minimum cosine similarity in the buffer, we need not compute its cosine similarity. This prevents a lot of unnecessary cosine similarity computations.

In this paper, we make use of the results of the variants of kNN given above to design a new algorithm that performs document classification.

### *E. Approaches to Document Summarization*

In the past, many approaches for query based extractive summarization have been adopted. These approaches dealt with different important aspects to be considered for the purpose of summarization.

1) *Hypergraph* based semi-supervised sentence ranking for query oriented summarization [8]: In this algorithm, the document is represented as a *hypergraph*, in which the sentences (along with the query) are represented as nodes and the similarity between them is represented by the edge weights between the nodes. Also the sentences are divided into clusters using the DBSCAN algorithm and a hyperedge is added for each cluster. Then the Hypersum algorithm is run to calculate the significance of the sentences and the ones with the highest significance and which are non-redundant are included in the final summary.
2) Sentence Feature Fusion [9]: This paper considered the following different features for ranking the sentences of the document: Term Frequency-Inverse Sentence Frequency, keyword feature, part of speech of the term, and length of the sentence containing the term.
3) Sentence clustering and extraction [10]: This paper adopted the clustering technique to determine the topic sentences from the given document. It also presented a strategy to determine the optimal value of k for the k clustering algorithm.

In this paper we present two different algorithms for document summarization. The first one tries to improvise on the *Hypergraph* technique described in [8] by using k clustering algorithm to form the clusters where the initial k centroids are chosen according to their relevance with the query. Also, the similarity measure between two sentences not only depends on the frequency of terms appearing in them but also on the other important features mentioned in [9]. The second algorithm tries to combine the two important aspects presented in [9] and [10] for obtaining a query-based multi document summary.

## III. Proposed Method

This section outlines the algorithms used in our Classification & Summarization Tool, CAST.

### *A. CAST Algorithm for Classification*

Based on the results of the 4 previous variants of kNN, we designed the CAST Algorithm for Classification which uses the results of the 4 variants. Each variant outputs a list of classes in decreasing order of likelihood that the test document belongs to that class. A weighted rank system is used to score each class based on its position in the list of classes output by each of the 4 variants.

The CAST algorithm for Classification is as follows:

```
Input: D = {$D_1$, $D_2$,..,$D_n$} is the set of training documents in the training set (TS)
        C = {$C_1$, $C_2$,...,$C_m$}  is the list of categories (classes)
        Test Document X {$w_1$, $w_2$,... $w_h$}, where h is the length of the feature space
        k, α, exponent
Output: Category(X)
Begin
resultAdaptive ← AdaptiveKNN(k, α)
resultNW ← NeighbourWeightedKNN(k, exponent)
resultYang ← YangVariantKNN(k)
resultImprovedKNN ←ImprovedKNN(k)
for each result do
    if class Ci is given rank r then
        CastScore(Ci) += maximum no. of classes - r;
Category(X) ←class with maximum CastScore
End
```

### *B. Preprocessing for Summarization*

The given document undergoes following stages of pre-processing:

1) Conversion from PDF format to text format
2) Tokenization & Part of Speech Tagging using Stanford POS Tagger
3) Stopword filtering
4) Stemming using Porter stemmer
5) Calculating tf-isf of the terms
6) Assigning weights to the terms considering tf-isf, tag assigned to the terms & their similarity to the query keywords
7) Assigning weights to sentences by summing all the term weights

### *C. Improved HyperSum Algorithm for Summarization*

The basic algorithm is same as that described in [8]. But instead of the DBSCAN algorithm, the clustering is done using k clustering algorithm. Also in the k clustering algorithm, instead of choosing the initial k centroids randomly, we choose the first k sentences with highest weights as the initial k centroids. Since the weight of the terms and hence the weights of the sentences depend partly on the query relevance, we choose those sentences which are most relevant to the query as the centroids. Also, in the original implementation, for calculating the cosine similarity between sentences, the sentences were expressed as vectors where the elements of the vectors consisted of only the term frequencies. In our version of the HyperSum algorithm, the elements of the vector consist of the weights of the terms. Hence the similarity measure includes many factors and not just term frequency.

### *D. CAST Algorithm for Summarization*

Initially, the sentences are represented as sentence vectors, where the elements are the term weights of the words calculated by considering the query relevance, tf-isf and part of speech tagging. The sentence weights are calculated as the sum of all the term weights in the sentence. The sentences are then sorted in descending order of their sentence weights.

Now the sentences are clustered using k- cluster algorithm. The value of k is derived from a formula as specified in the algorithm. Instead of assigning k random sentences to the k clusters initially, we choose the k sentences with the highest sentence weights and assign them to the k clusters. Then the k clusters are formed using the k cluster algorithm.

For generating the summary, we choose the sentence from each cluster with the highest cosine similarity with the centroid. This process is repeated till the summary length is reached. In case the summary length is less than the number of clusters k, then the sentences are selected from those clusters having more relevance to the document.

For generating the multi document summary, the CAST algorithm is run on the consolidated summaries of the individual documents.

The CAST Algorithm for Summarization is as follows:

```
Input: Document Set D = {D1, D2, D3,.... Dn}
       Query = Q
       Summarization factor = sf
Output: Summary S
Begin
A. foreach document Di in set D

1. foreach term t in Di
    1.1 Calculate tf_isf of t
        1.1.1 tf = frequency(t)/ total # terms in Di
        1.1.2 isf= log( total # sentences/ # sentences containing term t)
        1.1.3 tf_isf(t) = tf * isf
    1.2 Calculate wt of t
        1.2.1 According to tf_isf , wt(t)= tf_isf(t)
        1.2.2 According to Q & synonyms of Q
            if(t is query word) then
                wt(t) += max(wt(t))*
                QUERY_WORD_MULTIPLIER
            else if(t is synonym of query word)
                wt(t) += max(wt(t)) *
                QUERY_WORD_SYN_MULTIPLIER
        1.2.3 According to tag of word
            if(t is proper noun) then wt(t) += max(wt(t)) *
                PROPER_NOUN_MULTIPLIER
            if(t is common noun) then wt(t) += max(wt(t)) *
                COMMON_NOUN_MULTIPLIER
            if(t is verb) then wt(t) +=max(wt(t)) *
                VERB_MULTIPLIER
            if(t is adjective) then wt(t) += max(wt(t)) *
                ADJ_MULTIPLIER

2. foreach sentence s in Di
    wt(s) = summation of weights of all terms (t) in s

3. Sort the sentences S of Di in descending order according to wt(s)

4. Calculate the no. of clusters (K)
    $k = n\frac{|D|}{\sum_{i=1}^{n}|S_i|} = n\frac{\left|\bigcup_{i=1}^{n} S_i\right|}{\sum_{i=1}^{n}|S_i|}$

where |Di| is the number of terms in document Di, |Si| is the number of terms in the sentence Si, n is the number of sentences in document Di.
5. Assign the first K sentences from the sorted list obtained by step 4 as the central sentences of the clusters.
6. Form the K clusters using K-Clustering algorithm
    6.1 Assign each sentence s from the set of sentences S to the cluster that has the closest central sentence using the cosine similarity between the sentence s and the central sentence of the cluster c
    6.2 When all sentences have been assigned to a cluster, recalculate the central sentence of each cluster. The central sentence is the one with the highest accumulative similarity
    6.3 Repeat steps 6.1 and 6.2 until central sentence no longer moves. This produces a separation of the sentences into K clusters
7. Calculate the cosine similarity between each cluster c and the document Di
8. Sort the clusters in descending order of their similarity with document. Let this sorted list be C.
9. No. of sentences of Summary = Total No. of sentences* Summarization factor
10. Generate the summary of document Di
    10.1 Summarized Text ST = { }
    10.2 while size of ST < # sentences of Summary
        foreach cluster c in C
            Choose the sentence s from c such that s is not
            in ST & has the highest similarity with the
            centroid of c; ST = ST U s

B. Let DS be the document created by merging all the summaries of set D

C. Run the algorithm in A on the document DS and obtain the resultant summary S
```

## IV. Experimental Evaluation And Results

### A. *Classification Results*

The Training Data text files are all converted into the Attribute Relation File Format (.arff). Even the test document is converted into the .arff format. Within each file, every instance is represented as a list of terms and their frequency within the file. We normalize each term frequency by dividing it with the total number of words in the document. This method has been found to remove the bias towards longer documents.

Similarity of documents is measured using the Cosine Similarity function, refer to (1). We tested our algorithm with the standard dataset *Reuters-21578 Mod Apte* split, out of which we considered 4431 training documents categorized under 25 classes with strengths ranging from 25 to 400. The value of k is set to be the square root of the training set size. Therefore, in our evaluations, k was set to 65. The values of the algorithm parameters were set as α = 3 and exponent = 1.5.

Efficiency of the classification algorithms was measured by precision and recall. *Precision* for a class C is the number of correctly identified documents of C divided by the total number of documents classified under class C. *Recall* is the proportion of documents of class C that get correctly classified under C. It is desirable to have high values of both these parameters. Tables I and II list the precision and recall values of different algorithms for different classes (C: Classical kNN Algorithm, A: Adaptive kNN Algorithm, N: Neighbour Weighted kNN Algorithm, Y: Yang's Variant kNN Algorithm, CAST: CAST kNN Algorithm).

TABLE I: CLASSIFICATION PRECISION VALUES

| **Class** | **Size** | **C** | **A** | **N** | **Y** | **I** | **CAST** |
|---|---|---|---|---|---|---|---|
| *Acq* | 400 | .69 | .71 | .70 | .35 | .63 | **.71** |
| *Carcass* | 50 | .22 | .33 | .28 | .11 | .17 | **.33** |
| *Copper* | 47 | .50 | .72 | .56 | .67 | .11 | .61 |
| *Cpi* | 69 | .57 | .54 | .57 | .11 | .50 | **.57** |
| *Crude* | 389 | .87 | .79 | .83 | .46 | .77 | **.81** |
| *Ipi* | 41 | .50 | .58 | .50 | .75 | .42 | .58 |
| *Iron-steel* | 40 | .21 | .57 | .43 | .64 | .07 | .50 |
| *Jobs* | 46 | .24 | .33 | .33 | .38 | .29 | **.38** |
| *Livestock* | 75 | .17 | .33 | .25 | .33 | .17 | .25 |
| *Money-fx* | 400 | .56 | .51 | .61 | .28 | .73 | .60 |
| *M-supply* | 138 | .62 | .76 | .65 | .62 | .71 | **.79** |
| *Ship* | 197 | .43 | .63 | .61 | .56 | .37 | **.63** |
| *Trade* | 369 | .83 | .80 | .83 | .41 | .88 | **.85** |
| *Wheat* | 212 | .30 | .24 | .28 | .32 | .65 | .59 |

TABLE II: CLASSIFICATION RECALL VALUES

| **Class** | **Size** | **C** | **A** | **N** | **Y** | **I** | **CAST** |
|---|---|---|---|---|---|---|---|
| *Acq* | 400 | .97 | .96 | .97 | .89 | .96 | **.97** |
| *Carcass* | 50 | .44 | .35 | .50 | .08 | .43 | .40 |
| *Copper* | 47 | 1.0 | .50 | 1.0 | .23 | 1.0 | **1.0** |
| *Cpi* | 69 | .47 | .43 | .47 | .06 | .47 | **.48** |
| *Crude* | 389 | .66 | .66 | .7 | .39 | .72 | **.71** |
| *Ipi* | 41 | .67 | .58 | .67 | .31 | .50 | .58 |
| *Iron-steel* | 40 | .43 | .53 | .60 | .20 | .50 | **.64** |
| *Jobs* | 46 | 1.0 | .7 | 1.0 | .22 | 1.0 | **1.0** |
| *Livestock* | 75 | 1.0 | .53 | 1.0 | .24 | .80 | **1.0** |
| *Money-fx* | 400 | .58 | .63 | .59 | .27 | .53 | .58 |
| *M-supply* | 138 | .42 | .41 | .51 | .28 | .39 | .44 |
| *Ship* | 197 | .69 | .58 | .64 | .26 | .70 | .62 |
| *Trade* | 369 | .54 | .64 | .56 | .41 | .47 | **.60** |
| *Wheat* | 212 | .47 | .45 | .48 | .29 | .38 | .43 |

Thus, CAST algorithm is found to perform as well as, and in some cases, better than the other variants of kNN algorithm.

## B. Summarization Results

According to our experiments, the following values of the constants defined in the CAST algorithm gave satisfactory results:

*QUERY_WORD_MULTIPLIER* = 10

*QUERY_WORD_SYN_MULTIPLIER* = 20/3

*PROPER_NOUN_MULTIPLIER* = 1
*COMMON_NOUN_MULTIPLIER* = 0.9

*VERB_MULTIPLIER*= 0.7

*ADJECTIVE_MULTIPLIER* = 0.4

The summaries obtained using the two algorithms presented in the paper were compared with human summaries. The performances of the two algorithms were evaluated using ROUGE [11] and the results are as follows:

TABLE III: SUMMARIZATION PRECISION & RECALL VALUES

| **Algorithms** | ***HyperSum*** | | ***CAST*** | |
|---|---|---|---|---|
| Methods | Precision | Recall | Precision | Recall |
| *ROUGE-N* | **0.71** | **0.77** | **0.68** | **0.61** |
| *ROUGE-L* | **0.71** | **0.76** | **0.66** | **0.60** |
| *ROUGE-W* | 0.38 | 0.21 | 0.35 | 0.17 |
| *ROUGE-S* | **0.51** | **0.59** | 0.46 | 0.39 |

The results of evaluation (see Table III) using ROUGE show that the CAST algorithm performs as well as HyperSum. However, CAST deals only with clustering and uses sentence features whereas the HyperSum algorithm requires two processing-intensive operations -clustering and complex matrix calculations. Therefore, in applications where a summary is to be generated while being economical with resources, CAST algorithm would be a better option.

## V. CONCLUSION

In this paper, we set out to develop an improved version of the kNN Algorithm. After analyzing the strengths of the existing kNN algorithms, we arrived at the CAST algorithm of Classification which uses the weighted result of Adaptive kNN, Neighbor Weighted kNN, Yang's Variant of kNN and Improved kNN using a top-k buffer to output the class of a test document. After comparing results of Classical kNN, the four variants, and CAST, we are confident that CAST performs well even when classes in the training set are of different strengths. A comparison of CAST with other supervised classification approaches like SVM would be the next step.

Also, in this paper we proposed to improvise on HyperSum and adapt it for Multi-document summarization by using k clustering algorithm to cluster the sentences instead of the DBSCAN algorithm and considering the query relevance while assigning the initial k centroids of the clusters. We use the same improvisation for our CAST algorithm. In addition to this, in CAST algorithm, in case the summary length is less than the number of clusters k, then the sentences are selected from those clusters having more relevance with respect to the document. The evaluation results of Query Based Multi-Document Summarization suggest that this is a promising area for research to be continued.

## REFERENCES

[1] Y. Ko, and J. Seo, “Automatic text categorization by unsupervised learning,” in *Proc. 18th conference on Computational linguistics,* vol. 1, 2000, pp. 453-459.

[2] Y. Yang, “An evaluation of statistical approaches to text categorization,” *Inf. Retr.,* vol. 1, pp. 69-90, 1999.

[3] Y. Zhou, Y. Li and S. Xia, “An improved KNN text classification algorithm based on clustering,” *Journal of Computers,* vol. 4, pp.230-237, 2009.

[4] L. Baoli, L. Qin and Y. Shiwen, “An adaptive k-nearest neighbor text categorization strategy,” *ACM,* pp. 215-226, 2004.

[5] S. Tan, “Neighbor-weighted K-nearest neighbor for unbalanced text corpus,” *Expert Syst. Appl,* pp. 667-671, 2005

[6] Y. Yang, T. Ault, T. Pierce and C. W. Lattimer, “Improving text categorization methods for event tracking,” in *Proc. 23rd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2000,* pp. 65-72.

[7] M. Miah, “Improved k-NN algorithm for text classification,” in *Proc. Int. Conf. on Data Mining*, 2009, pp. 434-440.

[8] W. Wang, F. Wei, W. Li, and S. Li, “HyperSum: hypergraph based semi-supervised sentence ranking for query-oriented summarization,” in *Proc. 18th ACM conference on Information and knowledge management,* 2009, pp. 1855-1858.

[9] L. Suanmali, M. S. Binwahlan and N. Salim, “Sentence features fusion for text summarization using fuzzy logic,” *IEEE Computer Society, Hybrid Intelligent Systems, International Conference,* vol. 1, pp. 142-146, 2009.

[10] P. Zhang and C. Li, “Automatic text summarization based on sentences clustering and extraction,” *IEEE Computer Society*, *International Conference on Computer Science and Information Technology,* vol. 0, pp. 167-168, 2009.

[11] C. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Proc. Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL,* 2004.

**Suzanne D'Silva** will be graduating with a Bachelor's Degree in Engineering in computer science from Veermata Jijabai Technological Institute, Mumbai (India) in 2011.

She has interned as a Software Analyst at Morgan Stanley Advantage Services during the period May-July 2010. Her areas of interests are Algorithms, Artificial Intelligence, Data Structures, Data Mining and Biotechnology.

**Neha Joshi** holds a Diploma in Computer Engineering from Shri Bhagubhai Mafatlal Polytechnic (2008) and will be graduating with a Bachelor's Degree in Engineering in computer science from Veermata Jijabai Technological Institute, Mumbai (India) in 2011.

She has interned as a Software Developer at Nomura Services India Pvt. Ltd. during the period May-July 2010. Her areas of interests are Databases, Data Mining, Algorithms and Computer Networks.

**Sudha Rao** will be graduating with a Bachelor's Degree in Engineering in computer science from Veermata Jijabai Technological Institute, Mumbai (India) in 2011.

She has interned as a Software Developer at Microsoft IDC, Hyderabad during the period May-July 2010. Her areas of interests are Algorithms, Data structures, Operating Systems, Artificial Intelligence and Web Application Development.

**Sangeetha Venkatraman** will be graduating with a Bachelor's Degree in Engineering in computer science from Veermata Jijabai Technological Institute, Mumbai in 2011.

She has interned as a Program Analyst at Nomura Services Pvt. Ltd. during the period May-July 2010. Her areas of interest are Databases, Computer Networks, Operating Systems, Data Structures, Artificial Intelligence and Web Application Development.

**Mrs. Seema Shrawne** graduated with a Bachelor’s Degree in Engineering in computer science from Government College of Engineering, Amravati, in 1992.

She has been a lecturer in the computer technology department at Veermata Jijabai Technological Institute, Mumbai (India) for the last 10 years. Her areas of interest are Databases, Computer Networks and Information Retrieval.