Abstract—Document summarization is the way of reducing size of text document and that preserving the most important content of the original document into the reduced document (Summary). In current scenario there are lots of work has been done in document summarization. There are various techniques available for document summarization but most of the techniques used similarity of sentences to extract sentence, in the document summarization a context of the document are important, so our current method used term indexing model to gives index to document as well as sentences in that document. In this proposed system we used context based document indexing based on vector space model. This document indexing model works with document frequency (DF) and term frequency (TF). DF and TF model gives document indexing weight which is used for document summarization. I compare my system with traditional term based indexing model and will prove that our system gives better result than this system.

Keywords— Document Context, Vector Space Mode, Term frequency, Document Frequency.

I. INTRODUCTION

Text summarization is the process of automatically creating a compressed version of a given document preserving its information content. Automatic document summarization is an important research area in natural language processing (NLP). The technology of automatic document summarization is developing and may provide a solution to the information overload problem. Modern text retrieval systems principally rely on orthographic, semantic, and statistical analysis. The usual approach is to use white space to identify word boundaries, followed by stemming to conflate words with similar surface forms into a common term. A weight is then computed for each term in every document using the frequency of the term in the document, the selectivity of the term, and the length of the document. In vector space text retrieval, queries are represented in a manner similar to the documents, and the similarity of each document in the collection to the query is then computed as the normalized inner product of the document and query term weight vectors.

In probabilistic text retrieval, a term weight is treated as the probability of relevance of a document to a query, conditioned on the presence of that term in the query. Probabilistic and vector space techniques are often combined with Boolean text retrieval, in which the presence or absence of a term or combination of terms can be explicitly required in the query specification. The principal advantage of vector space and probabilistic text retrieval over a purely Boolean approach is that lists of documents that are ranked in order of decreasing probability of relevance allow users to interactively decide how many documents are worth examining. Unranked Boolean techniques, on the other hand, might be preferred when no user interaction is possible before the next processing stage. In either case, when the document collection is relatively stable it is structure on the feature set that can be searched in sublinear time. The utility of a text retrieval system depends strongly on how well the query is constructed, and that depends in turn on how well the user understands the collection and the way in which the indexed features can be used to select documents. It is usually fairly straightforward to find some relevant documents, but interactive inspection by the user is generally needed if the relevant documents must be more carefully separated from the irrelevant ones. An iterative query reformulation process such as Simulated Nucleation can be used to speed this process, leveraging inspection of a few documents to produce a query that better separates relevant and irrelevant documents.

II. LITERATURE SURVY

Text summarization can either be “abstractive” or “extractive.” The abstraction-based models mostly provide the summary by sentence compression and reformulation allowing summarizers to increase the overall information without increasing the summary length. However, these models require complex linguistic processing. Sentence extraction models, on the other hand, use various statistical features from the text to identify the most central sentences in a document/set of documents. Erkan and Radev proposed LexRank to compute sentence importance based on the concept of eigenvector centrality and degree centrality.
They used the hypothesis that the sentences that are similar to many of the other sentences in a cluster are more salient to the document topic. Sentence similarity measures based on cosine similarity was exploited for computing the adjacency matrix. Once the document graph is constructed using the similarity values, the “degree centrality” of a sentence si are defined as the number of sentences similar to si, with similarity value above a threshold. Eigenvector centrality is computed using the LexRank algorithm iteratively, which was an adaptation of the PageRank algorithm. Mihalcea and Tarau proposed TextRank, another iterative graph-based ranking framework for text summarization and showed that other graph-based algorithms can be derived from this model. None of the models, as described in this section, address the problem of “context insensitive document indexing.” A propose system which uses the knowledge derived from the underlying corpus to give a context-sensitive indexing weight to the document terms. Sentence similarity will be calculated using the indexing weights thus obtained.

III. EXISTING SYSTEM

Existing methods for single document keyphrase extraction [3] usually make use of only the information contained in the specified document. This study proposes to construct an appropriate knowledge context for a specified document by leveraging a few neighbor documents close to the specified document. The neighborhood knowledge can be used in the keyphrase extraction process and help to extract salient keyphrases from the document. In particular, the graph-based ranking algorithm is employed for single document keyphrase extraction by making use of both the word relationships in the specified document and the word relationships in the neighbor documents, where the former relationships reflect the local information existing in the specified document and the latter relationships reflect the global information existing in the neighborhood. The framework for the system described in [3] is as follows:

i. Neighborhood Construction: Expand the specified document d0 to a small document set D= {d0, d1, d2…dk} by adding k neighbor documents. The neighbor documents d1, d2… dk can be obtained by using document similarity search techniques.

ii. Keyphrase Extraction: Given document d0 and the expanded document set D, perform the following steps to extract keyphrases for d0:

- **Neighborhood-level Word Evaluation:** Build a global affinity graph G based on all candidate words restricted by syntactic filters in all the documents of the expanded document set D, and employ the graph-based ranking algorithm to compute the global saliency score for each word.

- **Document-level Keyphrase Extraction:** For the specified document d0, evaluate the candidate phrases in the document based on the scores of the words contained in the phrases, and finally choose a few phrases with highest scores as the keyphrases of the document.

It is noteworthy that the proposed approach has higher computational complexity than the baseline approach because it involves more documents, and we can improve its efficiency by collaboratively conducting single document keyphrase extractions in a batch mode. But the focus on more test data was lacking compromising with the robustness of the system.

Xiaojun Wan [4], proposed a novel unified approach to simultaneous single-document and multi-document summarization by making using of the mutual influences between the two tasks. Experimental results on the benchmark DUC datasets show the effectiveness of the proposed approach. Given a document set, in which the whole document set and each single document in the set are required to be summarized, we use local saliency to indicate the importance of a sentence in a particular document, and use global saliency to indicate the importance of a sentence in the whole document set.

TextRank demonstrated [5] is a system for unsupervised extractive summarization that relies on the application of iterative graph based ranking algorithms to graphs encoding the cohesive structure of a text. The distinguishing characteristics of the proposed system is that it does not rely on any language-specific knowledge resources or any manually constructed training data, and thus it is highly portable to new languages or domains. It is shown by the author that iterative graph-based ranking algorithms work well on the task of extractive summarization since they do not only rely on the local context of a text unit (vertex), however it takes the information recursively drawn from the entire text (graph) into account. [6] Proposed two enhancements to the above work investigated earlier by adding two more features to the existing one. Firstly, discounting approach was introduced to form a summary which ensures less redundancy among sentences.
Secondly, position weight mechanism has been adopted to preserve importance based on the position they occupy. They investigated in depth, two graphical methods for multi document summarization namely Sentence Rank (threshold) and Sentence Rank (Continuous). In each case, discounting methods proposed by us are found to be superior as compared to their basic methods and the proposed Sentence Rank methods which is a combination of discounting technique along and position weight is investigated to be the best. Multiple document summarizations have been widely studied recently. The summary can be either generic or query specific. In a generic summary generation, the important sentences from the document are extracted and the sentences so extracted are arranged in the appropriate order. In a query specific summary generation, the sentences are scored based on the query given by the user. The highest scored sentences are extracted and presented to the user as a summary. Following are the two broad level classifications of text summarization techniques.

Extractive summarization and abstractive summarization. Extractive summarization usually ranks the sentences in the documents according to their scores calculated by a set of predefined features, such as term frequency inverse sentence frequency (TF-ISF), sentence or term position, and number of keywords. Abstractive summarization involves information fusion, sentence compression and reformulation. Early work in summarization dealt with single document summarization where systems produced a summary of one document, whether a news story, scientific article, broadcast show, or lecture. As research progressed, a new type of summarization task emerged: multi-document summarization. Multi-document summarization was motivated by use cases on the web. Given the large amount of redundancy on the web, summarization was often more useful if it could provide a brief digest of many documents on the same topic or the same event. In the first deployed online systems, multi-document summarization was applied to clusters of news articles on the same event and used to produce online browsing pages of current events. A short one paragraph summary is produced for each cluster of documents pertaining to a given news event, and links in the summary allow the user to directly inspect the original document where a given piece of information appeared. Approaches presented so far are examples of pure techniques to apply, in order to develop summarization systems.

The predominant tendency in current systems is to adopt a hybrid approach and combine and integrate some of the techniques mentioned before (e.g. cue phrases method combined with position and word frequency based methods in [24], or position, length weight of sentences combined with similarity of these sentences with the headline. As we have given a general overview of the classical techniques used in summarization and there is a large number of different techniques and systems, we are going to describe in this section only few of them briery, considering systems as wholes.

There are two limitations with most of the existing multi-document summarization methods:

i. They work directly in the sentence space and many methods treat the sentences as independent of each other. Although few works tries to analyze the context or sequence information of the sentences, the document side knowledge, i.e. the topics embedded in the documents are ignored.

ii. Another limitation is that the sentence scores calculated from existing methods usually do not have very clear and rigorous probabilistic interpretations. Many if not all of the sentence scores are computed using various heuristics as few research efforts have been reported on using generative models for document summarization.

Recent work in multi-document summarization has leveraged information about the topics mentioned in a collection of documents in order to generate informative and coherent textual summaries. Traditionally, MDS systems have created informative summaries by selecting only the most relevant information for inclusion in a summary. In a similar fashion, coherent summaries have been created by ordering information extracted from texts in a manner that reflects the way it was originally expressed in a source document.

In recent years, graph-based ranking methods have been investigated for document summarization, such as TextRank (Mihalcea and Tarau, 2004; Mihalcea and Tarau, 2005) and LexPageRank (Erkan and Radev, 2004). Similar to PageRank (Page et al., 1998), these methods first build a graph based on the similarity relationships between the sentences in a document and then the saliency of a sentence is determined by making use of the global information on the graph recursively. The basic idea underlying the graph-based ranking algorithm is that of “voting” or “recommendation” between sentences.
IV. Proposed System

In the proposed system we used vector space model for document indexing. In the vector space model document is represented by Vector of terms as follows.

- Words (or word stems)
- Phrases (e.g. computer science)
- Removes words on “stop list”

Correlations between term vectors imply a similarity between documents. For efficiency, an inverted index of terms is often stored. In the vector space model we used term frequency which count of time terms occurs in the document. The more times a term occurs in document the more likely it is that it is relevant to the document. Document frequency the more a term occurs throughout all documents, the more poorly it discriminates between documents. The term frequency and inverse document frequency High value indicates that the word occurs more often in this document than average.

This document summarization system used context based document indexing based on vector space model. This document indexing model work with the document frequency and term frequency. DF and TF model gives document indexing weight which is used for document summarization. We divide the proposed system in sum phases. Phase one contain document preprocessing and suffix stripping using Porter streamer algorithm. Document preprocessing reduces the size of document and removes the frequently used words in those documents. Second phase of proposed system construct vector space model using result of first phase that preprocessed document and its term. By using this vector space model we calculate the document context in the third phase. Third phase is important step used to calculate document context. The last phase of system summarized the whole document using result getting from previous steps. We describe this all phase in details as follows.

A. Document Preprocessing

The Most frequently used words in English are useless in Text mining. Such words are called Stop words. Stop words are language specific functional words which carry no information. It may be of the following types such and, the, is. This phase involves two crucial steps namely stemming and stop word removal. For word streaming we used Porter streamer algorithm. The Porter stemmer was developed by Martin Porter in 1980. Porter stemming algorithm is a context sensitive suffix removal algorithm and is the most widely used of all the stemmers. The stemmer is divided into a number of linear steps that are used to produce the final stem. A consonant is a letter other than A, E, I, O, U and Y preceded by a consonant. A vowel is any letter that is not a consonant.

A list of consonants greater than or equal to length one will be denoted by a C and a similar list of vowels by a V. Any word can be represented by the single form; [C] (VC)m [V] Where the superscript m denotes m repetitions of VC and the square brackets [ ] denote the optional presence of their contents. The value m is called the measure of a word and can take any value greater than or equal to zero, and is used to decide whether a given suffix should be removed. All such rules are of the form S1 -¿ S2 means that the suffix S1 is replaced by S2 if the remaining letters of S1 will satisfy the condition. The first step in the algorithm is the most complex and is separated into three parts in the original definition, 1a, 1b and 1c. The first part 23 deals with plurals, for example ses → ss and removal of s. The second part removes ed and ing, or performs eed where appropriate. The second part continues only if ed or ing is removed and transforms the remaining stem to ensure that certain suffixes are recognized later. The third part simply transforms a terminal y to an i. The remaining steps in this stemmer contain rules to deal with different order classes of suffixes, initially transforming double suffixes to a single suffix and then removing suffixes provided the relevant conditions are met. Following code show the working of porter’s streamer algorithm.
Porter Stemmer Algorithm:

integers ( p1 p2 )
booleans ( Y found )
routines ( 
shortv R1 R2
Step 1a Step 1b Step 1c Step 2 Step 3 Step 4 Step 5a Step 5b )
externals ( stem )
groupings ( v v WXY )
declare v "aeiouy" define v WXY v + "wxY"
backwardmode ( 
declare shortv as ( non-v WXY v non-v )
declare R1 as p1 <= cursor
define R2 as p2 <= cursor
define Step 1a as
( [substring] among (//Rule 1a)
declare Step 1b as
substring
among (//Rule 1b))
declare Step 1c as (//Rule 1c)
declare Step 2 as [substring] R1 among (//Rule2a)
declare Step 3 as (//Rule 3)
declare Step 4 as (//Rule 4)
declare Step 5a as (//Rule 5a)
declare Step 5b as (//Rule 5b)
declare stem as ( unset Yfound do
( [`y0] 'Y' set Y found)
do repeat (goto (v ['y']) 'Y 0setY found)
p1 = limit
p2 = limit
do (gopast v gopast non-v setmark p1 gopast
v gopast non-v setmark p2) backwards(
do Step 1a do
Step 1b do
Step 1c do
Step 2 do
Step 3 do
Step 4 do
Step 5a do
Step 5b do
Yo found repeat (goto (['y']) 'y')))

Term frequency is related to how often the specific term appears within a document, and is calculated so to be independent of the length of the document. The equation used, in this study, to calculate TF is given below.

\[
TF = \frac{\log_{e} (t+1)}{\log_{e} (l)}
\]

In the equation t is the frequency of this term within the document, and l is the length of the document in terms. The term frequency is increased by 1 so that values of 0 are not produced by this equation. As an example if we have two documents both of which are ten terms in length, and both include the word algorithm, although the second document contains it twice, and the first only once then: TF1 = 0.30 and TF2 = 0.48. This clearly shows that documents which contain a term more frequently (with regard to document length), occur higher in the ranking. IDF measures how frequent the current term is across the entire document collection. The equation used to calculate IDF, in this study, is given below. In this equation N is the number of documents in the collection, and n is the number of documents which contain the current term.

As an example if we have two words algorithm and evaluation, and a document collection of 10 documents, then if algorithm appears in one document and evaluation in five: IDF algorithm = 2.30 and IDF evaluation = 0.69. This clearly shows that terms which only occur in a few documents are rated higher than more commonly occurring words. The two ranking algorithms used in this study are TF and . Using just TF leads to a very crude ranking algorithm which makes no real attempts to implement. Using , although it may not appear so at first glance, does implement the cosine equation. TF IDF For a single term the TF IDF can be defined and simplified as in Equation. This is still not the same as , but if we now generalize to all the tokens in a query-document combination we get . Now it should be clear to see that the above, is proportional to cosine(Q,Dd), with log(l) being the document weight, Wd, w being the terms in the query and document, and Wq being omitted as it is constant for a given query. So the equation given below is used in the implementation of the system to calculate TF and IDF.

B. Construction of Vector space model

Implementing the vector space model requires the use of two equations: to calculate Term Frequency, TF, and Inverse Document Frequency, IDF.

C. Calculating Document Context

To represents a position in source file. For languages where the source file may not be present, a document context identifies a position in a document typically generated by the run-time environment.
For example, a scripting engine might create a document from script. For extra information, see document Position. Describes a position in source document that corresponds to a code context. The symbol handler maps a code context to documentation context, using information generated by a compiler or interpreter. By using this TF and IDF we get the document context.

\[
TF\times IDF(Q,D) = \frac{1}{\log (l)} \sum_{w \in Q \land D} \log_e (r + 1) \log_e \left( \frac{N}{n} \right)
\]

**D. Document Summarization Based On Vector Space Model Results**

This is a final step of our system. Using this vector space model we get the words having higher weight. Using this weight we summarize the document. Using the vector space model we get the context of the document which is useful for the word ranking in the document. We can retrieve the more important information from the document with the help of this context modeling. This document summarization is filter using following things to identify sentences that should belong to summary, several features have been taken into consideration.

Sentence-Length Cut off Feature p if the sentence length is greater than 4 words, only then it is taken into consideration.

Position Feature Sentences have been given some weight based on their position in the paragraph whether it is in initial, middle or final. Keywords The sentence weight also depends not only on the number of keywords present in it but also the weight of each keyword. Upper Case Feature – Sentences containing upper case words have been provides additional weight as it is probable that they may contain proper nouns. Sentences with higher weight are taken as the similar sentences for the summary and arranged in the order they appear in the document yielding the required sum. The rearrangement is challenge in the case of multiple documents. In that case, sentences are kept in the place at which they appear in original document (initial/middle/final). This rearrangement technique provides clear and clean results.

**V. MATHEMATICAL MODEL**

1) Source File.
2) Document Preprocessing.
3) Vector Space Model.
4) Document Context.
5) Document Summarization.

Let the system be described by S,

\[S = \{D, SF, DP, VS, DC, DS\}\]

Where

S: is a System.
D: Set of Input Dataset.
SF: Source File.
DP: Document Preprocessing.
VS: Vector Space Model.
DC: Document Context.
DS: Document Summarization.
D= d1, d2... dn
Y= SF, DP, VS, DC, DS
Y is a set of techniques use for

A Context-Based Word Indexing Model for Document summarization.

F is the set of Function.
F= f1; f2; ::; fn
Fn1: Source File.
Fn2: Document Preprocessing.
Fn3: Vector Space Model.
Fn4: Document Context.
Fn5: Document Summarization

**VI. PERFORMANCE EVALUATION AND RESULT ANALYSIS**

The Most of the system are firstly preprocessing the document. In that removing the stop words and striping the words which are present the document. The vector space model obtains the vector by using this preprocessed document term. The each term having context in the document. So vector space matrix which contain the TF and IDF value of terms and documents. So I compare my current system gives better result than other existing system it’s obtain summary with preserving the meaning in document. The table shows and graph shows the original file before summarization and after summarization. Here my system shows it’s reduced the size of the document is near about 70% to 80%. Large document file are processed and summarized very clearly, and reduction is also more.

<table>
<thead>
<tr>
<th>File No.</th>
<th>File Name</th>
<th>Original Document Size(byte)</th>
<th>Summarized size(byte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Economic</td>
<td>10353</td>
<td>3253</td>
</tr>
<tr>
<td>2</td>
<td>Engineering</td>
<td>15466</td>
<td>4976</td>
</tr>
<tr>
<td>3</td>
<td>NW testing</td>
<td>5948</td>
<td>1705</td>
</tr>
<tr>
<td>4</td>
<td>Egypt</td>
<td>20897</td>
<td>7391</td>
</tr>
<tr>
<td>5</td>
<td>Washington</td>
<td>1573</td>
<td>782</td>
</tr>
</tbody>
</table>

Table 1: Summarization in Document
Thus we have investigated different methods for document summarization and have proposed a novel approach using vector space model for context-based document summarization. This document indexing model works with the document frequency and term frequency. The concept of using vector space model was used to modify the indexing weights of the document terms. Analysis of some of the documents and the corresponding summary figured out the specific advantage offered by the proposed vector space model-based context sensitive indexing. Vector space model provide better summary based on context of sentences than other summarization method.

Acknowledgment

I am very much thankful to my respected project guide and Head of Dept. Prof. Durugkar S. R. for his ideas and help, proved to be valuable and helpful during the creation of this paper and set me in the right path. I am also very much thankful to PG Co-coordinator Prof. Shaikh I. R. for helping us while selecting and preparing for paper work. I would also like to thank all the faculties who have learned all the major concepts that were involved in the understanding of techniques behind my paper. Lastly, I am thankful to my friends who shared their knowledge in this field with me.

REFERENCES


