Grenze International Journal of Engineering and Technology, Jan Issue

A Study and Implementation of Various Phases of Pre-Processing Techniques in Hindi Languages

Atul Kumar[1], Dr Vinodini Katiyar[2] and Dr Pankaj Kumar[3]
[1]DSMNRU/Information Technology Lucknow,India
[1]atulverma16@gmail.com
[2]DSMNRU Lucknow, India
[2]drvinodini@gmail.com
[3]SRMGPC, Lucknow, India
[3] pk79jan@gmail.com

***Abstract*—There are a lot of text available on the internet so it is very difficult to extract important information from large available text. Text summarization is a process of converting large information into smaller one by keeping all important points. Text Summarization involves two steps Text Pre-Processing and Processing. In this proposed paper we are going to discuss the different steps involved in Text Pre-Processing of Hindi Language and then we will find out the Pre-Processing time of given Hindi data set and then we will compare the Pre-Processing time of different sizes of datasets.**

***Index Terms*— Text Summarization, Stop words, Stemming, Tokenization, Delimiter.**

I. INTRODUCTION

In this new era, where large amount of information is available on internet it is very tough to find important facts. There are every day online papers which are accessible containing news on different themes like legislative issues, sports, Entertainment and so on In addition updates of specific news can show up in the paper for several days and in various papers with varieties. What's more, ordinarily individuals simply need a significance of what the news case is about as they don't have the opportunity to peruse all the news stories totally because of absence of time. To overcome from this problem there is a need of summarizer. Summarization is a process of shortening the text by keeping the important points. It is of two types:

1. Extractive Summarization
2. Abstractive Summarization

Extractive summarization generates the summary by selecting the keywords from given dataset while abstractive summarization generates the summary by analysing the given dataset into entirely new phrases by keeping the same meaning of the input data set.

Basically, each and every summarization consists of Pre-Processing and processing phase. In this paper we have discussed the different Pre-processing techniques for Hindi language. We have thought about Hindi as a language of Study. It is written in the Devanagari content which has biggest data set. Hindi is the most famous language of India. It the local language of the vast majority living in North India.

Grenze ID: 01.GIJET.7.1.507_2

II. LITERATURE REVIEW

Atul Kumar et al.[1] in 2019 proposed a framework for generation of summary for Hindi Language using Extraction based text summarization approach. In this approach sentences are selected on the basis of their scores.

Atul Kumar et al. [2] in 2018 has given a survey based paper for Text summarization in different languages. He has discussed the merits and demerits of different approaches of summarization n different languages.

Atul Kumar et al.[3] in 2019,Sentiment analysis is the mining of emotions associated with the text as positive, negative and neutral.The main obstacle in sentiment analysis is the presence of Sarcasm in the text. Sarcasm is a particular form of text which has different sentiment to the true sentiment in the text.

According to Atul kumar et al.[4], Automatic generation of textual description of an image is a challenging problem of the computer vision domain. This can be attributed to the fact that generating textual description of an image requires the application of image processing techniques in conjunction with natural language processing methods. This paper proposes a model to automate the process of generating textual description of an image using the concepts of neural networks. The proposed model can generate human-like sentence for expressing an image. The proposed model is validated using the MSCOCO dataset. The obtained results clearly indicate that the performance of the proposed system increases with an increase in convolution layers as well as increase in depth.

Conroy and O‟ leary [6] made a framework for sentence extraction utilizing two procedures named QR and HMM. In QR procedure the significance of the sentence was determined and the sentences having higher significance were added to the rundown. At that point, an overall score of the excess sentences was modified on the grounds that a portion of the leftover sentences was repetitive. Furthermore, the other procedure was the concealed Markov model (HMM) which is a consecutive model for programmed text outline. What's more, in HMM just three highlights were utilized: sentence position, complete no of terms in the sentence, and the likeness of sentence terms in the given archive. Eventually, the assessment was finished by looking at HMM produced outline with the human-created rundown.

Jawline Yew Lin [7] in 1999 additionally attempted to show a framework for sentence extraction utilizing choice trees for making useful conventional/question situated concentrates. Jawline Yew Lin recognized various highlights, for example, : title, pattern, position, tf-idf, question signature, sentence length, lexical network, citation, first sentences, formal person, place or thing, pronoun and descriptor, mathematical information, non-weekend days, and month. The score for all highlights was joined with programmed learning utilizing choice trees and blend work. Lin directed a profound investigation of various highlights impacts through a glass-box. The test result shows that there was not the single component which performs well for question-based rundowns. Highlights like mathematical information give better outcomes for the question which requires an answer in mathematical worth, while non-weekend days and month include gives the best outcome for inquiries like "when?"

S. P. Yong et al [8] in 2005 portrayed a computerized framework that had the capacity of learning by joining diverse later approaches, for example: a factual methodology, sentences extraction and neural organization.

Ladda Suanmali et al [9] in 2009 proposed a framework based on the fluffy rationale. Fluffy guidelines and fluffy sets were utilized for separating huge sentences dependent on sentence highlights.

Different highlights were utilized to register sentence importance and the worth was given somewhere in the range of „0‟ and „1‟ if a specific element was available in the sentence. The element utilized was title word in a sentence, sentence centrality, likeness to initially sentence, watchword, sentence length, sentence position, formal person, place, or thing and numeric information. The estimations of these highlights were given as a contribution to a fluffy framework, which made a yield dependent on highlights and IF-THEN principles characterized by sentence extraction.

Li Chengcheng [10] in 2010 proposed a novel ATS strategy in light of Rhetorical Structure Theory (RST). RST is the documentations of expository connection introduced between two non-overlapping writings called the core (N) and satellite (S). The trial perceptions were utilized to communicate the distinction among N and S, which was communicated by N that what is more imperative to the writer’s reason than S and the Logical core was intelligible autonomous of the S. The whole content was separated into little units called sentences dependent on delimiter like a full stop, commas, any accentuation mark found between sentences.

The whole cycle was based on a diagram, sentences which were less significant was taken out from the diagram and remaining sentences were summed up.

Atul Kumar et al.[11] in 2021 proposed a model in which he had compared a Text Pre-processing time in Hindi languages using Stanza and Spacy and pre-processing time of Stanza is very high as compared to Spacy.

III. STEPS INVOLVED IN TEXT PRE-PROCESSING:

There are number of steps in Pre-Processing of Text Summarization.

A. Boundary Identification

In boundary identification the first step would be to encapsulate the entire text form our dataset which contains the word in Hindi language. Then we store the initial index value of delimiter as 0.
Delimiter: A character or symbol that is basically used for marking the end of a sentence. Delimiter is used to identify the boundary of a particular sentence in case we have a cluster of sentences. Hence it is symbol used for separating the content into different sentences.
Index value: The position of sentences based on their location as starting or end. We often need to give a specific number to a sentence based on its position of where is it located. We then iterate through the string in our dataset content for scanning the symbols i.e., Delimiter. Based on whether we encounter a delimiter in a sentence we are iterating we have two choices:
1. If delimiter is found: We then slice from the last index i.e., position of our dataset and add to our collection of the processed data and then we take an exit from the program. This will then be the termination of the program.
2. If delimiter is not found: We keep on iterating through the remaining sentences which are still needed to be scanned and repeat the similar steps as we did in the previous case which is searching for the delimiter and then add it to our collection of the scanned data and then we exit from the program and we keep on doing this unless the entire sentence is scanned.

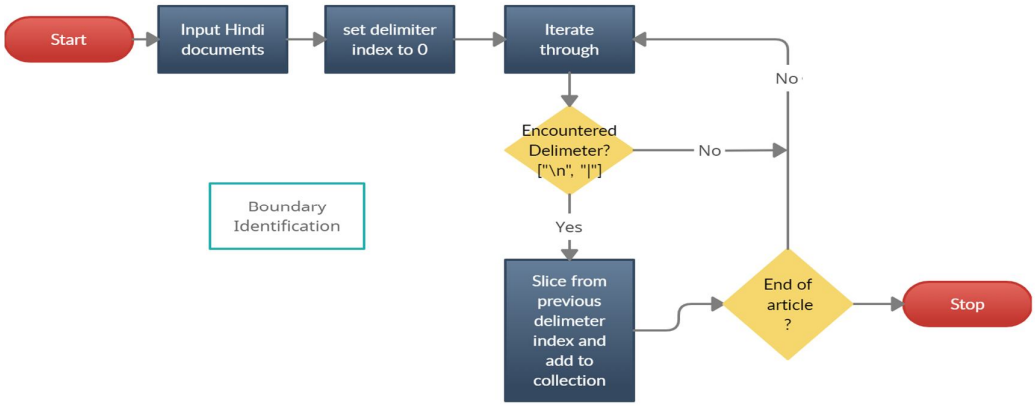


Fig1: Flowchart for Boundary Identification

B. Punctuation removal

In this step we scan through the dataset and create a different symbol table with the list of symbols to prune i.e., to remove those symbols which are no longer needed in our process for pre-processing of dataset. We then iterate through the character of dataset of every sentence in our dataset. We then look after whether we encountered symbols from the table we prepared. We then have two choices based on whether we found the symbol in our scanning or not:
1. If the symbol is found: If the symbol is already present in the table, we have taken reference of then we replace the matching character with the null value (“ ”). We then look into our collection whether it is exhausted or not if the collection is exhausted when just terminate the program. If collection does not get exhausted, then we just repeat the previous steps.
Null value: Empty value which does not contain anything meaningful.
2. If the symbol is not found: We then instruct our program to move to next line, then iterate through every character in that sentence and look for the symbols from that table and based on the response we then decide further steps to be followed. This means if we encounter the symbol in our symbol table then replace the matching character with null value, and then similarly look for the exhaustion of the collection of our dataset.

C. Tokenization

Tokenization means partitioning the sentence or a group of characters in sequence into individual tokens or simply into its constituent units. In case of text summarization, we can break up the pile of clustered sentences

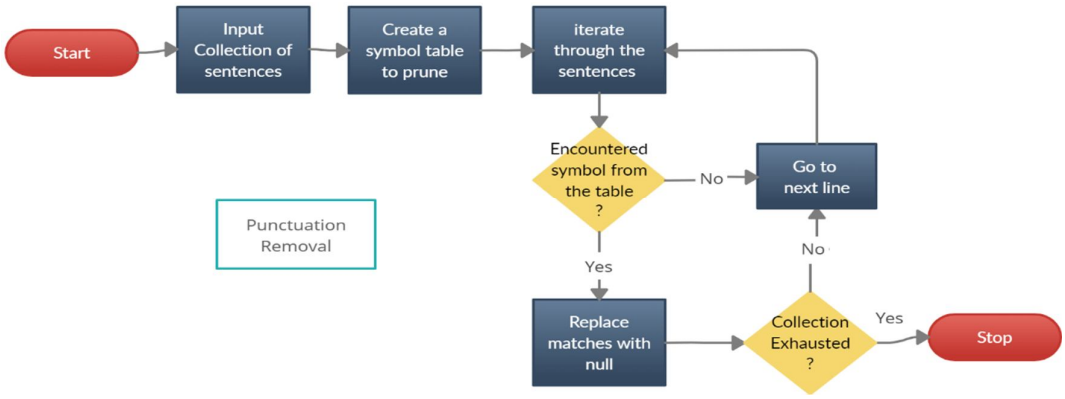


Fig2: Flowchart for punctuation removal

into its individual tokens which is basically the punctuations, phrases or separate words to make our work easier. In this process we take the collection of data to be scanned then iterate through every word in that sentence, we then store initial values of those words as zero, here we have to follow the similar approach of looking after the delimiters so we will be looking after the delimiter if it's found. We will again have two choices:

1. If delimiter is found: We slice the string from the last index (position or ranking) to obtain a word and add to our collection. We then update the index to our current value. After successfully completing these steps we will look if the collection has exhausted or not if yes then we will stoop and execute the code here for this step.
2. If delimiter is not found: We will continue scanning in case, we did not encounter the delimiter and then again look for delimiters. If found, then slice the string from the last index (position or ranking) to obtain a word and add to our collection. We then update the index to our current value. After successfully completing these steps we will look if the collection has exhausted or not if yes then we will stoop and execute the code here for this step. Again, if the delimiter is not encountered then we will repeat this step.

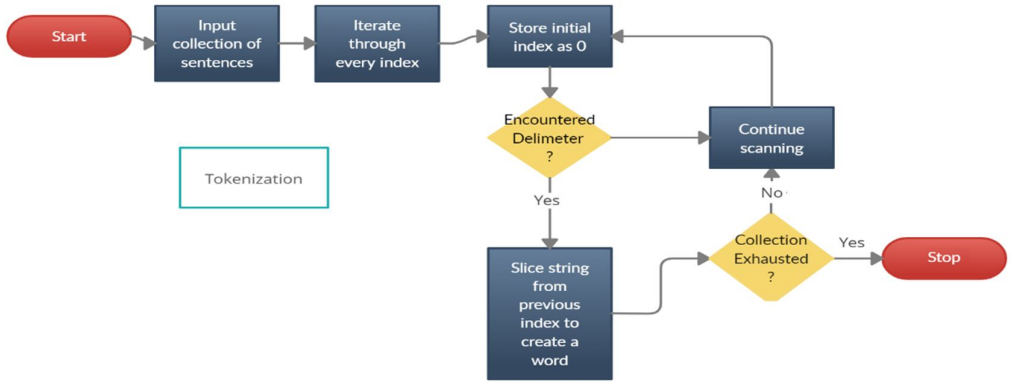


Fig 3: Flowchart for tokenization

D. Stop word filtration

In the stop word filtration, we take the collection of tokens and then we take a hash table of predetermined stop words we then iterate through the list of tokens and see if the current token has a matching hash from table then we have two options:

If the current token has a matching hash: We then remove it from the collection and then check whether it is the end of our collection or not if yes, we then stop the iteration and terminate if not, we continue the iteration iterate through the list of tokens then see if current token has a matching hash from table and then remove it from the collection of word respectively.

If the current token does not have a matching hash from table: We then continue the iteration iterate through the list of tokens see if the current token has a matching hash or not if yes then we remove it from the collection and check whether it is the end of collection and then terminate from the program and if not, we then follow the same steps which is continuing of the iteration and iterating through the list of tokens.

Hash table: it is a table that is used to collect and store key value items so that instead of storing just one value like the stack, array queue or list it stores two values.

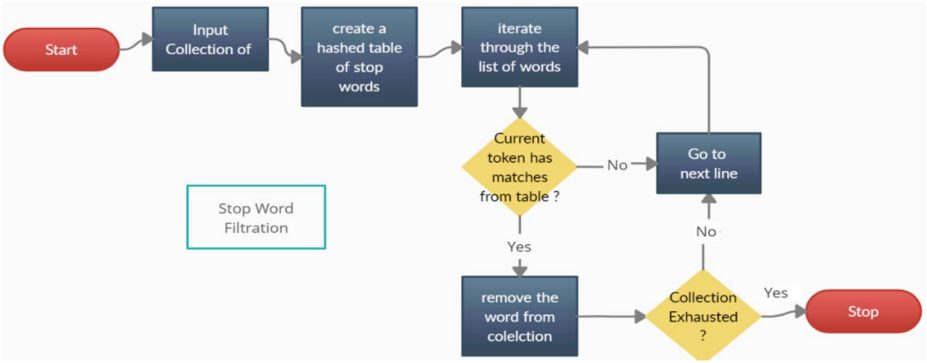


Fig 4: Stop word filtration flow chart

IV. Result and Comparison

For computing pre-processing time we imported time python library. We implemented this function at starting and end of pre-processing step in order to compute the pre-processing time. After that separate readings of the text is taken. The pre-processing time is analysed in units of seconds i.e. the time required in the execution of pre-processing of the various text documents. After applying all the steps of Text Pre-Processing, we will get the final Pre-Processing Time.

For comparison we have considered 10 different Hindi Text Documents of size starting from 10,000 to 1, 00,000 words. So, in this way we have 10 different observations and is quite sufficient to test the relationship of Pre-Processing time with data.

10k: 0.006004810333251953
20k: 0.009007692337036133
30k: 0.011009693145751953
40k: 0.014013051986694336
50k: 0.01701521873474121
60k: 0.019017696380615234
70k: 0.02201986312866211
80k: 0.02602386474609375
90k: 0.029026269912719727
100k: 0.032029151916503906
Total Time elapsed: 0.1852 seconds

The following graph shows the relationship between size of data set and Time taken in Pre-Processing phase, which clearly indicates that the size of data set is directly proportional to the time taken in Pre-Processing phase.

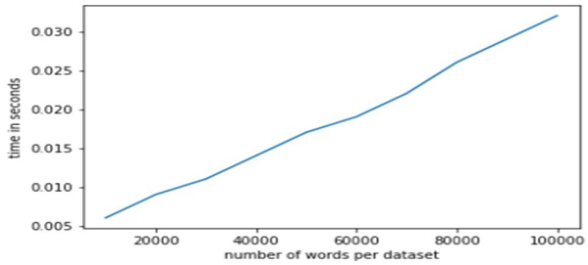


Fig 5: Comparison Graph

For Result analysis we have taken summary in the range of 5% to 40% for different Datasets,
Dataset: patrika_different-enjoyment-of-studies[https://www.patrika.com/opinion/different-enjoyment-of-studies-in-your-tongue-6308646/]

	length	recall	precision	f_score
0	3	0.033333	0.666667	0.063492
1	6	0.050000	0.500000	0.090909
2	9	0.066667	0.444444	0.115942
3	12	0.083333	0.416667	0.138889
4	15	0.100000	0.400000	0.160000
5	18	0.150000	0.500000	0.230769
6	21	0.150000	0.428571	0.222222
7	24	0.166667	0.416667	0.238095

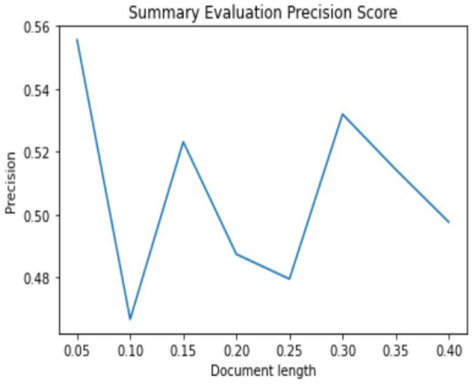
Dataset: patrika_politics-and-democracy[https://www.patrika.com/opinion/politics-and-democracy-6334104/]

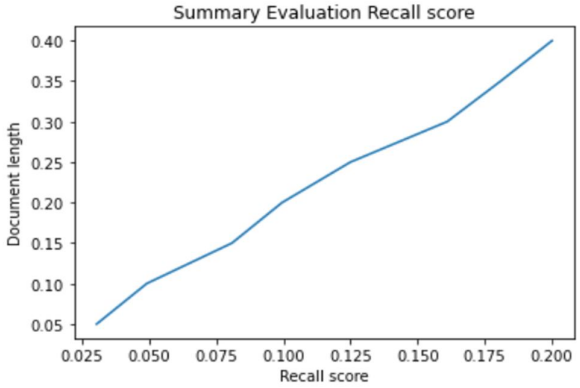
	length	recall	precision	f_score
0	3	0.020408	0.333333	0.038462
1	5	0.040816	0.400000	0.074074
2	8	0.081633	0.500000	0.140351
3	10	0.102041	0.500000	0.169492
4	13	0.142857	0.538462	0.225806
5	15	0.163265	0.533333	0.250000
6	17	0.204082	0.588235	0.303030
7	20	0.244898	0.600000	0.347826

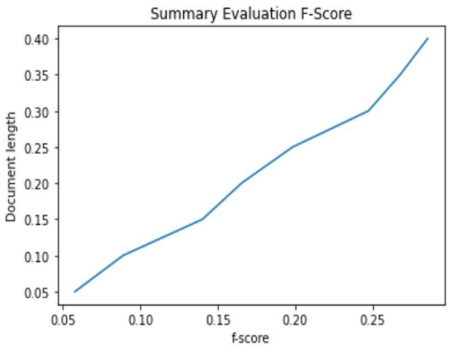
Dataset: patrika-new-education-policy-in-indian-scenario [https://www.patrika.com/opinion/new-education-policy-in-indian-scenario-6307764/]

	length	recall	precision	f_score
0	3	0.037736	0.666667	0.071429
1	6	0.056604	0.500000	0.101695
2	8	0.094340	0.625000	0.163934
3	11	0.113208	0.545455	0.187500
4	14	0.132075	0.500000	0.208955
5	16	0.169811	0.562500	0.260870
6	19	0.188679	0.526316	0.277778
7	21	0.188679	0.476190	0.270270

Precision, Recall and F-Score:







V. Conclusion

In this paper we have developed a light weight Text Summarization. Since there are lot of constraints in Hindi Language like: Spelling variations, Different Punctuations etc. and non-availability of datasets so the final result still requires some optimisation. In future we will work on mentioned parameters so that we will be able to achieve a better result with good accuracy.

References

[1] Atul Kumar and Vinodini Katiyar,"A Study Based Approach For Gist Generation", IJRAR - International Journal of Research and Analytical Reviews (IJRAR), E-ISSN 2348-1269, P- ISSN 2349-5138, Volume.6, Issue 2, Page No pp.623-628, April-2019, Available at : http://www.ijrar.org/IJRAR19K8597.pdf.

[2] Atul Kumar and Vinodini Katiyar, "A Comparative Analysis Of Different Text Summarizers", IJRAR - International Journal of Research and Analytical Reviews (IJRAR), E-ISSN 2348-1269, P- ISSN 2349-5138, Volume.5, Issue 4, Page No pp.610-613, November 2018, Available at : http://www.ijrar.org/IJRAR19D2463.pdf.

[3] Atul Kumar and Vinodini Katiyar," A Comparative Analysis of Sarcasm Detection", in the International Journal of Recent Engineering Research and Development (IJRERD) ISSN: 2455-8761 www.ijrerd.com,Volume 04 –Issue 08 ,August 2019,PP. 104-108.

[4] Kumar A., Kumar R., Shrivastava S.K. (2020) Describing Image Using Neural Networks. In: Khanna A., Gupta D., Bhattacharyya S., Snasel V., Platos J., Hassanien A. (eds) International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing, vol 1087. Springer, Singapore. Https://doi.org/10.1007/978-981-15-1286-5_53.

[5] Shukla, Utkarsh; Verma, Srishti; Verma, Atul Kumar Journal of Computational and Theoretical Nanoscience, Volume 17, Number 1, January 2020, pp. 499-504(6), American Scientific Publishers, https://doi.org/10.1166/jctn.2020.8697.

[6] M.J Conroy and o'leary, "Text summarization via hidden markov models," In Proceedings of SIGIR '01, pp.406-407, 2001.

[7] Chin-Yew Lin, "Training a selection function for extraction," Proceedings of the eighth international conference on Information and knowledge management, ACM, pp. 55-62, 1999.

[8] Md Haque, Suraiya Pervin and others, "Literature Review of Automatic Single Document Text Summarization Using NLP," International Journal of Innovation and Applied Studies, vol.3, no.3, pp. 857-865, 2013.

[9] Ladda Suanmali, Mohammed Salem Binwahlan and Naomie Salim, "Sentence features fusion for text summarization using fuzzy logic," Ninth International.

[10] Li Chengcheng, "Automatic Text Summarization Based On Rhetorical Structure Theory," International Conference on Computer Application and System Modeling (ICCASM), vol. 13, pp. 595-598, October 2010.

[11] Atul Kumar et al 2021, "A Comparative Analysis of Pre-Processing Time in Summary of Hindi Language using Stanza and Spacy" in IOP Conference Series: Materials Science and Engineering, volume 1110, number 1, pages 012019, Publisher: IOP Publishing, DOI: 10.1088/1757-899x/1110/1/012019.