Extractive Text Summarization using Meta-heuristic Approach

Doppalapudi Venkata Pavan Kumar[1,*,†], Srigadha Shreyas Raj[1,†], Pradeepika Verma[2,†] and Sukomal Pal[1,†]

[1]*Indian Institute of Technology (BHU), Varanasi, INDIA*
[2]*TIH, Indian Institute of Technology, Patna, INDIA*

Abstract

This paper describes a system for Indian Language Extractive Summarization which has been developed under the shared task ILSUM of FIRE 2022[1][2] Conference by SUMIL22 team. This system works by picking up sentences directly from the text using a ranking function to form the corresponding summary. In our approach, for each sentence, we first calculated various text features such as sentence position, sentence length, sentence similarity, frequent words, and sentence numbers. Then, these text features along with their optimized weights are used for ranking the sentences, and then the summary is generated by selecting top-ranked sentences. For optimizing the weights of the text features, we have used a population based meta-heuristic approach, Genetic Algorithm (GA). We submitted three runs and got an F-score of 0.3843 for ROUGE-1, 0.2584 for ROUGE-2, 0.1997 for ROUGE-3 and 0.2190 for ROUGE-4 in the best run.

Keywords

Extractive Text Summarization, Genetic Algorithm, Text Features, ROUGE

1. Introduction

Information has become the most important part of our life. People generally use a large number of sources from news articles to social media posts to know the information. Generating automatic summaries of larger texts will be useful for compressing the larger texts of information into smaller texts and also saves a lot of time. Our project focuses on automatic text summarization of news data. Text summarization is the creation of a shorter, precise, and more relevant summary of a larger text. Automatic summarization methods will be able to handle the ever-increasing amounts of internet text data, allowing users to find and consume important information more quickly.

Abstractive and extractive summarizations are the two broad types of automatic text summarization. Abstractive summarization methods attempt to create a summary by trying to interpret the text using advanced natural language methods to create a unique and more concise text of

Forum for Information Retrieval Evaluation, December 9-13, 2022, India
*Corresponding author.
†These authors contributed equally.
dvenkata.pavankumar.cse19@itbhu.ac.in (D. V. P. Kumar); srigadha.shreyasraj.cse19@itbhu.ac.in (S. S. Raj); pradeepikav.verma093@gmail.com (P. Verma); spal.cse@itbhu.ac.in (S. Pal)
https://ShreyasRaj4.github.io/ (S. S. Raj); https://cse-iitbhu.github.io/irlab/spal.html (S. Pal)

parts that may not show up in the original article, that symbolizes the most critical information from the original text, going to require paraphrasing of sentences and combining knowledge from the total article to generate summaries similar to what a human-written abstract does. An adequate abstractive summary is linguistically fluent and enwraps the core information of the input. On the other hand, extractive summarization methods create the summary by picking up sentences directly from the text, based on a ranking function to form a relevant summary. This method identifies essential sections of the text, cropping out and joining together parts of the content to produce an abridged version. We have used Extractive text summarization for English language in our approach.

The rest of the article is organized as follows. We review the state-of-the-art approaches related to extractive and abstractive document summarization in Section 2. Then, we presented our proposed approach in the Section 3. Section 4 discussed the results and analysis part of the work followed by conclusion and future scope in Section 5.

2. Literature Survey

A brief literature survey[3] of extractive and abstractive summarization has been discussed as follows.

Text Rank[4] is an unsupervised method to summarize text documents. It is based on graphs. Authors considered sentences of the document as the nodes of the graphs and similarity between the sentences is represented as the edges of the graphs. The sentences of the documents are transformed into the vectors using different techniques like bag-of-words, Tf-Idf, word2vec etc. Moreover, sentences are ranked using cosine similarity and the top sentences are extracted to form the summary. Next, Erkan and Radev[5] proposed a Graph-based summarization system known as LexRank. This system identifies the most central sentences in the cluster of sentences, which gives us sufficient information regarding the main topic of the cluster. The sentences of the articles are converted into vectors by Tf-Idf technique and thereafter sentence vectors are clustered in order to extract the summary of the article.

Further, a heuristic Approach for Telugu text summarization with improved sentence ranking had been proposed by Mamidal et al.[6]. In this work, extractive text summarization for Telugu languague had been done using optimized sentence ranking. This sentence scoring mechanism was based on the event and named entity scores. Scoring of sentences was done by applying statistical measures on events and named entities features. Next, Verma and Om[7] proposed a maximum coverage and relevancy with minimal redundancy based multi-document summarization system. Here, summarization is done by generating a single document from all the documents and then scoring each sentence based on different text features and their optimal weights assigned by using Shark Smell Optimization (SSO). Moreover, Verma et al.[8][9] also proposed some other effective methods for document summarization.

3. Text Summarization using Proposed approach

In the proposed work, the task of summarization has been done into four stages that are Pre-processing of text input, Text features analysis, optimal weight assignment of the text feature,

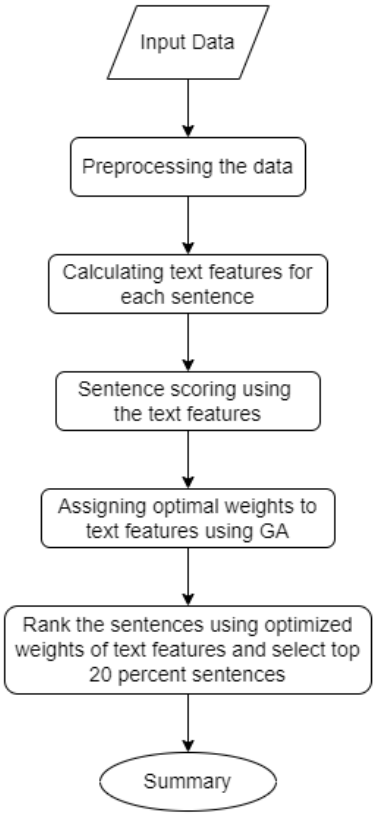

Figure 1: Flowchart of our approach

and summary generation. These steps are described as follows:

3.1. Preprocessing

Preprocessing is the first process to be done in our method. It covers removing of stop words, punctuations, unwanted data from the articles and doing sentence tokenization, word tokenization. Stop words are the words that are commonly occurred and do not have a specific significance meaning. Sentence tokenization divides the data into sentences and word tokenization divides the data into words. Stop words removal will remove all the stop words from the data after word tokenization. We have collected stop words for English language from Natural Language Tool Kit (NLTK) library. We have used NLTK library for word tokenization and for sentence tokenization, we used RegexpTokenizer from NLTK by writing our own regular expression.

3.2. Text feature's analysis

We have used various text features for scoring the sentences of each news articles. Let $s_{i,j}$ represent the j^{th} sentence of i^{th} news article. And t_x represent x^{th} text feature. Then $St_x\left(s_{i,j}\right)$ represent the score of x^{th} text feature for a sentence $s_{i,j}$. The text features used in our approach are as follows[10].

3.2.1. Sentence Position(t_1):

In a article, sentences present at the start and end are more relevant than those present in the middle of the article[11]. Since, the primary cause and the crucial details are present at the start and conclusions are present at the end of the articles. So, we assign the score to a sentence based on the equation:

$$St_1(s_{i,j}) = \frac{\left|\frac{L}{2} - j\right|}{\frac{L}{2}} \tag{1}$$

3.2.2. Sentence Length(t_2):

Sentences with longer lengths are more informative than sentences with fewer words. So, we assign less weight to sentences with shorter lengths. We calculate the score of each sentence by taking the ratio of the number of words in the sentence to the number of words in the longest length sentence.

$$St_2(s_{i,j}) = 1 - \frac{|AL(s_i) - |s_{i,j}||}{max(|s_{i,j}|)} \tag{2}$$

3.2.3. Sentence Similarity(t_4):

Sentences which are like other sentences and have more common information are considered more important. We calculate this feature score by taking the ratio of the intersection of words in two sentences and the length of comparing sentence[12].

$$st_4(s_{i,j}) = \frac{\sum_{j'=1,j\neq j'}^{|s_i|}(sim(s_{i,j}, s_{i,j'}))}{|s_i|} \tag{3}$$

3.2.4. Sentences with Frequent words(t_5):

Words which are more frequent in the text are more important. Hence, the sentences containing the frequent are also given more weightage than other sentences[7]. For our approach, we have taken top 20% words having maximum frequency as frequent words.

$$St_5(s_{i,j}) = St_5(s_{i,j}) = \frac{\sum_{fw\in s_{i,j}} count(gram_N)}{|fw|} \tag{4}$$

3.2.5. Sentences with Numerical Data(t_6):

Sentences containing numerical data are regarded as more informational as numbers describe more analytical information.

$$St_6(s_{i,j}) = \frac{\sum_{nd \in s_{i,j}} count(gram_N)}{|nd|} \tag{5}$$

3.3. Optimal weights assignment to text features

After computing the scores of each sentence for every text feature, a population based meta-heuristic approach Genetic Algorithm (GA) assigns the weights to these text features.

3.3.1. Overview of Genetic Algorithm

Genetic Algorithm is a model or abstraction of evolution through natural selection based on Charles Darwin's theory of natural selection. It is a form of evolution that occurs on computer[13]. It was proposed by John Holland and his collaborators in the 1960s and 1970s. Genetic Algorithms are adaptive and can be used to solve both constraint and non-constrained, search and optimization problems[14]. The genetic algorithm is based on the population's chromosome's genetic structure and behaviour. It will be helpful while working with large and complex datasets. Genetic algorithms are built on the following principles.

- Each chromosome represents a potential solution. As a result, the population is made up of collection of chromosomes.
- Each chromosome in the population is assigned a fitness value. As a result, higher fitness the better the solution is.
- The best chromosomes out of the existing chromosomes in the population are used for reproducing the following generation offsprings.
- Because of crossover, the offspring created will inherit characteristics from both parents. A minor change in the structure of a chromosome will be called mutation.

The main advantages of GA over general optimization algorithms are, it supports multiple objective optimization, it is good for noisy environments and has the ability to deal with complex problems and parallelism. The basic steps present in GA are initialization, fitness evaluation, selection, crossover, mutation, and termination. The fitness function varies with respect to the problem. The steps to obtain the optimized weights of the text features are as follows (we consider our method as a maximization problem)

3.3.2. Initialization of population

First, we generate random weights in (0,1) as the initial weights for the text features. These are generated in the form of solutions $A = \{a_1, a_2, a_3, ..., a_n\}$ [7] , where "n" is the number of solutions in each generation and each solution a_q can be further represented as $a_q = \{a_{q,1}, a_{q,2}, ..., a_{q,N}\}$ [7], where N = number of dimensions for each solution. Each element of a solution a_q can be represented as $a_{q,x}$, where $q = \{1, 2, ..., n\}$ and $x = \{1, 2, ..., N\}$.[7]
We have used the python library 'numpy.random' to generate these initial set of population.

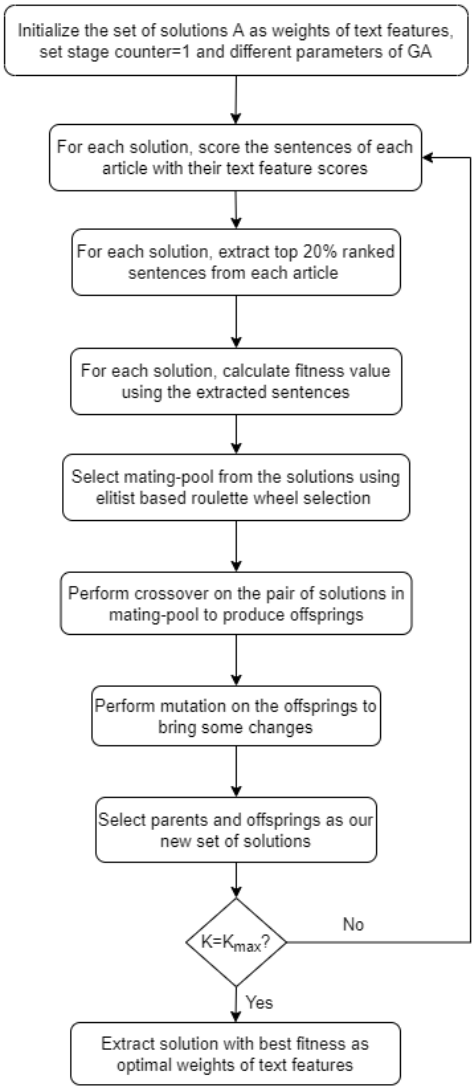

Figure 2: Flowchart for assigning optimal weights to text features

3.3.3. Fitness evaluation

Now, we have to evaluate the fitness of each solution. So, for calculating the fitness of each solution, we have to rank the sentences in each article, with each element of the solution as the weight of the corresponding text feature[7]. After ranking the sentences, select the top twenty percent ranked sentences in each article as the generated summary. Now, compare the generated summary with reference summary for each article using a fitness function and the sum of results obtained is the fitness value of the given solution. We have also optimized the fitness evaluation by passing the fitness values of previous generation to the fitness function, as it saves time by not calculating the fitness of the solutions passed from previous generation to this generation.

Sentence Ranking: For scoring the sentences of an i^{th} article, having feature's scores of St_p with a solution a_q, we use following equation.

$$score(s_{i,j}) = \sum_{x=1}^{N} a_{q,x} \times (St_x(s_{i,j})) \tag{6}$$

Where, $a_{q,x}$ represents the x^{th} dimension in the q^{th} solution. $St_x(s_{i,j})$ represents the x^{th} feature score for the j^{th}sentence in the i^{th} article.
After ranking the sentences, select the top twenty percent ranked sentences in each article as the generated summary. We will extract the sentences for each article for every solution. Now, these extracted sentences are compared with the use of a fitness function.

Fitness Function: Here, we use ROUGE metrics ROUGE-1 and ROUGE-2 for calculating the similarity between the generated summary and the reference summary. For calculating the fitness of a solution, we have to calculate the sum of similarity between the generated summary and the reference summary for every article.

$$fitness = \sum_{i=1}^{e} \frac{ROUGE1_i\,['f'] + ROUGE2_i\,['f']}{2} \tag{7}$$

Where, e is the total number of articles present and $ROUGE1_i\,['f']$ is the F1-score of the ROUGE-1 of i^{th} article.

3.3.4. Selection

After calculating the fitness values of each solution, now we have to select the parents from the set of solutions to produce the offsprings. We have to select half of the solutions for mating-pool. For selecting the parents, we have used elitist based roulette wheel selection. First, it selects the top 25 percent of the solutions as parents based on their fitness values. Then, it selects another 25 percent of the solutions from the remaining 75 percent solutions with the use of roulette wheel selection.

3.3.5. Crossover

Crossover, which is also known as recombination is used for combining the information of two solutions to produce a new offspring. There are many different types of crossovers available. But we have used simple single-point crossover. First, we select two parents from mating-pool and select a crossover point. A new offspring is produced by taking data before crossover point from first parent and combining it with data after crossover point from second parent.[15]

3.3.6. Mutation

Mutation is performed on the newly created offsprings to bring some changes in them. It is used to bring diversity from one generation of solutions to the next generation solutions. We have

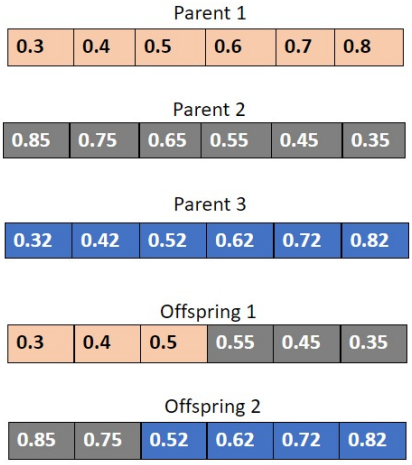

Figure 3: crossover

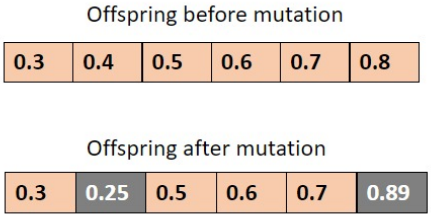

Figure 4: mutation

applied mutation on two different points for each solution in the offsprings. For each offspring, we select a point in first half of the offspring and select a random value in (-0.25,0.25) and add to the data at the selected point. Now, again for each offspring, we select a point in second half of the offspring and select a random value in (-0.25,0.25) and add to the data at the selected point. After performing mutation, all the new offsprings and the parents in the mating-pool will be combined to form the solutions for next generation.

3.3.7. Termination

Once the number of predefined iterations are completed, the best solution having the highest fitness value will be selected as the final weights of the text features.

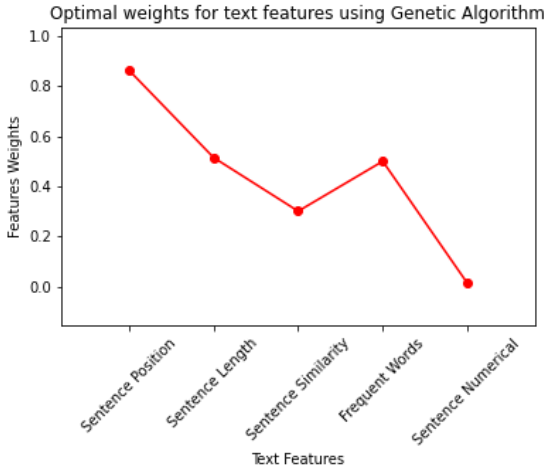

Figure 5: Optimal weights of text features using GA

3.4. Summary generation

Once the optimal weights for the text features are obtained with the use of genetic algorithm, we generate the final summary. For producing the summary, we first score the sentences in each article with the optimized weights of the features. If the GA gives the optimal weights for text features at solution q = O, then the score of $s_{i,j}$ is

$$score(s_{i,j}) = \sum_{q=O,x=1}^{N} a_{q,x} \times (St_x(s_{i,j})) \tag{8}$$

After ranking the sentences, the top twenty percent ranked sentences in each article are extracted as our summary.

4. Results and Analysis

In the English dataset, for each sentence in each of the articles, we calculate the values of all the text features. Then, we initialize 20 random individual population as the weights of the text features. Now, the GA is run for 200 generations. The optimized weights obtained using GA for the above described text features (in Section 3.3) are 0.86276778, 0.514368, 0.3003737, 0.49962974 and 0.01303291 respectively, which are shown in Figure-5.
The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scoring algorithm calculates the similarity between a text and a collection of reference articles. It determines the quality of the generated text. Rouge Score is the official metric for the ILSUM track.

By using these weights, we calculated the weighted average of the text feature scores of each sentence. We took the top twenty percentage ranked sentences in each article as the summary generated. The f-measure of ROUGE-1, ROUGE-2 and ROUGE-4 values of the summary generated and our rank for the English test dataset is shown in table-1. The detailed evaluation

Table 1
Evaluation results on test data and rank list for English Text Summarization

Team Name	ROUGE-I	ROUGE-II	ROUGE-IV	Rank
IIIT-H	0.5583	0.4458	0.4180	1/10
SUMIL22	0.3844	0.2584	0.2190	6/10

Table 2
Detailed evaluation results of test data for English Text Summarization using our approach

Evaluation Metric	F1-Score	Precision	Recall
ROUGE-I	0.3843673	0.3317808	0.5327141
ROUGE-II	0.2584236	0.2238869	0.359915
ROUGE-III	0.2313431	0.1997904	0.3252433
ROUGE-IV	0.2190241	0.1887224	0.3111874

results of English test dataset containing F1-Score, Precision and Recall for ROUGE-1, ROUGE-2, ROUGE-3 and ROUGE-4 is shown in table-2. From the obtained evaluation results, it can be understood that our technique does not give great results as expected at least upto 50% F1-Score. This is mainly because we produce 20% of the sentences as our summary but the actual summary can be any number of sentences and also during the sentence tokenization, we used a regular expression which reduced the length of the sentences than expected as compared to expected sentences.

5. Conclusion

This work reports for the shared task of the Automatic Text Summarization for English Language Text -FIRE 2022. We have experimented with several types of algorithms for the shared task, including abstractive text summarization like BERT, BART, etc. For extractive text summarization, we have performed sentence ranking using text features. The text features are weighted using genetic algorithm, which gave us the better results. However the scores can be improved further by using slightly tweaking the genetic algorithm, by training for many more generations and by increasing the text features. In future, we can use this approach to summarize the documents written in other Indian languages such as Hindi, Bengali, Tamil, Telugu etc.

Acknowledgement

This work is supported by TIH, Indian Institute of Technology, Patna and in lined with it's theme.

References

[1] S. Satapara, B. Modha, S. Modha, P. Mehta, Findings of the first shared task on indian language summarization (ilsum): Approaches, challenges and the path ahead, in: Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, Kolkata, India, December 9-13, 2022, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[2] S. Satapara, B. Modha, S. Modha, P. Mehta, Fire 2022 ilsum track: Indian language summarization, in: Proceedings of the 14th Forum for Information Retrieval Evaluation, ACM, 2022.

[3] P. Verma, S. Pal, H. Om, A comparative analysis on hindi and english extractive text summarization, ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 18 (2019) 1–39.

[4] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[5] G. Erkan, D. R. Radev, LexRank: Graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research 22 (2004) 457–479. URL: https://doi.org/10.1613%2Fjair.1523. doi:10.1613/jair.1523.

[6] K. K. Mamidala, et al., A heuristic approach for telugu text summarization with improved sentence ranking, Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12 (2021) 4238–4243.

[7] P. Verma, H. Om, Mcrmr: Maximum coverage and relevancy with minimal redundancy based multi-document summarization, Expert Systems with Applications 120 (2019) 43–56.

[8] P. Verma, A. Verma, S. Pal, An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms, Applied Soft Computing 120 (2022) 108670.

[9] P. Verma, A. Verma, S. Pal, A fusion of variants of sentence scoring methods and collaborative word rankings for document summarization, Expert Systems (2022) e12960.

[10] P. Verma, H. Om, A novel approach for text summarization using optimal combination of sentence scoring methods, Sādhanā 44 (2019) 1–15.

[11] M. A. Fattah, F. Ren, Automatic text summarization, World Academy of Science, Engineering and Technology 37 (2008) 192.

[12] P. Achananuparp, X. Hu, X. Shen, The evaluation of sentence similarity measures, in: International Conference on data warehousing and knowledge discovery, Springer, 2008, pp. 305–316.

[13] S. Forrest, Genetic algorithms: principles of natural selection applied to computation, Science 261 (1993) 872–878.

[14] D. Beasley, D. R. Bull, R. R. Martin, An overview of genetic algorithms: Part 1, fundamentals, University computing 15 (1993) 56–69.

[15] A. J. Umbarkar, P. D. Sheth, Crossover operators in genetic algorithms: a review., ICTACT journal on soft computing 6 (2015).