electronics

MDPI

Article

Extractive Article Summarization Using Integrated TextRank and BM25+ Algorithm

Vaibhav Gulati [1], Deepika Kumar [1], Daniela Elena Popescu [2] and Jude D. Hemanth [3,*]

[1] Department of Computer Science and Engineering, Bharati Vidyapeeth's College of Engineering, New Delhi 110063, India
[2] Faculty of Electrical Engineering and Information Technology, University of Oradea, 410087 Oradea, Romania
[3] Department of Electronics & Communication Engineering, Karunya Institute of Technology and Sciences, Coimbatore 641114, India
* Correspondence: judehemanth@karunya.edu

Abstract: The quantity of textual data on the internet is growing exponentially, and it is very tough task to obtain important and relevant information from it. An efficient and effective method is required that provides a concise summary of an article. This can be achieved by the usage of automatic text summarization. In this research, the authors suggested an efficient approach for text summarization where an extractive summary is generated from an article. The methodology was modified by integrating a normalized similarity matrix of both BM25+ and conventional TextRank algorithm, which resulted in the improvised results. A graph is generated by taking the sentences in the article as nodes and edge weights as the similarity score between two sentences. The maximum rank nodes are selected, and the summary is extracted. Empirical evaluation of the proposed methodology was analyzed and compared with baseline methods viz. the conventional TextRank algorithm, term frequency–inverse document frequency (TF–IDF) cosine, longest common consequence (LCS), and BM25+ by taking precision, recall, and F1 score as evaluation criteria. ROUGE-1, ROUGE-2, and ROUGE-L scores were calculated for all the methods. The outcomes demonstrate that the proposed method can efficiently summarize any article irrespective of the category it belongs to.

Keywords: article; text; summarization; extractive; graph-based; BM25; ROUGE; TextRank

Citation: Gulati, V.; Kumar, D.; Popescu, D.E.; Hemanth, J.D. Extractive Article Summarization Using Integrated TextRank and BM25+ Algorithm. *Electronics* **2023**, *12*, 372. https://doi.org/10.3390/electronics12020372

Academic Editor: Dah-Jye Lee

Received: 21 October 2022
Revised: 3 January 2023
Accepted: 9 January 2023
Published: 11 January 2023

1. Introduction

In the present age, the quantity of data on the Internet is rising exponentially. A huge quantity of textual data is uploaded on the Internet every single day. This explosion of textual data has made it impractical for us to go through all the data available on a particular topic and extract essential information [1]. Hence, there is an urgent need for technical innovations that can automatically parse through and filter huge bundles of textual data, thereby extracting the essence of the text [2]. One way to achieve this is via the usage of automatic text summarizers (ATSs). The ATS systems follow the process of text summarization, i.e., the process of condensing a large quantity of data and extracting the most important parts from the original text [3] and providing them to the users. A suitable text summarization system allows the user to obtain the desired data (most informative summary) from the text without having to read the entire document [4].

Automatic data summarization is a process of highlighting and extracting information and summary from an article or text document by using a computer program in such a manner that most essential components of the original text document are included. It is a part of machine learning, data mining, and natural language processing. In general, text synopsis methodologies are grouped into two categories: extractive and abstractive summarization [5,6]; the former delivers the summary from expressions or sentences in the article or text document, and the latter communicates the idea in the source record utilizing various words. It has been seen that extractive methods sometimes work in a way that

is better than the abstractive ones, most likely because extractive method does not need common language ages or semantic portrayals. Hence, the extractive approach outweighs the abstractive approach in terms of simplicity and effectiveness, so our proposed algorithm is based on the extractive text summarization technique [7].

In the extractive text summarization technique, the ATS system iterates through the whole textual data and extracts the most informative parts from the textual corpus. In some other cases, the user might not wish to completely iterate through the whole article to obtain essential information. In such scenarios, sentence extraction [8] is preferred over text summarization as it is a better and more efficient approach for that particular case. Hence, different technological methods incorporate different edge cases when the condensing of textual data is considered.

One notable and outstanding way to deal with extractive text summarization is the TextRank algorithm [9]. The TextRank algorithm is a graphical method in which pre-processed data are split, sentences/words act as vertices of a graph, and the weight on edges between them depends on the similarity between sentences/words. This similarity is calculated by creating a similarity matrix [10]. The major advantage of TextRank is that it is an unsupervised graph-based algorithm, i.e., it does not require any human summary (or training dataset) to decide the essential features of a textual document. Unsupervised algorithms require less manual data preparation and hence less overall time than similar supervised algorithms [11]. Along with being an unsupervised algorithm, TextRank is a language-independent ATS system that works on word occurrence. It calculates the importance of sentences/words and selects only the most informative sentences for the output summary.

The remainder of the paper is organized as follows: Section 2 discusses the literature in this area, and Section 3 elucidates the dataset used in this research along with the data collection procedure. Section 4 covers the methodology used for experimentation and the proposed algorithm for the research. Section 5 explains the results obtained by the proposed model and the comparison between the suggested model and other existing state-of-the-art models. The discussion is also included in this section. Lastly, Section 6 concludes the work and is followed by the references section.

2. Literature Review

The field of computerized synopsis has pulled interest since the late 1950s. Creating a summary is an inconvenient assignment for a domain master who has a corpus of information, but it is a more challenging task for machines. The machine should be able to process natural language and create a human-reasonable rundown, notwithstanding the foundation information field [12]. A major part of past work in the literature has been extractive, which encompasses recognizing key sentences or entries in the source record and replicating them as a summary. Luhn et al. [13] proposed rundowns dependent on term recurrence to recognize the significance of a sentence in the archive. The outcomes showed the possibility of consequently choosing sentences that demonstrate the overall topic, in the form of customary digests. Another approach which was a cohesion-based approach that considers durable relations between the idea inside the content, for example (antonyms, repetitions, synonyms, and so forth) utilizing lexical chains [14]. The outcomes showed that quality demonstrative rundowns are created. Machine-learning-based approaches mainly consist of supervised learning or semi-supervised and unsupervised learning approaches [15]. There have been many advancements in unsupervised learning approaches, especially in clustering and deep learning techniques in the past few years. A query-based document summarizer based on the OpenNLP tool and clustering technique was introduced in [16]. The extractive summarizer makes use of the WordNet dictionary and OpenNLP tool to extract syntactical connections between different parts of the input textual data. This cohesion is determined by the semantic parsing process. Thereafter, the K-mean clustering algorithm is applied to collect comprehensible sentences with each depending on the associativity degree as indicated by a cluster threshold value, which is

taken as input from the user. Lastly, the top five sentences (nodes) are extracted to shape the most informative summary. The outcomes showed the possibility of consequently choosing sentences that demonstrate the overall topic. Another algorithm known as the Louvain clustering algorithm was presented with the assistance of a dependency graph for single document summarization [17]. The algorithm builds a dependency graph for sentences, and the Louvain algorithm for word clustering is applied so that words inside each group are scored dependent on the reliance relations. The outcomes indicated that this work can accurately perform email document summarization tasks and can generate short event summaries. Another technique based on agglomerative clustering [18] was proposed. The algorithm implements cosine similarity and AGNES clustering models to rank and extract the most informative sentences. The main caveat of this method was the lack of coherence in the machine-generated summary. This deficiency is noticed in most of the extractive text summarization works as it requires a unified and intelligible collection of sentences, which is not generally found in extractive ATS systems. The most recently proposed technique is the recurrent neural network [19], which depends on the gated recurrent unit (GRU) neural network. It handles a single-record extractive rundown as a grouping characterization task in which a binary decision is registered for each sentence. The result demonstrated that it is preferred over or similar to the best-in-class deep learning models. Graph-based approaches are among the most well-known text summarization techniques. This methodology is based on the page ranking algorithm [20]. In this method, the sentences are spoken to by hubs in a weighted graph, with weighted edges deciding to utilize likenesses between hubs. TextRank and Lexrank are the other two graph-based approaches. In TextRank [9], the importance scores of hubs utilize voting-based weighting, while LexRank [21] is a cosine-transform-based weighting algorithm. Another methodology based on an affinity graph [22] was introduced for topic-based multi-document summarization. Summarization is carried out by obtaining the most elevated data wealth and oddity by calculating similarities in differentiating intra-document and inter-document connections between links. The results demonstrated excellent performance in topic-based multi-document summarization.

There have been many pieces of research related to graph-based approaches in the past few years. GRAPHSUM [23] was developed in 2013 as a graph-based summarizer that integrates the affiliation rules gained from an input document to find correlations between various terms during the summarization process. Some of the recent graph-based approaches used lexical association to determine the topic of the document. Chinnam and Ravinuthala [24] used lexical association to define the relationship among words in the document. Their approach used two vertex centrality methods, namely in-degree centrality and in-degree strength centrality, to implement a keyword extraction mechanism. This mechanism was used to extract the main words that built the subject of the document. Fuzzy-logic approaches can demonstrate sound judgment thinking in addition to dealing with uncertainty in an unsupervised manner. There have been many kinds of research based on the fuzzy-logic approach in the past few years. A technique was proposed that incorporated fuzzy logic with the swarm algorithm [25] to change feature scores and use them as commitments for the fuzzy inference system to accumulate the last scores. Fuzzy logic was integrated with latent semantic analysis for single document summarization [26]. In this approach, both methods generate a summary and finally intersect with one another to find the final one.

Extractive text summarization was utilized in the experiment of the present study, specifically the TextRank algorithm to automatically summarize all types of articles. The traditional TextRank model is based upon the PageRank algorithm that takes textual data as input instead of web pages [27]. It works by creating an undirected and weighted graph, considering sentences present in the article as nodes, taking the similarity present among any two sentences as the edge weight between those two nodes, and assigning an importance score to each node according to the number of connections present. The research revolves around the introduction of an altered form of this TextRank algorithm.

For the study, a dataset of 75 articles of 25 different categories was utilized and subjected to the extractive summarization process. The similarity function component of the original TextRank algorithm was tested, and a new similarity function is proposed for the extractive text summarization process. The output summary generated using the proposed algorithm in this research contains the top 'k' sentences from the input article in order of their lexical and mathematical importance.

3. Materials and Methods

The main aim of this study is to introduce a new and effective method for the extractive article summarization process. The starting point for any text-based research is the generation of a specific textual data corpus. This is considered the stepping stone for the study since the kind of textual data determines the domain and the technicalities of the research. This process demands great effort due to the availability of various types of textual data such as news articles, scientific research, and blogs. Out of these, the category currently untouched in the field of summarization is online blogs.

The main objective of an online blog is to provide a way to share someone's experience, thoughts, views, and knowledge on a specific topic with like-minded people available virtually in the form of an active blogging community. This allows people to connect and gain vast knowledge related to a specific topic. With the expansion of the virtual world and low-cost internet availability all around the world, there has been a large rise in the number of blogs available on the Internet. Thus, the authors decided to develop and research an intuitive way to automatically summarize online blogs. The following subsections elucidate the online blog data collection process along with the formation of a reference summary using the textual data corpus.

A Data Collection

Online blogs constitute the main textual data corpus for this research. Out of several open and freely available blogging and article-writing sites, the current most popular one is Medium's online publishing platform. Because of its global virtual presence and the variety of articles, the Medium platform can be considered an excellent textual data source for the extractive article summarization process. Hence, the dataset for this research was generated using the articles published on the Medium platform. The dataset was generated using 75 different articles that were extracted from the medium article site using several web scraping techniques [28]. The features of the dataset are depicted below in Table 1.

Table 1. Characteristics of the dataset.

Dataset Parameters	Value
Number of articles	75
Categories	25
No. of sentences per article (average)	89.5
No. of sentences per article (maximum)	229
No. of sentences per article (minimum)	30
Summary length (%)	20

B Dataset Description

The dataset for this study includes 75 different blogs published in the year 2018 on the Medium platform. These articles belonged to 25 different domains, including technology, science, business, nature, and entertainment. The length of these articles varies from the minimal 1003 words to 5185 words, with an average length of 100 sentences and 1865 words per article. Along with each raw article and its domain extracted from the Medium platform, the dataset also consists of an expert summary. The expert summary is a manually generated summary that acts as a reference summary in the evaluation process of ATS systems. For the dataset preparation, expert summaries were generated manually by the authors for all 75 articles. Each expert summary is strictly extractive and contains

some text from the original article that constitutes approximately 20% of the original article. The expert summary size was chosen to be around 20% in order to achieve significant coverage and quality [29]. The target summary size has to be chosen beforehand as the summarization accuracy score is directly proportional to the summary size. This is evident from the study of [29] in which generic single document and generic multi-document summarization tasks with target lengths of 50, 100, 200, and 400 words were performed. During the evaluation of the proposed summarization process, the reference summary is compared with the machine-generated summary.

4. Proposed Methodology

This section elucidates the proposed approach for the single-document-based extractive text summarization of online blogs. The study considers the TextRank algorithm as its baseline and proposes a modified version of it in this research. The study experiments with the similarity function of the TextRank algorithm and proposes a new way to generate the similarity matrix for a particular document. Figure 1 depicts the pictorial representation of the proposed model. The proposed model and its phases are described in detail in the following subsections.

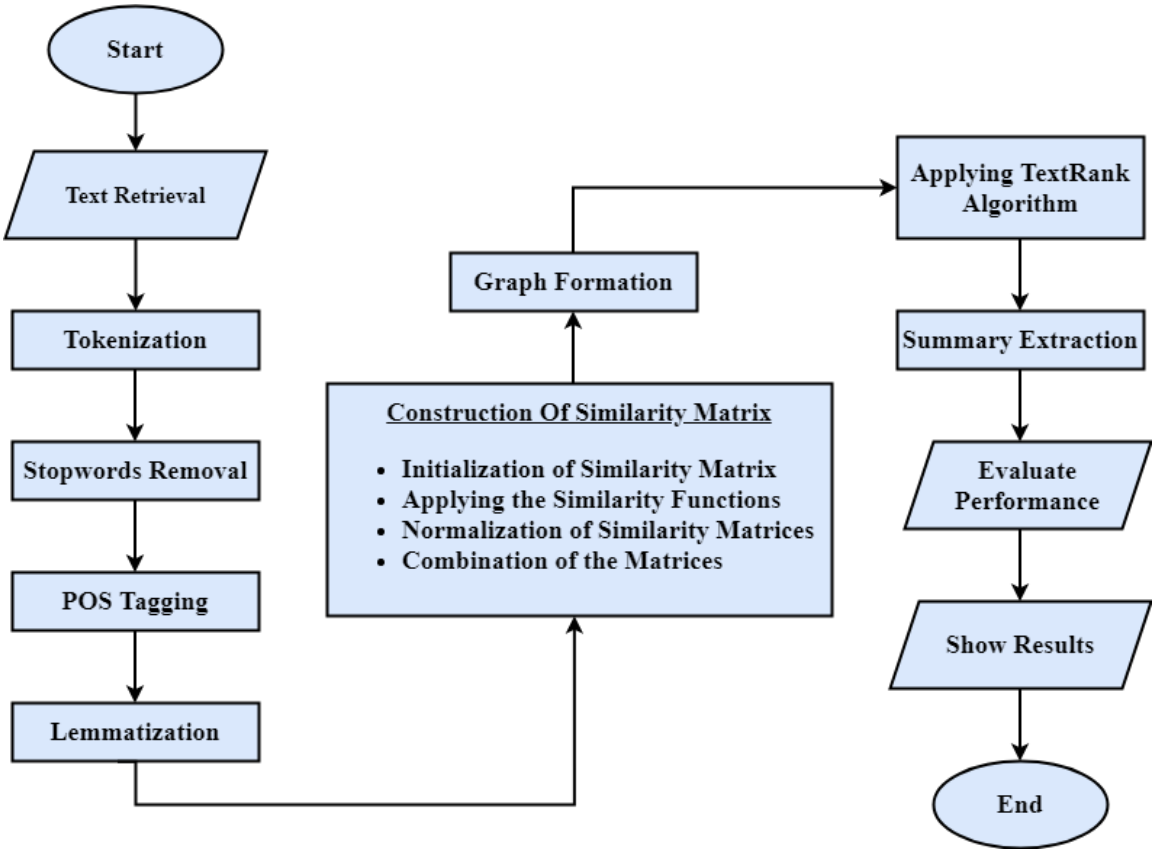

Figure 1. Proposed model overview.

The proposed methodology consists of three main phases: pre-processing, generation of the similarity matrix (processing stage), and finally the sentence ranking component. In the first stage, various text pre-processing techniques are applied to the input article to remove inconsistencies and normalize the text for the processing stage. In the next phase, the construction of a similarity matrix takes place. In the final stage, the TextRank algorithm is applied to rank the sentences of the input article as per their importance. The most informative sentences collectively form the output summary of the article.

4.1. Text Pre-Processing

The first phase of the proposed approach consists of several steps focused on the preparation of textual data for the extractive text summarization process. This phase transforms the textual data input into a clean and consistent format that can be fed to the proposed summarization model. It includes text pre-processing techniques such as stopwords removal, sentence and word tokenization, POS tagging, and lemmatization [30]. For the implementation of the pre-processing phase, Python's NLTK package [31] was used. The NLTK package is a collection of various text-processing modules that can be used to apply natural language processing pragmatically. The text pre-processing techniques are depicted in Figure 2 and are discussed in the following subsections:

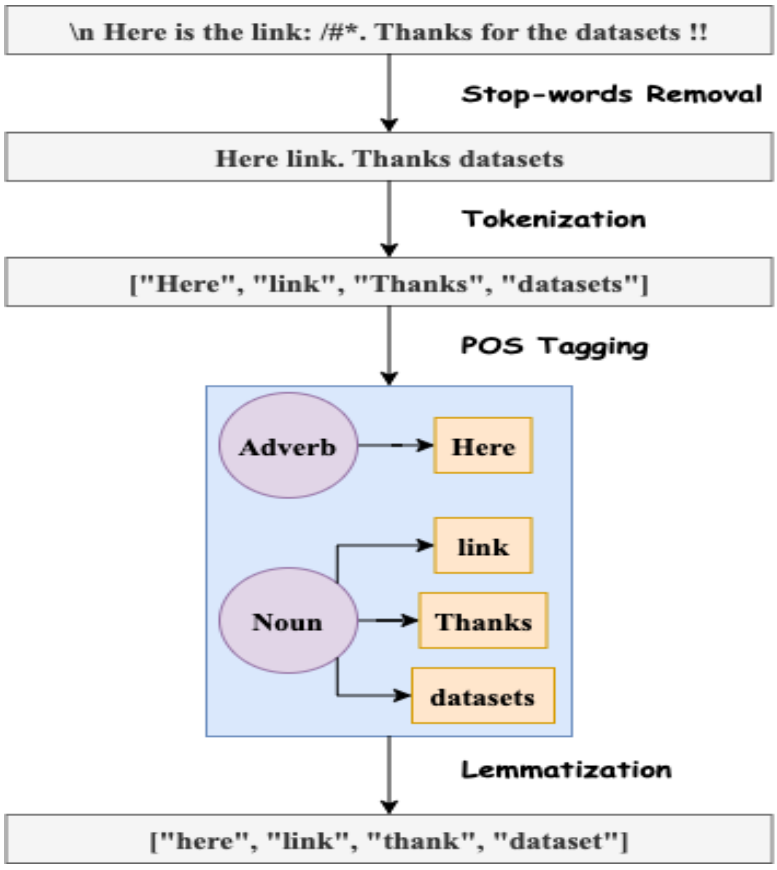

Figure 2. Text preprocessing phase.

4.1.1. Sentence Tokenization

Sentence tokenization is the procedure of dividing text into smaller units or tokens. As a result, we have to split the text corpus into a set of sentences. After which, the sentences are further tokenized to represent them by a collection of words. This is performed using the sent_tokenize(text) and word_tokenize(text) functions of the NLTK package.

4.1.2. Stopwords Removal

Stopwords is a term used to define those words and phrases that are used to join the meaningful words in a sentence. Hence, these irrelevant words can easily be removed from the text. Therefore, after dividing the corpus into sentences and words, we have to remove all the irrelevant words from all the sentences. To do this, we used the stopwords module from the corpus library of the NLTK package. By comparing if a particular word belongs to a category of stop words or not, all the stop words are removed from all the N sentences.

4.1.3. POS Tagging

Parts-of-speech (POS) is an essential component of a language's grammar that determines the meaning and the behavior of a word which directly affects the meaning of a particular sentence. Therefore, after removing all the stop words, we have to determine the type of words we are dealing with in a particular sentence. Hence, we used the pos_tag() function from the NLTK package, which helps us to identify the part of speech used in the sentence.

4.1.4. Lemmatization

In a sentence, a particular word might contain a prefix or a suffix along with the base or root word. The root word contains the most meaningful information about any word entity present in a sentence. Hence, identifying the root word is a major task in textual data pre-processing. In NLP, lemmatization is the process of identifying the root word of a particular word in a sentence. It is supported by POS tagging as it tells the type and behavior of a word which helps in identifying the possible suffixes and prefixes the word may contain along with the base. The root words present in all the N sentences are discovered, which means all the N sentences are lemmatized. To lemmatize all the sentences, the WordNetLemmatizer() from the NLTK package was used.

4.2. Formation of the Similarity Matrix

The TextRank algorithm works by modeling the text corpus as an undirected graph in which every sentence of the text corpus is illustrated by a particular node present in the graph, and the weight of the edge connecting two nodes is given by the similarity score between those two particular nodes (sentences) [32]. The higher the similarity score, the higher the importance of the edge in the TextRank graph. To obtain the similarity scores between each pair of sentences, a similarity function is required, which forms a square matrix of dimension NxN, where N is the number of sentences present in the text document. This similarity matrix permits the TextRank graph to be generated.

The construction of the similarity matrix consists of three stages. The first step starts by initializing the similarity matrix, i.e., generating a null matrix of dimensions NxN. After the initialization of the similarity matrix, the second step is to calculate the similarity scores using two different similarity functions, namely the original TextRank similarity function and the BM25+ similarity function. The second phase concludes with the generation of two similarity matrices, one generated by each of the similarity functions used. In the last stage, these two similarity matrices are normalized and combined to form the final similarity matrix from which the TextRank graph can be generated.

4.2.1. Initialization of Similarity Matrix

Firstly, we have to initialize a similarity matrix of dimensions (N $\times$ N) with all values in the matrix equal to zero. (Here, 'N' is the total number of sentences present in the text document.) This is carried out by providing the dimensions of the null matrix required as a parameter to the zeros() function, imported from the NumPy package [33].

4.2.2. Determining the Sentence Similarity Scores Using a Similarity Function

A similarity function is a mathematical model that can be used to calculate the similarity between two sentences. Various mathematical functions and their combinations can be used to find the similarity scores for the text corpus and can be easily integrated with the TextRank algorithm to find the summary of an article. In the proposed approach, two similarity functions are used:

- BM25+ Similarity Function

The similarity between each pair of sentences is calculated using BM25+ Similarity [34]: Firstly, a dictionary of all the words present in the article is created using Gensim Package [35]. Using this dictionary, a bag-of-words model [36] is created, which forms the

corpus for the implementation of the best matching 25 (BM25) algorithm. For each sentence Q containing words qi, the BM25+ score of corpus D is calculated using:

$$\text{score}(D,\ Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \left[\frac{f(q_i,\ D) \cdot (k_1 + 1)}{f(q_i,\ D) + \ k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} + \ \delta \right] \tag{1}$$

where, f (q_i, D) is q_i's term frequency in corpus D, |D| is the number of words in the corpus, and avgdl is the average number of words in a sentence in the corpus. Here, k1 (= 1.5), b (= 0.75), and δ (= 0.25) are constants, the values of which are predefined and extracted from [37]. Here, IDF (q_i) is the inverse document frequency weight of the word q_i.

- Original TextRank Similarity Function.

The traditional TextRank algorithm uses similarity measure as a relation between two sentences and focuses on the common content possessed by them, i.e., the sentence pair similarity simply calculates the total number of common lexical tokens present divided by the sum of the logarithm of the length of each sentence. Taking a logarithmic value avoids the unnecessary influence of long sentences [38]. The original TextRank Similarity Function is given by:

$$\text{Sim}\ (S_i,\ S_j) = \frac{|\{\ w_k |\ w_k\ \in\ S_i\ \&\ w_k\ \in\ S_j\ \}|}{\log(|S_i\ |) + \log(|S_j\ |)} \tag{2}$$

Here, S_i, and S_j are two sentences represented as $S_i = w_1^i,\ w_2^i,\ w_3^i, \ldots,\ w_n^i$. The original TextRank similarity function is based on the lexical analysis of the text, i.e., it determines the similarity between a pair of sentences based on their lexical features, whereas the BM25+ similarity function is based on the semantic analysis of the article as it is based on the occurrence of frequent words in a sentence pair. Hence, a combination of both the lexical as well as semantic features of the text is used as the proposed similarity function.

4.2.3. Normalization and Combination

After calculating the BM25+ similarity score and conventional TextRank similarity score for each sentence pair, the scores are stored in two different similarity matrices. These matrices are then normalized by dividing the similarity matrices with the maximum similarity score present in the particular matrix. It limits the similarity scores between the [0, 1] range. Let the normalized matrices be Sim_A for BM25+ and Sim_B for the conventional TextRank algorithm. These two normalized similarity matrices, i.e., Sim_A and Sim_B, are combined using the matrix addition operation to generate the final similarity matrix:

$$\text{Similarity\_Matrix} = \text{Sim\_A} + \text{Sim\_B} \tag{3}$$

The above equation generates a normalized similarity matrix which can be fed directly to the TextRank algorithm to generate the extractive summary of the input article.

4.3. Summary Extraction Using TextRank

After the formation of the similarity matrix, the next step is the modeling of the textual data into a weighted undirected graph formed by representing the sentences of textual data as nodes and taking the similarity scores between the pair of sentences as edge weights. The formed graph is given as an input to the PageRank algorithm, which is a graph-based technique that iteratively ranks the nodes of a graph based on the connectivity of a node in the graph. Traditionally, the PageRank algorithm was designed to rank web pages by taking a directed graph as the input. PageRank is applied to weighted undirected graphs by considering edge weights. Then, a score is assigned to each sentence by taking into consideration how deeply a node is connected to other nodes, i.e., a densely connected node will have a higher PageRank score, whereas a sparsely connected node will have a low importance score. Finally, the sentences with high PageRank scores are selected to form the summary of the corresponding text corpus.

4.3.1. Graph Formation

The text corpus is modeled into a graph G (V, E, W); where 'V' is the set of vertices of the graph, 'E' is the set of edges, and 'W' is the set of edge weights. It is formed by representing each sentence present in the text corpus as a node/vertex in the graph with edges between two particular nodes corresponding to the similarity score between the two sentences. The graph of the text corpus was generated using the from_numpy_array() function of the NetworkX package [39]. The similarity matrix is provided as an argument to the from_numpy_array() function, which generates the graph by considering the similarity matrix as the adjacency matrix for the graph.

4.3.2. PageRank Algorithm

The PageRank algorithm is a graph-based algorithm formulated in 1996 by Larry Page and Sergey Brin, which ranks webpages according to their importance based on the search performed using a web search engine. In this algorithm, a directed graph is created by taking the web pages as nodes and the number of links defined in a webpage as branches. Then, each node is given an importance score value depending upon the number of nodes it is connected to. Accordingly, a higher importance score is assigned to a more densely connected node, whereas a low importance score is assigned to a sparsely connected node. The PageRank algorithm [40] is explained as follows:

Assume webpage A has T_1, T_2,..., and T_n webpages that contain links pointing to it. Let C(A) define the links going out of page A and the parameter 'd' be the damping factor. The damping factor helps in reducing the influence of other web pages, thereby damping the total vote. Generally, the value of 'd' is set to 0.85 [40]. Then, the PageRank of A is shown by:

$$PR(A) = (1 - d) + d \cdot \left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \cdots + \frac{PR(T_n)}{C(T_n)} \right) \quad (4)$$

The philosophy of the PageRank algorithm laid the foundation of Google's web-search engine [20], along with several innovations in various fields of science and technology. One of these fields is textual data mining, for which the TextRank algorithm [9] was introduced.

4.3.3. TextRank Algorithm

The TextRank algorithm is a graph-based ranking algorithm inspired by the PageRank algorithm that is used in the extractive and single-document text summarization process by ranking sentences using an undirected graph instead of web pages. In TextRank, the graph is generated using sentences as nodes and the similarity between sentences as the weight of branches. Hence, in TextRank, an undirected weighted graph is formed which differs from the original directed graph formed in the PageRank algorithm.

4.3.4. Summary Extraction

In this process, the nodes of the graph are sorted in descending order of their ranks, and the top k sentences are selected by the system. Here, the constant 'k' constitutes the length of the summary, which is taken as 20% of the length of the article. Hence, the constant 'k' is found using:

$$k = \text{floor}\left(\frac{20 * N}{100} \right) \quad (5)$$

Here, N is the number of sentences present in the text corpus.

4.4. Notable TextRank Similarity Matrix Models

Numerous studies and modifications of the summarization process have been carried out, particularly in the processing phase, i.e., in the similarity function of the TextRank algorithm. Some of the notable variations include the TF-IDF cosine method, longest common subsequence (LCS) method, and the BM25+ similarity calculation method. Out

of all these methods, the latest one, the BM25+ similarity calculation method, has shown major improvement in the summarization process. Figure 3 depicts the deviations in the process of the construction of the similarity matrix.

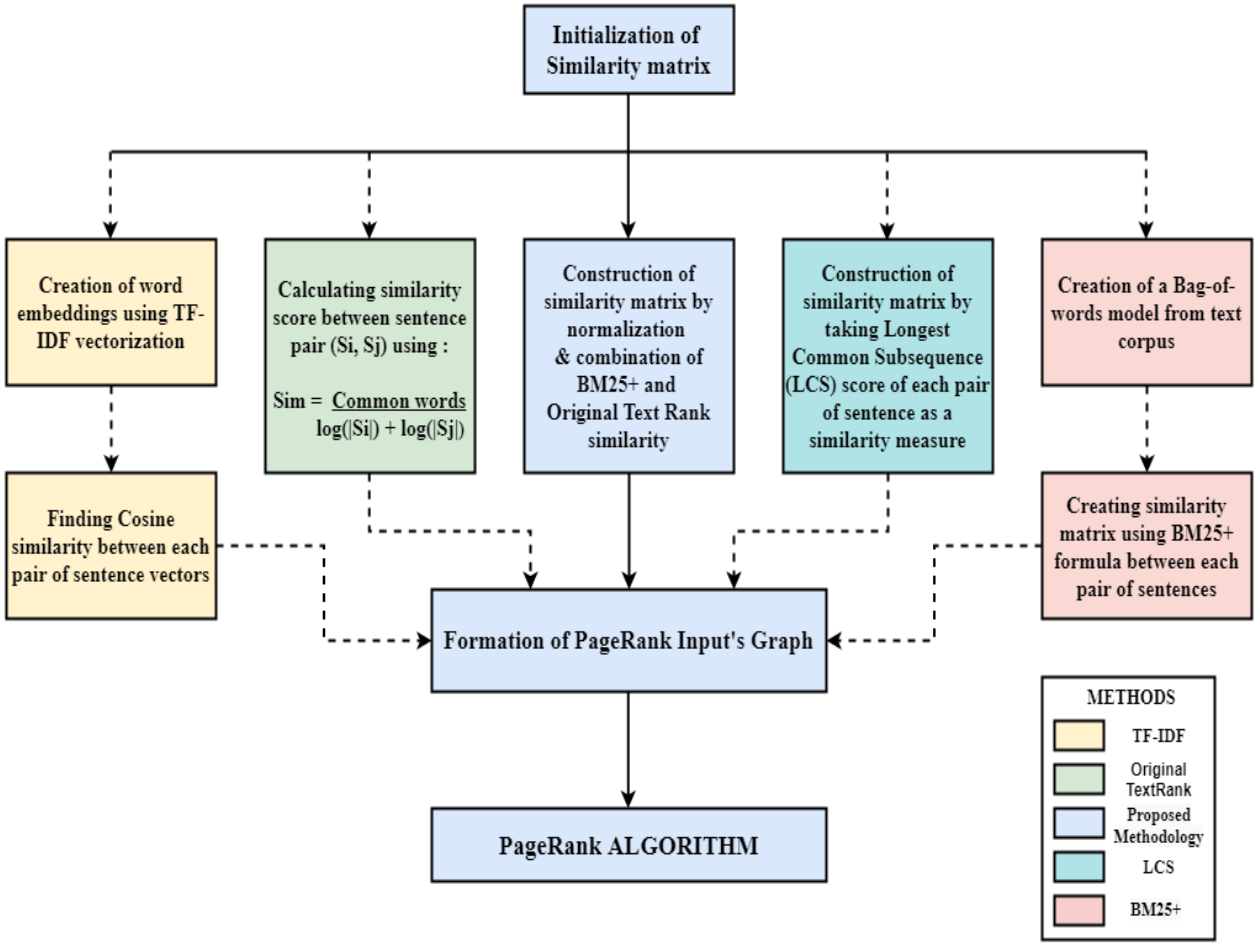

Figure 3. Variations in similarity function of TextRank algorithm.

The above figure covers the various modifications applied to the similarity function of the TextRank algorithm to achieve an efficient extractive text summarization process. The main objective of the similarity function is to generate a metric that could act as a measure of similarity between a pair of sentences of a document. This measure has been generated in various ways in the literature. The most notable ones are discussed in the following subsections:

4.4.1. Original TextRank Similarity Function

The traditional process of TextRank implements the similarity measure as a relation between two sentences focusing on the common content both of them possess, i.e., the sentence pair similarity is simply calculated by counting the total number of common lexical tokens present in both of them divided by the sum of the logarithm of the length of each sentence, to avoid the unnecessary influence of long sentences.

4.4.2. TF-IDF Cosine Similarity Function

Term frequency (TF)–inverse document frequency (IDF), i.e., TF–IDF, is a way to determine the value of a specific word in a text document [41]. Its value is calculated by its repetition in a text document or an article. Using this, the sentences are vectorized, and cosine similarity is calculated. To calculate the cosine similarity, two-sentence vectors that are TF-IDF vectors are required, and their dot product acts as the similarity between them.

After calculating the similarity score between each sentence pair, the similarity matrix is constructed, the contents of which act as branch weights in the Graph formed during the TextRank summarization process.

4.4.3. TF-ISF Cosine Similarity Function

In this method, the sentences are assigned a TF-ISF score, and cosine similarity is calculated for each sentence pair, which is treated as a similarity matrix for the TextRank graph [42]. The first step is the calculation of term frequency (TF) and inverse sentence frequency (ISF). Here, the TF determines the number of occurrences of a particular word in the sentence. Similarly, the ISF determines how unique or rare a particular word is. ISF suggests that if a term is occurring less frequently in the whole text corpus, then it is valuable for the text document. Thus, both TF and ISF values are combined to form the TF-ISF score. The original cosine similarity only takes the TF of all the words present in a sentence. Moreover, in the case of cosine similarity, the vectors of the corresponding sentences should have a fixed length, whereas in the case of ISF-modified cosine similarity, we can consider different levels of importance for a particular word, and the length of the sentence vectors can be different as well.

4.4.4. LCS Similarity Function

In this method, the longest common subsequence (LCS) score between a sentence pair acts as branch weight (similarity measure) for each node (sentence) in the TextRank input graph. Here, LCS is the longest subsequence that is common to all the given strings, provided that the elements of the subsequence need not occupy adjacent positions within the original string.

4.4.5. BM25+ Similarity Function

The BM25+ similarity function is the latest improvement in the text summarization process that includes the implementation of a bag-of-words model of the text corpus, in which the BM25+ score acts as a similarity measure for each sentence pair. It is a variation of the TF-IDF model that is essentially a ranking function widely used as the state-of-the-art model for information retrieval tasks.

4.4.6. Karcı Summarization

Karcı summarization is an extractive summarization technique that uses Karcı entropy [43] for its implementation. The main advantage of the model is that it does not require any kind of training data. The Karcı summarization model contains two main stages. The first stage includes removing the missing words from the dataset with no distinctiveness (for example, pronouns). After that, text pre-processing is achieved with the help of the KUSH toolkit [43], which is used to perform certain kinds of normalizations. Highly synonymous words which are derived semantically from the common root word are checked so that they are not processed as different words during graph formation. Then, the common properties between word units are depicted in graphical and mathematical form. In the second stage, the Karcı entropy is applied, and different summaries are generated by changing the value of parameter 'α', i.e., the exponent coefficient of the Karcı entropy, while changing the exponent parameter that modifies the text summarization process as the Karcı entropy function determines the effect of sentences over one another in the total graph.

4.4.7. Word Mover's Distance

In this method, word2vec vector embeddings of words are used along with a bag-of-words model to find the minimum 'traveling distance' between two sentences in a meaningful way [44]. It is based on the assumption that similar words should have similar vectors. In simpler words, the word mover's distance (WMD) is calculated for each sentence pair, i.e., the most efficient way to 'move' the distribution of one sentence to the other. It is based on the concept that the embedded word vectors' distances are semantically

meaningful to some extent. A model can learn from local co-occurrences in sentences with semantically meaningful representations of words. It is free from hyperparameters and is a straightforward technique that uses word embeddings to treat text documents as a weighted point cloud of embedded words.

5. Results

A Evaluation Metrics

The similarity matrix construction methods were thoroughly analyzed and evaluated using the medium article dataset. To evaluate the performance of the proposed approach, we implemented different variations of the TextRank algorithm on the article dataset and used the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [45]. ROUGE-1, ROUGE-2, and ROUGE-L were used as performance metrics. The ROUGE metrics work by measuring the match rate of a group of tokens (words) between the automated model-generated summary and the expert summary. ROUGE-1 evaluates the intersection of individual words in a sentence pair, ROUGE-2 alludes to the intersection of word pairs (bigrams), and the ROUGE-L metric alludes to the longest matching sequence of words or LCS-based statistics.

Suppose P is the set of sentences of the reference summary, Q is the set of sentences of the machine-generated summary, and LCS-based F-score (F_{lcs}) specifies the similarity between P and Q, then ROUGE-L recall (R_{lcs}), precision (P_{lcs}), and F-score (F_{lcs}) [45] are calculated using:

$$R_{lcs} = \frac{LCS(P,Q)}{n} \tag{6}$$

$$P_{lcs} = \frac{LCS(P,Q)}{m} \tag{7}$$

$$F_{lcs} = \frac{\left(1+\beta^2\right) R_{lcs} P_{lcs}}{\left(R_{lcs} + \beta^2 P_{lcs}\right)} \tag{8}$$

Here, LCS (P, Q) designates the length of the LCS of P and Q, and $\beta = \frac{P_{lcs}}{R_{lcs}}$.

The article summarization is obtained using the proposed integrated TextRank and BM25+ algorithm. The analysis is based on the ROUGE toolkit result. Apart from the traditional TextRank summarization method, three more variations (Figure 3) were implemented and evaluated on the same article dataset along with the proposed method. The results of the proposed methodology were compared and analyzed with conventional TextRank, TF-IDF Cosine, LCS similarity, and BM25+ similarity. The detailed results are shown in Table 2.

Table 2. Comparison analysis of various methodologies using medium article dataset.

Model		ROUGE-1	ROUGE-2	ROUGE-L
Original TextRank				
	Recall	0.757743	0.652379	0.726952
	Precision	0.739719	0.637702	0.714006
	F-score	0.744394	0.641341	0.716935
TF-IDF Cosine Similarity				
	Recall	0.670118	0.546507	0.623722
	Precision	0.733197	0.598681	0.696064
	F-score	0.695501	0.567516	0.653939
LCS Similarity Function				
	Recall	0.733842	0.557659	0.696512
	Precision	0.579289	0.442102	0.524561
	F-score	0.644777	0.491148	0.596118

Table 2. *Cont.*

Model		ROUGE-1	ROUGE-2	ROUGE-L
BM25+ Similarity Function				
	Recall	**0.804444**	**0.699020**	**0.779963**
	Precision	0.709480	0.618667	0.682280
	F-score	0.750457	0.653352	0.724864
Proposed Approach				
	Recall	0.776544	0.674904	0.748617
	Precision	**0.739773**	**0.643668**	**0.715275**
	F-score	**0.753685**	**0.655442**	**0.728332**

The bold values show the highest scores among the numbers in the table.

The proposed model outperformed the past implemented models in terms of precision and F-score values. However, it reflects a different picture for recall. This improved performance of the proposed method is linked to the concept of a combined similarity matrix that combines the mathematic and semantic concepts of textual data. Figure 4 shows that the proposed method can efficiently summarize any article irrespective of the category it belongs to.

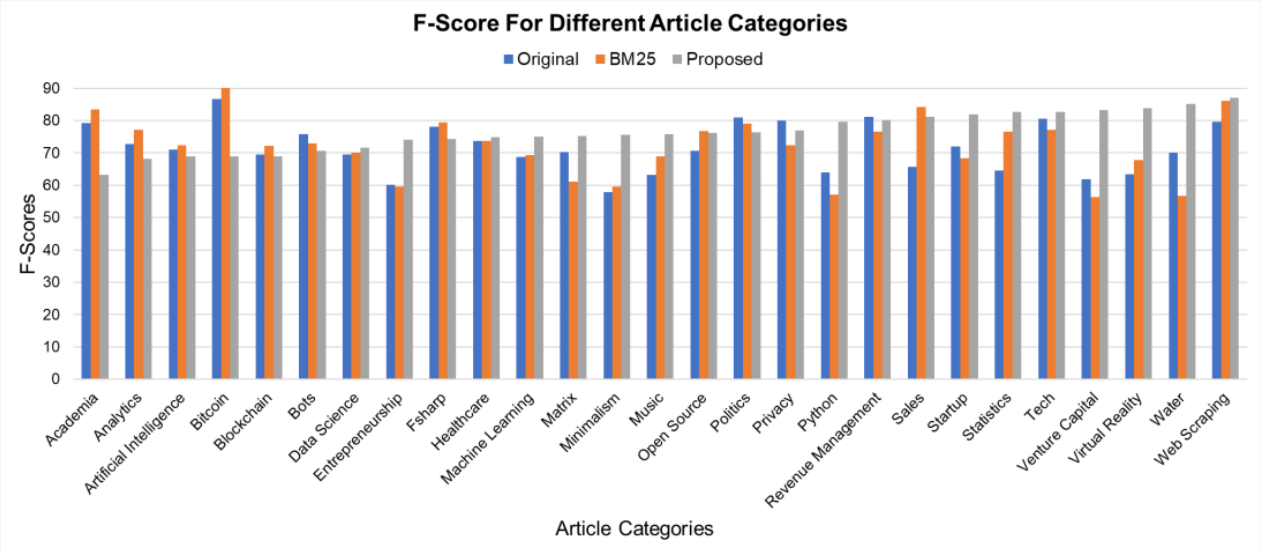

Figure 4. F-Score for different article categories.

It is evident from the above figure that the proposed integrated TextRank algorithm can be used to summarize various types of online blogs. It collectively outperforms its two baselines' models, i.e., the BM25+ model and the original TextRank models in the extractive summarization process.

B Comparison with state-of-the-art methods

The extractive summarization performance results obtained using the proposed methodology were compared with the results of various state-of-the-art single-document extractive summarization methods. These ATS system results were reported from the previous studies in the literature. The comparison of various ATS systems is discussed in Table 3.

Table 3. Comparison of the proposed model with various state-of-the-art methods based on average F-scores (ROUGE metric).

Source	Author	Methodology	F-Score * (in %)		
			ROUGE-1	ROUGE-2	ROUGE-L
[19]	Ramesh Nallapati, et al.	SummaRuNNer (two-layer RNN-based sequence classifier)	46.60	23.10	43.03
[43]	Cengiz Harka, Ali Karcı	Karcı entropy-based summarization	49.41	22.47	46.13
[46]	Wafaa S. El-Kassas, et al.	EdgeSumm (unsupervised graph-based framework)	53.80	28.58	49.79
[47]	Chirantana Mallick, et al.	Modified TextRank (inverse sentence frequency-cosine similarity)	73.53	67.33	67.65
[48]	Yang Liu	BERT-SUM (BERT with interval segment embeddings and inter-sentence transformer)	43.25	20.24	39.63
[49]	Kamal Al-Sabahi, et al.	HSSAS (hierarchical structured LSTM with self-attention mechanism)	52.10	24.50	48.80
[50]	Aishwarya Jadhav, et al.	SWAP-NET (Seq2Seq Model with switching mechanism)	41.60	18.30	37.70
Proposed Methodology		Integrated TextRank	75.37	65.54	72.83

* Direct comparison of the papers mentioned above is not fully equitable as these ATS systems differ along many dimensions. The experiment results vary due to the usage of different datasets or the length of the output summary. In the proposed model, the authors used an integrated approach for extractive text summarization, which was tested on an author-generated article dataset. This integrated approach is efficient and robust enough for a new textual dataset due to its unsupervised nature. This is verified and depicted in Table 3 as the proposed methodology achieves higher accuracy than that of all state-of-the-art systems.

C Discussion

In this study, an attempt was made to explain and implement extractive text summarization using TextRank and its variations. A modified similarity matrix was used in comparison with the original TextRank algorithm. This modified similarity matrix improved the accuracy of the summarization process as it resulted in more accurate weights on the edges of the TextRank graph. Among different variations of the similarity matrix implemented, the proposed algorithm performs better than the traditional TextRank and the BM25+ similarity model. To prove the efficiency of this method, ROUGE scores were calculated (ROUGE-1, ROUGE-2, and ROUGE-L). Overall, the proposed model achieved 0.73 averaged recall, 0.70 averaged precision, and 0.7125 averaged F-score. Compared to the original TextRank and the BM25+ similarity model of TextRank, the proposed algorithm uses a combination of normalized similarity matrices of the two methods which in turn improved the results (mean of F-score for all the ROUGE metrics) by 1.654% and 0.413%, respectively.

6. Conclusions

The proposed algorithm implements the best-of-both-worlds approach by using the combination of the new bag-of-words model and traditional similarity calculation, connecting new branches to the roots of the methodology tree. This work can further be extended by using more robust page rank algorithms to obtain better extractive summary generalization. There are other areas too where this work can be extended. This research does not consider the anaphora resolution problem, which can be resolved using the Mitkov Anaphora Resolution System (MARS) algorithm [51,52] to improve the accuracy of the computer-generated summary. This research also limits whether the generated summary may have repeated sentences, which can be removed by using a suitable clustering algorithm. These changes will improve the efficiency of our computer-generated summary.

Author Contributions: V.G., D.K., D.E.P. and J.D.H. participated in (a) conception and design and analysis and interpretation of the data and (b) drafting the article and revising it critically for important intellectual content. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available in the extractive-text-summarizer repository at https://github.com/gulvaibhav20/extractive-text-summarizer/tree/main/dataset. These data were derived from the following resources available in the public domain: https://medium.com/ and accessed on 5 July 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vlainic, M.; Preradovic, N.M. A comparative study of automatic text summarization system performance. In Proceedings of the 7th European Computing Conference (ECC 13), Dubrovnik, Croatie, 25–27 June 2013.
2. Lloret, E.; Palomar, M. Text summarisation in progress: A literature review. *Artif. Intell. Rev.* **2011**, *37*, 1–41. [CrossRef]
3. Gambhir, M.; Gupta, V. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.* **2016**, *47*, 1–66. [CrossRef]
4. Gong, Y.; Liu, X. Creating generic text summaries. In Proceedings of the Sixth International Conference on Document Analysis and Recognition, Seattle, WA, USA, 10–13 September 2001.
5. Chopra, S.; Auli, M.; Rush, A.M. Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016.
6. Bhargava, R.; Sharma, Y.; Sharma, G. Atssi: Abstractive text summarization using sentiment infusion. *Procedia Comput. Sci.* **2016**, *89*, 404–411. [CrossRef]
7. Kan, M.-Y.; McKeown, K.R.; Klavans, J.L. Applying natural language generation to indicative summarization. *arXiv* **2001**, arXiv:cs/0107019.
8. García-Hernández, R.A.; Montiel, R.; Ledeneva, Y.; Rendón, E.; Gelbukh, A.; Cruz, R. Text Summarization by Sentence Extraction Using Unsupervised Learning. In Proceedings of the Mexican International Conference on Artificial Intelligence, Atizapán de Zaragoza, Mexico, 27–31 October 2008; Springer: Berlin/Heidelberg, Germany, 2008.
9. Mihalcea, R.; Tarau, P. Textrank: Bringing order into text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004.
10. Zha, P.; Xu, X.; Zuo, M. An Efficient Improved Strategy for the PageRank Algorithm. In Proceedings of the 2011 International Conference on Management and Service Science, Bangkok, Thailand, 7–9 May 2011.
11. Moratanch, N.; Chitrakala, S. A survey on extractive text summarization. In Proceedings of the 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 10–11 January 2017.
12. Ghodratnama, S.; Beheshti, A.; Zakershahrak, M.; Sobhanmanesh, F. Extractive document summarization based on dynamic feature space mapping. *IEEE Access* **2020**, *8*, 139084–139095. [CrossRef]
13. Luhn, H.P. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.* **1957**, *1*, 309–317. [CrossRef]
14. Onwutalobi, A.C. Using Lexical Chains for Efficient Text Summarization. Available online: https://ssrn.com/abstract=3378072 (accessed on 16 May 2009).
15. Neto, J.L.; Freitas, A.A.; Kaestner, C.A.A. Automatic text summarization using a machine learning approach. In Proceedings of the Brazilian Symposium on Artificial Intelligence, Porto de Galinhas, Recife, Brazil, 11–14 November 2002; Springer: Berlin/Heidelberg, Germany, 2002.
16. Jain, H.J.; Bewoor, M.S.; Patil, S.H. Context sensitive text summarization using k means clustering algorithm. *Int. J. Soft Comput. Eng* **2012**, *2*, 301–304.
17. El-Kilany, A.; Saleh, I. Unsupervised document summarization using clusters of dependency graph nodes. In Proceedings of the 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, India, 27–29 November 2012.
18. Sharaff, A.; Shrawgi, H.; Arora, P.; Verma, A. Document Summarization by Agglomerative nested clustering approach. In Proceedings of the 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), New York, NY, USA, 2–3 December 2016.
19. Nallapati, R.; Zhai, F.; Zhou, B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
20. Brin, S.; Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **1998**, *30*, 107–117. [CrossRef]
21. Erkan, G.; Radev, D.R. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.* **2004**, *22*, 457–479. [CrossRef]

22. Wan, X.; Xiao, J. Towards a unified approach based on affinity graph to various multi-document summarizations. In Proceedings of the International Conference on Theory and Practice of Digital Libraries, Budapest, Hungary, 16–21 September 2007; Springer: Berlin/Heidelberg, Germany, 2007.
23. Baralis, E.; Cagliero, L.; Mahoto, N.; Fiori, A. GRAPHSUM: Discovering correlations among multiple terms for graph-based summarization. *Inf. Sci.* **2013**, *249*, 96–109. [CrossRef]
24. Ravinuthala, V.V.M.K.; Chinnam, S.R. A keyword extraction approach for single document extractive summarization based on topic centrality. *Int. J. Intell. Eng. Syst.* **2017**, *10*, 5. [CrossRef]
25. Binwahlan, M.S.; Salim, N.; Suanmali, L. Swarm based text summarization. In Proceedings of the 2009 International Association of Computer Science and Information Technology-Spring Conference, Singapore, 17–20 April 2009.
26. Babar, S.A.; Patil, P.D. Improving performance of text summarization. *Procedia Comput. Sci.* **2015**, *46*, 354–363. [CrossRef]
27. Wongchaisuwat, P. Automatic keyword extraction using textrank. In Proceedings of the 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), Tokyo, Japan, 26–29 April 2019.
28. Mitchell, R. *Web Scraping with Python: Collecting More Data from the Modern Web*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2018.
29. Nenkova, A. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, Pittsburgh, PA, USA, 9–13 July 2005.
30. Anandarajan, M.; Hill, C.; Nolan, T. *Text preprocessing. Practical Text Analytics*; Springer: Cham, Switzerland, 2019; pp. 45–59.
31. Loper, E.; Bird, S. Nltk: The natural language toolkit. *arXiv* **2002**, arXiv:cs/0205028.
32. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
33. Oliphant, T.E. *A Guide to NumPy*; Trelgol Publishing: Austin, TX, USA, 2006; Volume 1.
34. Robertson, S.; Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends®Inf. Retr.* **2009**, *3*, 333–389. [CrossRef]
35. Rehurek, R.; Sojka, P. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks, Valletta, Malta, 22 May 2010.
36. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [CrossRef]
37. Barrios, F.; López, F.; Argerich, L.; Wachenchauzer, R. Variations of the similarity function of textrank for automated summarization. *arXiv* **2016**, arXiv:1602.03606.
38. Lee, D.; Verma, R.; Das, A.; Mukherjee, A. Experiments in Extractive Summarization: Integer Linear Programming, Term/Sentence Scoring, and Title-driven Models. *arXiv* **2020**, arXiv:2008.00140.
39. Hagberg, A.; Swart, P.; S Chult, D. Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference, Pasadena, CA, USA, 19–24 August 2008; No. LA-UR-08-05495; LA-UR-08-5495. Los Alamos National Lab.(LANL): Los Alamos, NM, USA, 2008.
40. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*; Stanford InfoLab: Stanford, CA, USA, 1999.
41. Christian, H.; Agus, M.P.; Suhartono, D. Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF). *ComTech Comput. Math. Eng. Appl.* **2016**, *7*, 285–294. [CrossRef]
42. Meena, Y.K.; Gopalani, D. Feature priority based sentence filtering method for extractive automatic text summarization. *Procedia Comput. Sci.* **2015**, *48*, 728–734. [CrossRef]
43. Hark, C.; Karcı, A. Karcı summarization: A simple and effective approach for automatic text summarization using Karcı entropy. *Inf. Process. Manag.* **2019**, *57*, 102187. [CrossRef]
44. Wang, H.-C.; Hsiao, W.-C.; Chang, S.-H. Automatic paper writing based on a RNN and the TextRank algorithm. *Appl. Soft Comput.* **2020**, *97*, 106767. [CrossRef]
45. Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In Proceedings of the Workshop on Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
46. El-Kassas, W.S.; Salama, C.R.; Rafea, A.A.; Mohamed, H.K. EdgeSumm: Graph-based framework for automatic text summarization. *Inf. Process. Manag.* **2020**, *57*, 102264. [CrossRef]
47. Mallick, C.; Das, A.K.; Dutta, M.; Das, A.K.; Sarkar, A. Graph-based text summarization using modified TextRank. In *Soft Computing in Data Analytics*; Springer: Singapore, 2019; pp. 137–146.
48. Liu, Y. Fine-tune BERT for extractive summarization. *arXiv* **2019**, arXiv:1903.10318.
49. Al-Sabahi, K.; Zuping, Z.; Nadher, M. A hierarchical structured self-attentive model for extractive document summarization (HSSAS). *IEEE Access* **2018**, *6*, 24205–24212. [CrossRef]
50. Jadhav, A.; Rajan, V. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1.

51. Ge, N.; Hale, J.; Charniak, E. A statistical approach to anaphora resolution. In Proceedings of the Sixth Workshop on Very Large Corpora, Montreal, QC, Canada, 15–16 August 1998.
52. Mitkov, R. Multilingual anaphora resolution. *Mach. Transl.* **1999**, *14*, 281–299. [CrossRef]