Marathi Extractive Text Summarizer using Graph Based Model

Vaishali V. Sarwadnya
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, India
vaishalisarwadnya@gmail.com

Sheetal S. Sonawane
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, India
sssonawane@pict.edu

***Abstract*—Manual summarization of large documents of texts is tedious and error prone. Also, the results in such kind of summarization may lead to different results for a particular document. Thus, Automatic text summarization has become important due to the tremendous growth of information and data. It chooses the most informative part of text and forms summaries that reveal the main purpose of the given document. It yields summary produced by summarization system which allows readers to comprehend the content of document instead for reading each and every individual document. So, the overall intention of Text Summarizer is to provide the meaning of text in less words and sentences. Summarization can be categorized as: Abstractive summarization and Extractive summarization. This case study is based on an extractive concept implemented on the studied models. Numerous automatic text summarization systems are handy today for English and other foreign languages. But when it comes to Indian languages, we observe inadequate number of automatic summarizers. Our efforts in this direction are mainly for developing automatic text summarizer for marathi Language. We look forward to evaluate the obtained summaries using ROUGE metric. This paper describes a multi document marathi extractive summarizer.**

Keywords-Marathi Text Summarizer, Extractive Summarization, Graph based model, Feature Extraction, TextRank.

I. Introduction

Text Summarization is a technique of condensing actual text into abstract form which provides same meaning and information as provided by actual text. It chooses the most informative part of text and forms summaries that reveal the main purpose of the given document. It yields summary produced by summarization system which allows readers to comprehend the content of document instead for reading each and every individual document. So, the overall intention of text summarizer is to provide the meaning of text in less words and sentences. Summarization systems can be sorted into two categories: Abstraction-based summarization and Extraction-based summarization.

Extractive summaries involve extracting appropriate sentences from the source text in sequential manner. The appropriate sentences are extracted by applying statistical and language reliable features to the input text. But there is limit in extraction. The extracted phrases and sentences are in chronological order. While, abstractive text summaries are formed by enacting natural language understanding concepts. This kind of summarizer generally, incorporates terms that do not exist in the document. It aims to imitate methods used by humans, such as representing a concept that is available in the original article in a better and more comprehensive way. It is effective summarizer however, it is very difficult to implement.

A. Abstractive Summarization

In this type of summarization, language understanding tools are used to generate a summary. The main focus is on choosing phrases and lexical chains from the documents. General steps used in this technique are withdrawing basic features, obtaining the relevant information, revising and reducing information. Since the formulation of this technique in mathematical or logical form is cumbersome, it is referred as a complex technique to implement . Also, the quality of generated summaries relies on the depth of linguistic strength. These techniques are generally categorized as Structured and Semantic based. Structured based approaches, obtain most significant information from the documents through cognitive schemas such as templates, frames and scripts [3].

Semantic based approach makes use semantic depiction of documents which is further used to supply into natural language generation (NLG) system as input. This method focuses on obtaining noun phrases and verb phrases by managing linguistic data. Phrases thus obtained are then related to concepts, attributes and relations of a domain-specific ontology. The important document areas (like sentences or paragraphs) are selected by using ontology-based annotations and clustering techniques. The information obtained as an outcome is used to transform those areas into semantic representation. This NLG system takes this representation as input and then produces abstracts.

Word parsing, rhetorical parsing, statistical parsing and a mixture of all are some of the common techniques used. The drawbacks of abstractive summarization approach are

1) Machine generated automatic summaries would result into lack of clarity even within a sentence as sentence synthesis is an emerging field.
2) As they heavily depend on the adaptation of internal tools to perform information extraction and language generation, they are difficult to replicate [3].

3) Abstracting is not easier as it needs semantic understanding of content present in the document.

B. Extractive Summarization

Extractive summaries are formed by extracting crucial text (sentences or passages) from the document, based on features such as word/phrase frequency, location or cue words to locate the sentences to be extracted. The most important text is treated as the most frequent or the most suitably positioned text. Such an approach thus, shuns the labour on depth of content understanding. They are theoretically simple and easy to implement.

This system constitutes two important phases, which are : Pre-Processing and Processing phase

1) Representing the text in a structured manner is the main aim of Pre-Processing phase .
2) Processing phase represents various features that decide the importance of sentences.

Certain statistical features used for marathi language are keywords identification, sentence length feature and numerical literals count feature.

An equation of summation of feature weights is used to generate score of sentences and high scored sentences in a specific order of input text are considered for final summary. This report describes multi document marathi extractive summarizer. It is text extraction based summarization system which is used to summarize the marathi document by retaining the appropriate sentences based on features.

II. Related Work

Various automatic text summarization systems are accessible for most often used languages. Most of these text summarization systems are for English and other foreign languages. Moreover, technical documentation is often minimal or even absent. When it comes to Indian languages, automatic summarization systems are very limited. Very little research and work has been done in text summarization for the Indian language marathi (an Under-Resourced language). Various fields make use of text summarization systems like, education field, social media (news articles, twitter, facebook messages), search engines, bio-medical field, government offices, researcher, etc [2].

Virat V. Giri and et al. reviewed text summarizers based on various Indian languages and their performances. They studied and proposed summarization method for marathi in detail wherein marathi stemmer, marathi proper name list, English-Marathi noun list, marathi keywords extraction, marathi rule based named entity recognition etc. for pre-processing of text followed by processing of text [1]. Sheetal Shimpikar and et al. studied various techniques of text summarization for various Indian languages [2]. Sunitha C and et al. worked on Abstractive summarization methods that are used for Indian languages. They explained Abstractive summarization technique, classified in two approaches such as structure based approach and semantic based approach [3]. Hamzah Noori Fejer and et al. gave a major contribution by proposing a combined approach of clustering technique and extracting keyphrases. They have proposed a new approach of clustering which combines hierarchical and k-means clustering. The results obtained from their experiments proved the proposed model gives better performance when compared with existing ones [4]. An unsupervised approach for marathi stemmer has been discussed by Mudassar Majgaonker and et al. [6]. The present work on text summarization of marathi text with question based system using rule based stemmer technique. For generating question, we used rule based approach of abstractive text summarization and POS tagger, NER tools and rule based stemmer. Here marathi text is taken as input, on it POS tagger is applied and then questions are generated for the given input as per marathi language rules by Deepali K. Gaikwad and et al. At this stage they have framed rules of stemmer only for ẃhoťype questions [5]. Thus it can be extended to learning all What type questions too.

Mangesh Dahale proposed text summarizer using inverted indexes [9]. Jayshri Patil and et al. reviewed different approaches of Named Entity Recognition (NER) and discussed issues and challenges arising in Indian languages [8]. Pooja Pandey and et al. discussed extraction of root words using morphological analyzer for devanagari script [11]. Aishwarya Sahani and et al. contributed to automatic text categorization of marathi language documents [7]. Rafael Ferriera et. al used four dimensional graph based model for text summarization which relies on four dimensions(similarity,semantic similarity,co-reference,discourse information) to create the graph [16]. Federico Barrios et. al used variations in similarity measures along with TextRank for summarization [15]. Our work includes use of TextRank along with positional distribution of sentence scores and considering thematic similarity which gave promising results.

Next sections describe challenges that are faced while dealing with influential language like marathi and method to implement an extractive summarizer using graph based algorithm-TextRank.

III. Challenges observed for marathi Language

Very little work has been done for constructing a text summarizer for marathi language. marathi language is morphologically very rich. A single stem word may have numerous morphological variants like, for word 'maati' we have maatichi, maatine, maativar, etc. morphological variants. Hence, it is required to study its morphology and pre-process the document before extracting features and then process those features which are important in each of the sentences.

There are certain challenges that need to be faced while constructing an automatic text summarizer for marathi language:

1) Marathi text is inflectional and morphologically rich [5]. Marathi is agglutinative language. Unlike english prefixes and suffixes are added to root words in marathi to form meaningful contexts.
2) Non-availability of large Gazetteer
Its difficult to construct gazetteer for such a influential and morphologically variant language.
3) Marathi language do not have capitalization. Unlike english words have capital letters in beginning for proper names or representing important words.
4) Scarcity of resources
It becomes difficult to obtain test data as very limited resources are available as compared to other natural languages used worldwide.
5) Ambiguities in names where the words have multiple interpretations while analyzing the text containing words. For example,
Figure 1 refers same name describing a person and

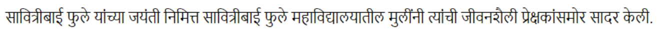

सावित्रीबाई फुले यांच्या जयंती निमित्त सावित्रीबाई फुले महाविद्यालयातील मुलींनी त्यांची जीवनशैली प्रेक्षकांसमोर सादर केली.

Fig. 1. Example

a place which causes ambiguity for the summarizer to identify that the text interprets two different contexts.
6) Many a times, marathi words containing vowels do not make phonetic difference but differ in writing and spellings [9].
7) Foreign words in some instances of person, organization, location and miscellaneous names that are English words appear in marathi texts which are spelled in Devanagari script.
8) Marathi is spoken using many dialects such as standard Marathi, Warhadi, Ahirani, Dangi, Vadvali, Samavedi, Khandeshi, and Malwani in various regions of Maharashtra. There are specific words used in each dialect to express the text.

IV. Problem Analysis

Let U be the universe of all potential sentences.

$$U = \bigcup_{d \in D} d \tag{1}$$

D be the given set of input documents in database. If we ignore sequence of sentences in documents, then per document, $d \in D$ and $d \subset U$.
Given, a set of input documents D, containing w words in all, our aim is to form a summary document $\hat{d}$ (containing atmost x words) where $x \ll w$. Conceptually,

$$\hat{d} = max_{d' \subset U} sim(d' \in D) \tag{2}$$

Extractive summarization is restricted version. Under some considerations, the problem is lowered to weighted set cover problem. We select a set $\hat{d} \subset U$ of sentences such that $\hat{d}$ maximally achieves information in D.

The set cover problem is familiar as NP-Hard, so even if $sim()$ is to be calculated as efficient and optimal, the problem persists to be NP-Hard.

V. Methodology

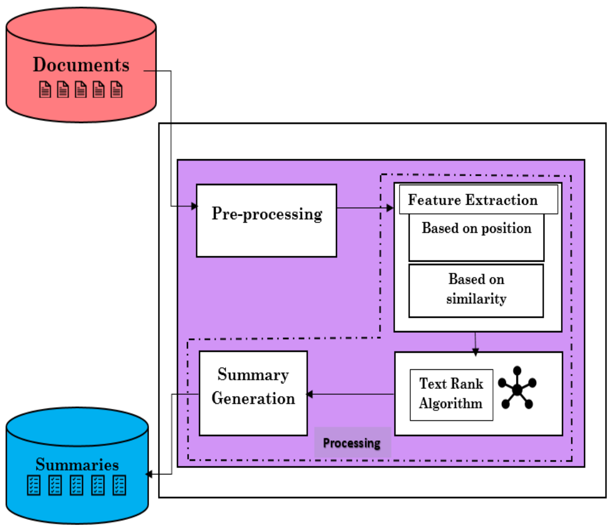


Fig. 2. Architecture

Figure 2 shows our multi-document extractive marathi text summarizer proposed system, which uses graph based model for extracting important sentences for summary generation.

A. Dataset

Multiple documents from dataset will be used to extract texts from documents on different subjects, such as education, politics and news articles. We have 634 documents based on news articles discussing the different topic in marathi language.

The EMILLE (Enabling Minority Language Engineering) which includes monolingual, parallel and annotated corpora for Asian Languages including marathi is used for obtaining multi documents. The system can be divided into two broad stages: Pre-Processing and Processing stage.

B. Pre-processing Stage

Pre-processing stage is essential in text summarization. It results into pre-processed data, which is ideally fit for processing stage. In general pre-processing stage consists of steps to remove punctuation marks, tokenization, stop word removal, stemming, etc. In this section we will discuss various steps used in pre-processing stage.

1) Boundary identification and punctuation marks removal: Every sentence ends with a punctuation mark depending on the nature of sentence, whether interrogative, exclamatory, imperative or declarative. Also, use of quotation marks (” ”,’), commas(,), special characters(&,*,—) and symbols(#,@), etc.

is frequent. But when it comes to extract important words for processing stage, we need to eliminate these punctuation marks. Hence, we use techniques for removal of punctuation marks. The output of this step is punctuation marks free sentences in the document.

2) Stop words elimination: Frequently occurring non essential words for processing in text summarization are generally termed as stop words. In marathi language, we use stop words like shivay, ase, eetar etc. in day to day use. We should eliminate them for obtaining meaningful context while processing the documents. The output of this sentence is stop words free sentences in the document.

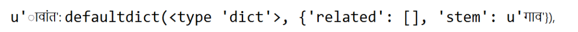
3) Stemming and lemmatization: The process of obtaining stem / radix or root word for morphological variants present in the documents. Lemmatization identifies lemma of a word. It is mapping of verbs into their infinitive and nouns into their singular form. Methods used for constructing stemmers include : Rule based-Porter's Stemmer, Husk stemmer, Unsupervised stemming, suffix stripping-Lovins stemmer, Dawson stemmer, N gram method, HMM method, YASS(Yet Another Suffix Stripper) stemmer etc.

The Extractive Summarization System for marathi till pre-processing stage has been implemented. The generated pre-processed data content is stored in a text file. The results format in which pre-processed data is stored follows a specific structure.

u'[stem suffixes]': defaultdict(⟨ type 'dict'⟩, {'related': [u'word], 'stem': u'[stem word]'),}

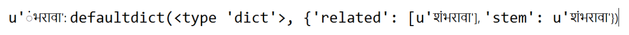
The format shows python's key/value hash table format well known as "dict". The content in a dict can be written as a sequence of key:value pairs within curly brackets { }, for example, dict = {key1:value1, key2:value2, ... }. The "empty dict" is simply an empty pair of curly brackets {}.

Figure 3 and Figure 4 are examples of pre-processed text



```
u'ावांत': defaultdict(<type 'dict'>, {'related': [], 'stem': u'गाव'}),
```

Fig. 3. Example 1



```
u'ंभरावा': defaultdict(<type 'dict'>, {'related': [u'शंभरावा'], 'stem': u'शंभरावा'})
```

Fig. 4. Example 2

from pre-processed file.

C. Feature extraction

The features like SOV (Subject Object Verb - Experimental) verification, sentence positional value (POS tagging), TF-ISF (Term Frequency/ Inverse Sentence Frequency) or TF-IDF (Term Frequency/ Inverse Document Frequency) are extracted from pre-processed sentences. Sentences are further ranked on basis of features extracted.

Sentence Scoring is the process of assigning scores to each of the sentence, denoting its importance. The scoring is carried out depending on the feature extracted.
This phase involves :

1) Identifying the feature specified by the user and extracting sentences accordingly.
2) Assigning score to the extracted sentence using respective sentence scoring technique.

$$Score_{sentence} = \sum_{i=1}^{n} Score_{feature} \qquad (3)$$

General formula for determining sentence score can be formulated as shown in equation 3.
Following are the general approaches followed for english texts: (i) Word Scoring - Focuses on assigning scores to words. (ii) Sentence Scoring - sentence position, title name matching, etc (iii) Graph scoring - examining relationship between sentences [12].

1) Word Scoring: It observes features like Word frequency, TF/IDF, Proper noun, Word co-occurrence and Lexical similarity to determine respective scores of the words in the sentences. Each word obtains a score and by summing up the scores we obtain weights of each sentences.

2) Sentence Scoring: This approach analyzes features of the sentences like Cue-phrases (domain specific phrases), numerical literal count, length of sentences, position, centrality, resemblance with title, etc.

3) Graph Scoring: In this method, score is calculated on the basis of relationship found among the sentences. It includes TextRank model, Bushy path, Aggregate similarity algorithms, etc. When a sentence concerns to another it forms a link with an allied weight between them. The weights are used to obtain the score of sentences.
Bushy path of a node: In this method we find number of links connecting it to other nodes.
Aggregate Similarity: This method uses same approach as bushy path of a node but it sums the weights (similarities) on the links.
TextRank Algorithm: TextRank is an algorithm based upon PageRank for text summarization. In TextRank, the vertices of the graph are sentences, and the edge weights between sentences denotes the similarity between sentences. The TextRank Algorithm is described below:

Algorithm:
1) Obtain the text units that best define the job at hand, and add those units as vertices in the graph.
2) Find relations that connect such text units to draw edges as connectors between those vertices.
3) Iterate the graph-based ranking algorithm until all possible connections are achieved.
4) Classify vertices based on their final obtained score. Utilize the values attached to each vertex for ranking/selection purpose.

5) Consider, $G = (V, E)$ be a digraph with the set of vertices V and set of edges E. Where, $In(V_i)$ the set of vertices V_i that point to it (predecessors), and $Out(V_j)$ set of vertices V_j that points to (successors). Then, score can be calculated as:

$$Score(V_i) = (1 - d) + d * \sum_j \frac{S(V_j)}{Out(V_j)} \tag{4}$$

where $j \in In(V_i)$ in equation 4.

and d is a damping factor that can be set between 0 and 1, which has the role of combining into the model the probability of moving from a given vertex to another arbitrary vertex in the graph. The factor d is traditionally set to 0.85 [13]

Our work implements TextRank combined with weighted positional distribution of sentence scores and TextRank combined with word/thematic similarity. The pseudo code can be described as below:

```
1) Use naive method for splitting text into sentences and paragraphs.
2) Calculate intersection between each of the 2 sentences:
   (i)Split sentences into words/tokens.
   $s1 = set(sent1.split(""))$
   $s2 = set(sent2.split(""))$
   (ii)If there is no intersection
   $if\ (len(s1) + len(s2)) = 0:$
   $then,\ return\ 0.$
3) Normalize results by taking average number of words.
   $len(s1.intersection(s2))\ /\ ((len(s1) + len(s2))\ /\ 2)$
4) Remove non alphabetical characters.
   $format\_sentence(self, sentence):$
   $sentence = re.sub(r'W+', '', sentence)$
   $return\ sentence$
5) Create a dictionary of the obtained formatted sentences
   with $s : formatted\ sentence;$
   $R : Rank\ of\ the\ sentence$
6) To obtain rank for sentences: (i)Split content into sentences
   $sentences = split\_content\_to\_sentences(content)$
   (ii)Calculate intersection between sentences
   $values[i][j] = intersection(sentences[i], sentences[j])$
   (iii)Build sentence dictionary where, score = number of intersections
   $sentences\_dic[format\_sentence(sentences[i])] = score$
7) Return best sentence in paragraph: (i)Split paragraph into sentences
   $sentences = split\_content\_to\_sentences(paragraph)$
   (ii)Ignore short paragraphs
   $if\ len(sentences) < 2:$
   $return\ ""$

8) Build the Summary: (i)Split content into paragraphs
   $paragraphs = split\_content\_to\_paragraphs(content)$
   (ii)Add best sentences from each paragraph
   $for\ p\ in\ paragraphs:$
   $sentence = get\_best\_sentence(p, sentences\_dic).strip()$
   $if\ sentence : summary.append(sentence)$
   $return\ ("").join(summary)$
```

To implement TextRank together with sentence position:

1) Create a weighted positional distribution of sentence scores based on their position in the text corpus.
2) Generate a graph based ranking model for the tokens, return keyphrases that are most relevant for generating summary.
3) Generate summary and write to a file.

To implement TextRank together with thematic similarity:

1) Obtains similarity between sentences.
2) Generates scores for sentences based on similarity found between words.
3) Generate graph for input text using TextRank.
4) Generate summary.

VI. Evaluation Measures

Evaluation can be done Qualitatively or Quantitatively based on the methods used. Qualitatively, the objective is to find a summary grammatically and semantically correct, that is relevant, and that the user can approve (as opposed to disapprove) and/or give a score which constitutes an accepted summary (as opposed to a rejected summary proposal). Quantitatively, the most used way to evaluate the factualness of text summaries is to compare them with human-made summaries. These summaries will be compared with the output of our program using the ROUGE metric [10].

ROUGE i.e. Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary against a reference or a set of references (human-produced) summary. In our work we have compared the system generated summaries with two sets of human generated summaries. Based on the n-gram overlap between candidate and reference summaries ROUGE calculates the scores of a candidate summary [14].

Based on n-grams used in the evaluation ROUGE-N consists of different metrics, such as ROUGE-1, ROUGE-2, ROUGE-3 and so on . N varies between 1-4.
The recall (R) measure is calculated by the proportion of n-grams from reference summaries occurring in a candidate summary, the precision (P) is calculated by the proportion of n-grams from a candidate summary occurring in reference summaries and F-score can be then calculated by combining recall and precision into one metric.

$$Recall = |\frac{G_{ref} \cap G_{can}|}{|G_{ref}|} \tag{5}$$

$$Precision = |\frac{G_{ref} \cap G_{can}|}{|G_{can}|} \quad (6)$$

ROUGE-N scores can be calculated as follows:

$$ROUGE_{F-score} = \frac{2 * ROUGE_{recall} * ROUGE_{precision}}{ROUGE_{recall} + ROUGE_{precision}} \quad (7)$$

where, G_{ref} refers to the grams of reference summary and G_{can} refers to the grams of candidate summary in equation (5), (6) and (7).

Our work uses ROUGE-N metric for evaluation, where N varies from 1-4. The result shows that we have achieved good f-scores for both methods. TextRank when used along with thematic similarity gives best precision, while TextRank with positional distribution gives best recall. Table 1 shows evaluation results obtained on our dataset. It consists of F-scores generated for using ROUGE N, where N varies from 1-4. Figure 5 shows plotting of F-scores with ROUGE-N results for N=1-4.

Technique	**ROUGE-1**	**ROUGE-2**	**ROUGE-3**	**ROUGE-4**
TextRank+ Positional	0.8004	0.7909	0.785	0.7799
TextRank+ Similarity	0.8684	0.8602	0.8521	0.8447

TABLE I
ROUGE RESULTS

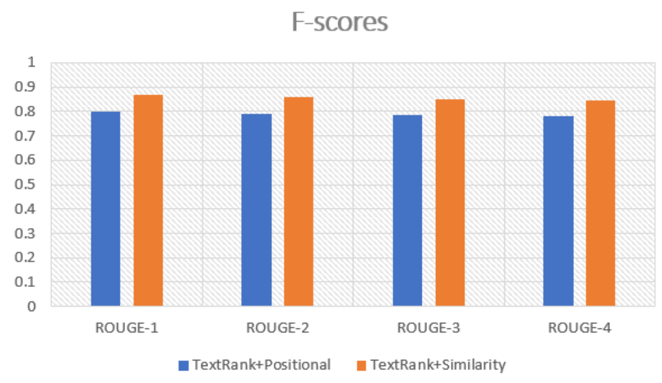


Fig. 5. ROUGE Results

VII. CONCLUSION AND FUTURE SCOPE

With the tremendous increase in the amount of content accessible online, there is a need of fast and effective automatic summarization system. The most important steps in this system approach are feature extraction, scoring and graph generation. This system can be used in various fields like education, in search engines to improve their performances, for Marathi news clustering, Question generation purpose and many other application oriented areas, etc.

The scope can be extended to abstractive summarization by including NLP features and implementing more scoring techniques. Use of semantic ranking can also be done to obtain meaningful summaries.

ACKNOWLEDGMENT

We sincerely thank our Head of Department Dr. R. B. Ingle and Post Graduation Coordinator, Department of Computer Engineering Dr. A. R. Buchade for their constant support, providing all the help, motivation and encouragement from beginning till end to make this work a grand success.

REFERENCES

[1] Virat V. Giri, Dr.M.M. Math and Dr. U. P. Kulkarni , A Survey of Automatic Text Summarization System for Different Regional Languages in India *Bonfring International Journal of Software Engineering and Soft Computing, Vol. 6, Special Issue, October 2016*

[2] Sheetal Shimpikar and Sharvari Govilkar, A Survey of Text Summarization Techniques for Different Regional Languages in India, International Journal of Computer Applications, Vol. 165, No. 11, May 2017

[3] Sunitha C, Dr. A Jaya and Amal Ganesh, A Survey of Abstractive Summarization Techniques in Indian Languages, 2016

[4] Hamzah Noori Fejer and Nazlia Omar, Automatic Multi-Document Arabic Text Summarization Using Clustering and Keyphrase Extraction *ICIMU IEEE 2014 International Conference*,978-1-4799-5423-0.

[5] Deepali K. Gaikwad, Deepali Sawane and C. Namrata Mahender, Rule Based Question Generation for Marathi Text Summarization using Rule Based Stemmer*IOSR Journal of Computer Engineering (IOSR-JCE),* e-ISSN: 2278-0661. 2015

[6] Mudassar Majgaonkar and Tanveer Siddiqui, Discovering suffixes: A case study for Marathi Language *(IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 08, 2010, 2716-2720*

[7] Aishwarya Sahani, Kaustubh Sarang, Sushmita Umredkar, and Mihir Patil, Automatic Text Categorization of Marathi Language Documents *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (5) , 2016, 2297-2301*

[8] Ms. Jayshri Arjun Patil, Ms. Poonam Bhagwandas Godhwani, Review of Name Entity Recognition in Marathi Language *IJSART - Volume 2 Issue 6 , June 2016*

[9] A report on Text Summarization for Compressed Inverted Indexes and snippets by Mahesh Dangale *CS 297 Report* July 2013.

[10] Feifan Liu, Yang Liu, Exploring Correlation between ROUGE and Human Evaluation on Meeting Summaries*IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*

[11] Pooja Pandey, Dhiraj Ahim, Sharvari Govilkar, Rule based Stemmer using Marathi WordNet for Marathi Language*International Journal of Advanced Research in Computer and Communication Engineering, Volume 5, Issue 10, 2016*

[12] Rafael Ferreira, Luciano de Souza Cabral, Assessing sentence scoring techniques for extractive text summarization*Expert Systems with Applications, Elsevier 2013*

[13] TextRank: Bringing order into texts, Mihalcea, Rada., and Tarau, Paul. (2004), In Conference on empirical methods in natural language processing, Barcelona, Spain.

[14] Rouge: A package for automatic evaluation of summaries, Lin, C. Y. In Text summarization branches out, Proceedings of the ACL-04 workshop (Vol. 8).

[15] ”Variations of the Similarity Function of TextRank for Automated Summarization”, Federico Barrios, Federico Lopez, Luis Argerich, Rosita Wachenchauzer, arXiv, 2017.

[16] ”A Four Dimension Graph Model for Automatic Text Summarization”, Rafael Ferreira, Frederico Freitas, Luciano de Souza Cabral, Rafael Dueire Lins, Rinaldo Lima, Gabriel Franca, Steven J. Simske, and Luciano Favaro, IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT) 2013.