information

Article

Graph-Based Extractive Text Summarization Sentence Scoring Scheme for Big Data Applications

Jai Prakash Verma [1], Shir Bhargav [1], Madhuri Bhavsar [1], Pronaya Bhattacharya [2,3], Ali Bostani [4], Subrata Chowdhury [5,*], Julian Webber [6] and Abolfazl Mehbodniya [6,*]

1 Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, Gujarat, India; shirbhargav18@gmail.com (S.B.)
2 Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Kolkata 700135, West Bengal, India; pbhattacharya@kol.amity.edu
3 Research and Innovation Cell, Amity University, Kolkata 700135, West Bengal, India
4 College of Engineering and Applied Sciences, American University of Kuwait, Salmiya 20002, Kuwait
5 Department of Computer Science and Engineering, Sreenivasa Institute of Technology and Management Studies, Chittoor 517127, Andra Pradesh, India
6 Department of Electronics and Communication Engineering, Kuwait College of Science and Technology (KCST), 7th Ring Road, Kuwait City 13133, Kuwait; jwebber@ieee.org
* Correspondence: subrata895@gmail.com (S.C.); a.niya@kcst.edu.kw (A.M.)

Abstract: The recent advancements in big data and natural language processing (NLP) have necessitated proficient text mining (TM) schemes that can interpret and analyze voluminous textual data. Text summarization (TS) acts as an essential pillar within recommendation engines. Despite the prevalent use of abstractive techniques in TS, an anticipated shift towards a graph-based extractive TS (ETS) scheme is becoming apparent. The models, although simpler and less resource-intensive, are key in assessing reviews and feedback on products or services. Nonetheless, current methodologies have not fully resolved concerns surrounding complexity, adaptability, and computational demands. Thus, we propose our scheme, *GETS*, utilizing a graph-based model to forge connections among words and sentences through statistical procedures. The structure encompasses a post-processing stage that includes graph-based sentence clustering. Employing the Apache Spark framework, the scheme is designed for parallel execution, making it adaptable to real-world applications. For evaluation, we selected 500 documents from the WikiHow and Opinosis datasets, categorized them into five classes, and applied the recall-oriented understudying gisting evaluation (ROUGE) parameters for comparison with measures ROUGE-1, 2, and L. The results include recall scores of 0.3942, 0.0952, and 0.3436 for ROUGE-1, 2, and L, respectively (when using the clustered approach). Through a juxtaposition with existing models such as BERTEXT (with 3-gram, 4-gram) and MATCHSUM, our scheme has demonstrated notable improvements, substantiating its applicability and effectiveness in real-world scenarios.

Keywords: text mining; extractive text summarization; sentence scoring scheme; graph analytics; graph-based clustering; opinion mining

Citation: Verma, J.P.; Bhargav, S.; Bhavsar, M.; Bhattacharya, P.; Bostani, A.; Chowdhury, S.; Webber, J.; Mehbodniya, A. Graph-Based Extractive Text Summarization Sentence Scoring Scheme for Big Data Applications. *Information* **2023**, *14*, 472. https://doi.org/10.3390/info14090472

Academic Editor: Fei Liu

Received: 11 July 2023
Revised: 11 August 2023
Accepted: 15 August 2023
Published: 22 August 2023

1. Introduction

Modern Internet communication has shifted towards microblogging, community formation (on Twitter, Facebook, and other sites), and opinion-based polling. Most of the generated information falls under the wider umbrella of text mining (TM) schemes, where the data or information are gathered from devices and distributed to different users for consumption [1]. Another field is crowdsourcing [2], which includes the involvement of public opinions, work distribution, and community shares from a large group of persons. Mostly, the information is generated by mobile devices and the content is user-generated, with a significant amount of textual information. In social communities, many people post

raw data, which need to be analyzed for opinions in real time to make informed decisions. The problem is considered challenging in TM applications as a micro-blog post is usually very short and colloquial, and traditional opinion mining algorithms in the TM scope do not cater to a one-fit-all scheme in such scenarios.

Thus, effective text analytics schemes have been widely studied over the last decade [3,5**?** ,6], which has improved interactions in human life. Social media, blogs, and e-shopping are those domains that have benefited from this study. A rich environment for expressing ideas about a variety of topics, including service providers, logistics, and e-commerce platforms, has been created within the social community landscape thanks to the growth in feedback channels, customer reviews, and polling systems [7]. The tremendous amount of data that have been produced by this ecosystem makes it difficult to sort through. Specialized mining approaches are needed to analyze these data to find patterns, preferences, and behavior. Deploying artificial intelligence (AI) models is also essential in light of the emergence of open application programming interfaces (APIs), which enable the seamless exchange of information across social media and e-commerce platforms. These models assist in trend forecasting and allow for a more complex comprehension of consumer behavior in the digital market. The concern, however, remains that the data are shared over open public channels (Internet), and, thus, at the security front, crypto-primitives and privacy preservation techniques are important [8]. Once data are collected, different statistical and machine learning (ML) models are built for text summarization (TS), from which effective predictions can be made, which facilitates business decisions in smart communities [9,10].

Considering the scenario of e-commerce portals, the text document generally includes reviews, comments, and feedback provided by users on the purchased product [11]. In such cases, document summarization becomes an effective tool for analysis [12] as it filters out important and necessary information from multiple text documents. However, on the downside, as the number (frequency) of documents generated increases, the prediction analytics models become bulkier and costlier, which limits organizations to implement them due to budgetary constraints [13]. This limitation presents the requirement of robust algorithms for abstractive text summarization (ATS) [14] systems, which generate summaries of the documents based on the provided ML algorithm [15]. Figure 1 shows the classification taxonomy of ATS, where a three-tier classification is presented, namely based on the file size, the summarization approach, and the summarization algorithm approach [14].

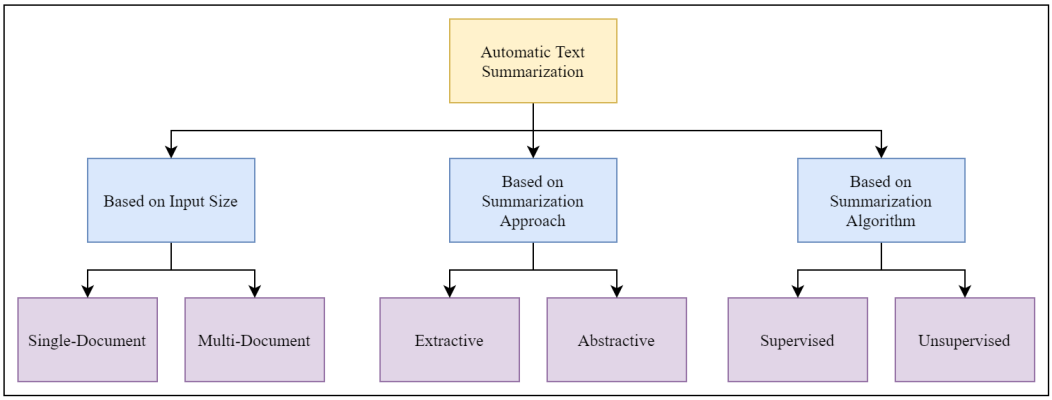

Figure 1. A classification taxonomy of ATS ecosystem [14].

Input-based ATS systems are further classified into two types, single-document and multi-document classification [16]. Single-document ATS, as the name suggests, summarizes the content of one document, whereas multiple-document ATS takes many documents in parallel and provides the summarization output [13]. Based on the document retrieval and semantic value obtained from the summary, an ATS is classified as an extractive or abstractive ATS, respectively [14]. An experimental review comparing abstractive and extractive summarization methods is presented in Refs. [17,18]. The review likely examines

their effectiveness and performance in generating summaries. Computationally, Extractive ATSs are less computationally expensive but suffer from semantic misinterpretation. An ATS involves semantic nets and grammar rules, which renders it ineffective for real-time applications. It is useful in cases where the quality of information is more important and thus accurate summary generation is highly required. Further, the classifications can be both supervised and unsupervised for the extraction of relationships from the main text document [19–21]. The primary goal of this study is to implement extractive text summarization (ETS) in a big data analytics environment. Based on a comprehensive literature survey, we have considered a substantial volume of text for summarization while simultaneously reducing computational expenses. As a result, this paper advocates for a statistical analysis approach to text summarization in contrast to abstractive text summarization. The discussion on this subject is also prominently highlighted in the paper.

Mostly in big data applications, owing to the text content bulk, an extractive ATS is a preferred choice. Figure 2 presents a generic overview of the extractive ATS, which can be further classified into three phases.

1. *Pre-processing:* Most text mining big data are unstructured and in raw form, and, thus, pre-processing steps are required to handle the text documents, along with natural language processing (NLP) schemes [22]. With NLP, the pre-processing steps include noise removal from text [23]. The noise removal comprises steps like case removal, tokenization, stemming, stop-word removal, parts of speech tagging, and many others. Once data are pre-processed, two types of feature extraction mechanisms are presented based on sentence level and word level. Sentence-level features include sentence length, sentence position, and frequency of word type in a sentence. Word-level features include word-related statics like word type, term frequency inverse document frequency (TF-IDF), and other related schemes.
2. *Sentence Scoring*: In this phase, rank is calculated from the given sentences based on which sentence selection. Different methods of sentence scoring are employed, namely TF-IDF, TextRank, word2vec, and others based on the similarity of the occurrence of nouns and numerical values [24]. However, the method of rank calculation is not universal and differs based on end application requirements. Recently, graph-based schemes have used the ranking algorithm PageRank based on graph traversal. ML models pose summarization as a classification problem [25].
3. *Summary Generation*: This phase generates the initial summary based on the rank calculated in the previous phase. For a graph-based model, it just selects the top required number of sentences based on rank. An ML model then just selects the sentences and classifies them as a true label and generates the summary.
4. *Post-processing*: The initial summary often contains different sentences having the same information, which increases redundancy and decreases the overall quality of a generated summary. So, various redundancy elimination techniques are implemented in this phase. Grouping techniques like clustering and itemset mining are preferred in this phase to group similar types of sentences. Miscellaneous processes like rearranging the sentences in summary based on their occurrence in the original document or inclusion of additional information like tables, figures, equations, and others are conducted.

In the ETS scheme, the task is to generate a mined document that contains the important information, without any hindrance to the semantical construct. In such cases, graph-based schemes are proven to be highly effective [26]. Graph-based ETS methods are powerful enough to capture the underlying structure of the text and form a graph $G = \{V, E\}$, where the vertices V (nodes) are the sentences in the text and edges E denote the relationship. The edges are formed based on similarity and coherence criteria. Thus, they are better than feature-based ETSs as they tackle the complex relationships between sentences and can be extensible as new sentences (nodes) can be added as required. Thus, graph-based ETSs are adaptable and highly flexible. The popular graph-based ETS methods include PageRank, LexRank, TextRank, Sumbasic, and the ensemble methods [27,28].

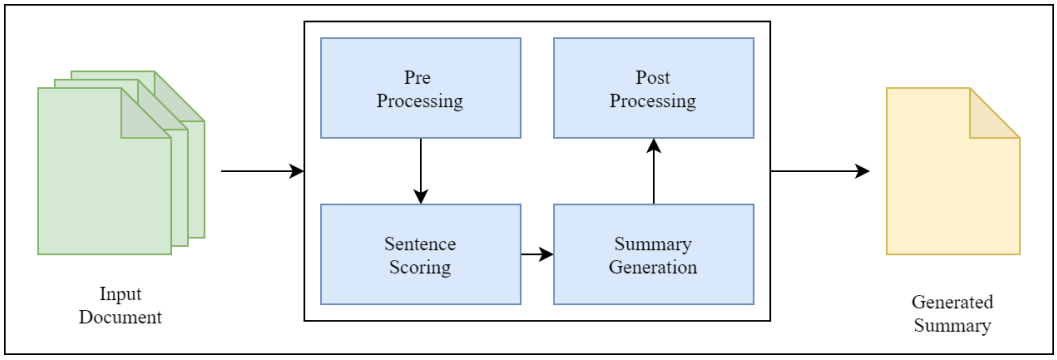

Figure 2. A generic model for extractive ATS [14].

1.1. An Illustration of Graph-Based ETS with Sentence Scoring Mechanism

In this subsection, we present the basics of graph algorithms to represent and process documents and illustrate the sentence scoring mechanism with details of ETS. Formally, we view a document as a graph $G = (V, E)$, where V represents a set of nodes and E presents a set of edges. Any node $v_i \in V$ corresponds to a sentence in the document, and an edge $(v_i, v_j) \in E$ indicates a semantic relationship between sentences v_i and v_j.

To indicate the strength of the semantic relationships between nodes and edges, we introduce weight w_{ij} of the edge connecting v_i and v_j. The weight computation is based on different similarity measures, like cosine similarity [29], Jaccard similarity [30], and semantic similarity [31]. Formally, the similarity measure Sim can be connected to w_{ij} as follows.

$$w_{ij} = Sim(v_i, v_j) \tag{1}$$

To score sentences, we consider the centrality measure, which quantifies the significance of the document (node) in the graph. A common choice is the degree of centrality, which is presented as follows.

$$C_D(v_i) = \sum_{j \in V} w_{ij} \tag{2}$$

The degree centrality $C_D(v_i)$ of a sentence v_i is thus the sum of the weights of the edges connected to it. Sentences with a higher degree of centrality are more likely to be selected for the summary as they are deemed more important.

As an example, let us consider four sentences as follows.

1. "The cat is on the roof."
2. "The dog is in the garden."
3. "The cat, from the roof, watches the dog."
4. "The dog is unaware of the cat."

The corresponding graph G would have four nodes, v_1, v_2, v_3, v_4, each representing a sentence. An edge is established between nodes that share semantic relationships, like, for example, between v_1 and v_3, and v_2, v_3, and v_4. Edge weights w_{ij} are assigned based on the strength of the semantic relationship. Applying the degree centrality measure, v_3 has the most connections (highest degree), and, thus, it would be deemed as an important sentence.

1.2. Novelty

In contemporary research, recent graph-based ETS schemes have instigated a migration towards optimized ranking algorithms. These algorithms are based to facilitate opinions, provide recommendation services, and work in a multitude of NLP applications. For ETS, in most of the existing approaches, simple heuristics (similarity matrix) are mainly used to rank the importance of nodes, which might not be the global picture of the entire text [32]. However, it is essential to recognize that these methods might not always capture the full contextual nuances and coherence necessary to represent the global picture of the entire text [33].

As research in ETS advances, there is a growing emphasis on exploring more sophisticated algorithms and techniques that can address the limitations of these heuristics and yield more comprehensive and cohesive extractive summaries. By delving into more context-aware and holistic models, ETS systems are striving to produce summaries that better reflect the overarching narrative and essence of the original text, thereby enhancing their utility and value for various applications. Motivated by these discussions, we present our scheme, *GETS*, which is tailored to extract use opinions based on sentence scoring algorithms. It enhances linguistic and contextual attributes, such as sentence length, relevance to the subject, and grammatical construction. Consequently, this fosters a superior level of semantic comprehension of the text, facilitating the creation of more refined and comprehensive summarizations. The scheme harbors the capacity to substantially augment the accuracy of the summaries generated, by integrating more sophisticated and exhaustive attributes into the scoring framework.

1.3. Research Contributions

The section provides the research objectives presented in the paper as follows.

1. Develop a graph-based sentence selection model to generate summaries for selected datasets. The text conversion to the graph model is conceptualized, and the relationships are presented.
2. Analyze the impact of sentence clustering as post-processing with the proposed scheme. Clustering applied at the post-processing level for the selection of the final summary can enhance the attainability of the summary of large documents.
3. Evaluate the proposed scheme by comparing it with the bidirectional encoder representations from Transformer (BERT)-based ETS.
4. The comparative analysis is presented in terms of ROUGE-1, ROUGE-2, and ROUGE-L for clustered and non-clustered work on the WikiHow dataset. The presented results indicate the scheme viability in real practical application setups.

1.4. Paper Organization

The rest of the paper is structured as follows. Section 2 presents a literature review and discusses the basic schemes and scoring algorithms that have been proposed by various researchers in the graph text summarization domain. Section 3 describes the proposed scheme with system architecture and execution of the proposed scheme and its components. Section 4 presents the methodology and concept applied to achieve defined objectives with a comparative analysis of various ranking algorithms. Sections 5 and 6 present the experimentation analysis, results, and discussion, respectively. Section 7 presents the conclusion and future scope for this paper. Table 1 shows the abbreviations and intended meanings used in the paper.

Table 1. Abbreviations and their meanings.

Abbreviations	Meanings	Abbreviations	Meanings
AI	Artificial Intelligence	LSTM	Long Short-Term Memory
ALBERT	A Lite BERT	ML	Machine Learning
API	Application Programming Interface	NLP	Natural Language Processing
ATS	Abstractive Text Summarization	NNW	Non-Noun Words
BERT	Bidirectional Encoder Representations from Transformers	POS	Parts of Speech
BLEU	Bilingual Evaluation Understudy	PPF	Positional Power Function
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	RAKE	Rapid Automatic Keyword Extraction
ETS	Extractive Text Summarization	RDD	Resilient Distributed Datasets

Table 1. *Cont.*

Abbreviations	Meanings	Abbreviations	Meanings
GAN	Generative Adversarial Networks	ROBERTa	A Robustly Optimized BERT Pretraining Approach
GloVe	Global Vectors for Word Representation	ROUGE	Recall-Oriented Understudying Gisting Evaluation
GPT	Generative Pretrained Transformers	SCU	Summary Content Unit
GRU	Gated Recurrent Unit	SVD	Singular Value Decomposition
HITS	Hyperlink Induced Topic Search	TF-IDF	Term Frequency Inverse Document Frequency
IDF	Inverse Document Frequency	TM	Text Mining
LCS	Longest Common Subsequence	TS	Text Summarization
LSA	Latent Semantic Analysis	UMLS	Unified Medical Language Systems

2. Background Schemes

As discussed in the introduction section, researchers have demonstrated the capability of the graph in the ATS system. It ranges from the simple implementation, for example, PageRank, to various combinations to improve the specific task of an ETS system, like domain coverage or the semantic value of summary, and others. The details are presented as follows.

2.1. Graph Analytics Frameworks

With the surge in graph applications, many graph-related frameworks have been established [34,35]. Neo4j is one of them, which is ACID-compliant, open-source, and supports the REST for device-to-device communications. Meanwhile, the GraphX framework provides support for multiple data sources, like CSV, MySQL, AWS storage, and others. It also provides faster execution of graph operations by using a distributed computing environment with Spark's Resilient Distributed Datasets (RDDs) to persist the data in RAM for in-memory execution. Apart from these, various high-scale graph frames are also defined [34], like Pregel, a vertex-centric graph processing framework, Blogel, the block-level type of graph processing framework, and PEGASUS, which focuses on the matrix representation of a graph and improves the processing power for the matrix-related operations by implementing functionality on map-reduce architecture.

2.2. Graph Model Generation

Based on the vertex type used in the graph, the graph model can be further divided into two approaches: 1. sentence-centric approach [36] and 2. word-centric approach [37], discussed as follows. In the later sections, different ranking algorithms and similarities are discussed, which are mainly used in the sentence-centric approach.

Sentence-centric approach: This model uses a sentence as a node, and various similarity measures like cosine, Jaccard, and others are used as a weight of an edge between two nodes (sentences). After that, a node-ranking algorithm is used to calculate the rank of a node. Based on the rank, sentences are selected either for the summary directly or by using the post-processing steps. The sentence-based model is preferable over the word-based model because it is less complex, requires small numbers of computation steps, and is visually more understandable. However, it also has a drawback as these types of graphs are generally fully connected. So, for the document with a higher count of sentences, it loses its simplicity and becomes a computer-intensive model. Figure 3 shows the graph model proposed by [37], which is the fully connected graph. Figure 4 demonstrates a basic process to generate the graph-based model.

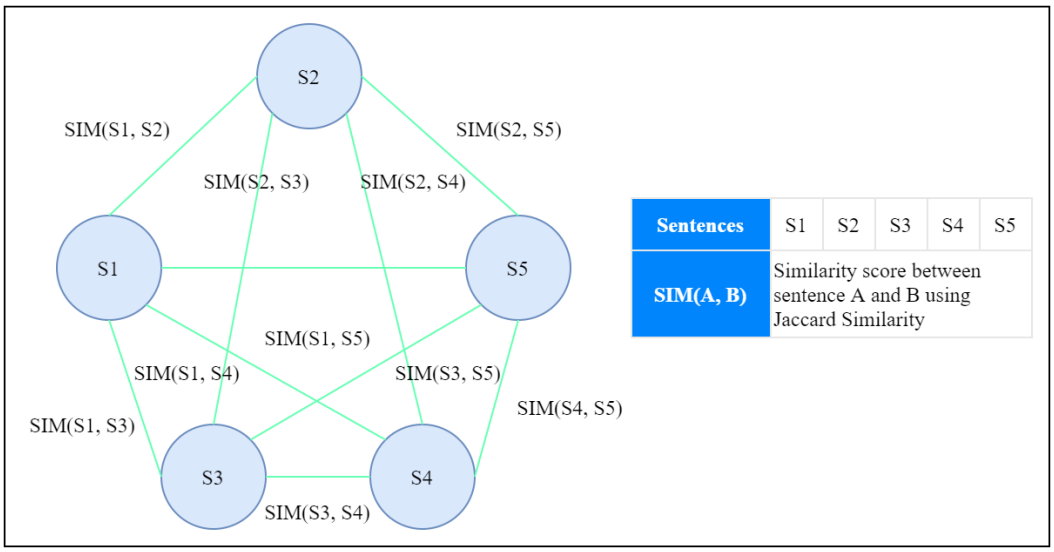

Figure 3. Sentence-based model proposed by [37].

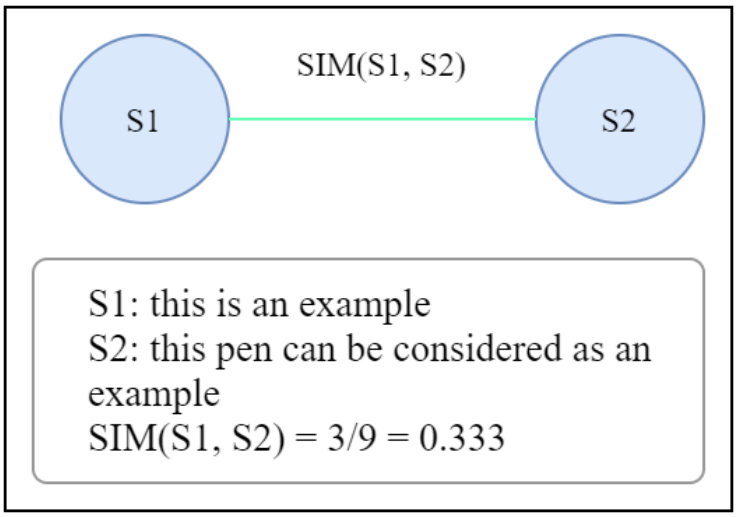

Figure 4. An example to generate sentence-centric model.

Word-centric approach: Opposite to the sentence-based model, the word is assigned as a node in a given scheme. The main challenge in this type of approach is to assign weights to the edges. Many parameters are considered, like N-gram pairs of words, adjacency words with position, POS tags, and others, but very few models have been implemented based on the given approach for the ATS system. There are several parameters behind this situation; first is the identification of a set of words for calculating the weight of the edge, as mentioned earlier, and the most impactful parameter is the number of words. With the comparison of the number of sentences, the count of words in each document is much higher, which leads to the high count of the nodes to represent in a graph as well as compute the high numbers of weights for each edge. So, in other words, this type of approach demands a more computation-intensive process. Some researchers have proposed a word-based model. We have used the model proposed by [38] to explain the basic idea behind this approach. Figure 5 shows the directed graph model proposed by them.

We first applied POS tagging to identify the grammar label of each word. Then, based on the grammar label, a graph is generated. As shown in Figure 5 above, they have assigned proper nouns as a node and words between two nouns as a directed edge. Each noun word has assigned some weight based on the position, count, occurrence in sections, and relation to the domain of the document. After the graph has been generated, each sentence is ranked by traversing words in the graph with respect to the given sentence. Averaged weight is assigned

to sentences, based on which the sentence selection process takes place. To demonstrate the graph, we used a single sentence to generate its word graph, which is shown in Figure 6.

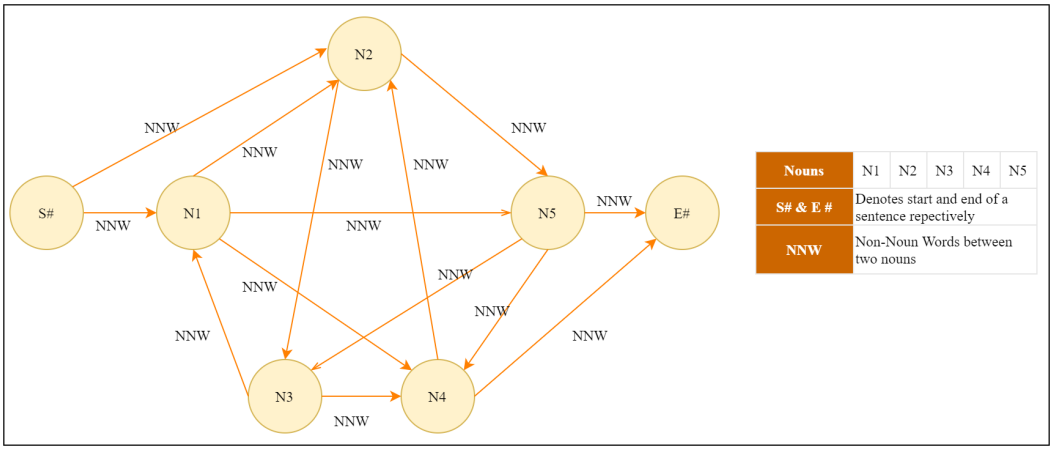

Figure 5. Word-based model proposed by [38].

In the sentence "In a word-based graph, a noun is represented as a node and other words between two nouns represented as a directed edge", "word", "graph", "noun", and "edge" are identified nouns in a test sentence, so these words will act as a node. Non-noun words (NNWs) are shown as an edge value. For example, in a sentence, NNW between ("Noun", "Edge") is "represented as a directed", which also reflects in a graph.

The similarity between sentences was measured using Jaccard similarity. As shown in Figure 6, it is a ratio between the common words and total words from both sentences. After this, a sentence-scoring algorithm has been applied to measure the node score. Then, with some post-processing steps, top-n sentences are selected for the summary.

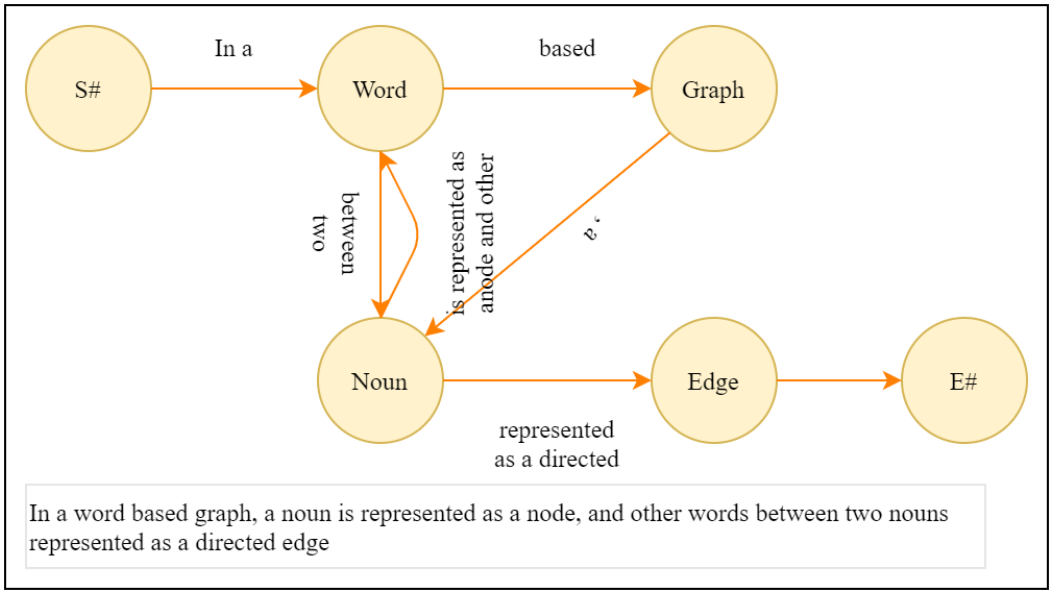

Figure 6. Example of word graph of a sentence.

2.3. Sentence Scoring Algorithms with Graph-Based Analytics

This section shows the sentence-scoring algorithms that can be applied with graph-based analysis models.

2.3.1. Hyperlinked Induced Topic Search (HITS)

As the name suggests, HITS is a link-ranking algorithm to identify important pages that contain information related to the search query. It was introduced by Jon Kleinberg in

1998 [39]. Even though it is targeted for webpage ranking in a WWW, it is used in many graph analytics processes. It calculates scores of the webpages based on pages that are linked with it as well as information stored in those pages. It generally maintains two parameters, 1. *Authorities:* It is a set webpage that contains highly related information related to the search query. 2. *HUBs:* It is a set of webpages with a link to the pages in authorities. So, HUB pages do not need to contain information regarding the search query.

2.3.2. PageRank

PageRank is another webpage ranking algorithm to estimate the importance of the webpage based on the number of pages that it links to and the quality of the pages. It was first presented by Sergey Brin and Lawrence Page [40]. PageRank mainly focuses on the outgoing links from webpages. Equation (3) provides the simplified version of PageRank.

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)} \tag{3}$$

where u = Current webpage, B_u = Set of webpages linked to u, $L(v)$ = Number of outgoing links from webpage v, $PR(x)$ = PageRank of webpage x.

2.3.3. TextRank

TextRank is a widely used sentence-ranking method in the field of text summarization, originally proposed by [41]. It finds application in two essential areas of NLP: sentence ranking and keyword extraction. In both cases, the algorithm leverages the PageRank approach to analyze a generated graph. For keyword extraction, TextRank applies the PageRank algorithm to a directed word-based graph. This graph is constructed based on the occurrence of words in the sentences (see Figure 5 for reference). By utilizing this graph, TextRank effectively identifies and ranks keywords within the text. On the other hand, for sentence ranking, TextRank adopts a sentence-centric graph-based model and applies the PageRank model to rank the sentences [17,42,43]. This method allows TextRank to determine the importance of each sentence in the text, leading to an effective text summarization process. It combines outgoing links from sentences and nodes with the similarity between two sentences, as mentioned in Equation (4).

$$TR(u) = \sum_{v \in B_u} \frac{sim(u,v) * TR(v)}{L(v)} \tag{4}$$

where u = Current sentence node, B_u = Set of sentence node linked to u, $sim(u,v)$ = Similarity score between sentence node u and v, $L(v)$ = Number of outgoing links from sentence node v, $TR(x)$ = TextRank of sentence node x.

2.3.4. LexRank

All the above-mentioned ranking algorithms use similarity measures that either work with the frequency of words or use common words between sentences. They do not capture the weight of words based on their, e.g., keywords. LexRank tackles this problem by including an extra measurement called inverse document frequency (IDF). LexRank combines IDF with cosine similarity (IDF-modified cosine) and uses it along with the PageRank algorithm, which can be derived in Equation (5).

$$LR(u) = \sum_{v \in B_u} \frac{IDF\_modified\_sim(u,v) * LR(v)}{\sum_{z \in B_v} IDF\_modified\_sim(z,v)} \tag{5}$$

where u = Current sentence node, B_x = Set of sentence node linked to x (adjacent nodes of x), $IDF\_modified\_sim(u,v)$ = IDF based similarity score of sentence node u and v, $L(v)$ = Number of outgoing links from sentence node v, $LR(x)$ = LexRank of sentence node x.

Table 2 provides the abstract view of the usage of different ranking algorithms in the research area. From the table, it is clear that PageRank is a much more trusted and used sentence-scoring method by researchers in the summarization domain.

Table 2. Ranking algorithms in research area.

Ref.	Objective	Pros	Cons	Node	1	2	3	4	5
Moradi et al. [44]	Comparison of effects of different input encoders in graph-based models.	Inclusion of domain-specific word embedding improves the generated summary	Cannot be used for general-purpose summarization as it more relies on the domain of the document.	S	✓	X	X	✓	X
Alzuhair et al. [45]	Implementation the combinations of two similarity measures and two ranking algorithms to find the optimal one.	Can be used for both single and multiple document summarization.	Results comparison can be improved by including more similarity and ranking algorithms.	S	✓	X	X	✓	X
Yang et al. [46]	Use combined weightage of words, bigrams, and trigrams to generate better sentence scores.	An integrated graph can learn and use semantic relatedness from three different dimensions.	It may require a high amount of resources in case of a big number of documents.	S	X	✓	X	X	X
Bhargava et al. [47]	Improve semantic values in summary by considering word position and overlapping of words.	A simple POS tag can be used to improve semantic value in summary.	A model can be complex for an extractive ATS system.	W	X	X	X	X	✓
Mao et al. [48]	Optimal balance between machine learning and graph-based sentence selection method.	ML model and graph model combined provide a better summary.	Author has not implemented any redundancy handling technique.	S	X	X	✓	X	X
Fang et al. [37]	Inclusion of importance of words during sentence score calculation to improve the sentence selection process.	Less complex and easy to implement	The assumption has taken that, initially, all words have the same weightage.	S	✓	X	X	X	X
El-Kassas et al. [38]	Induce word's weightage based on its type and position to improve the sentence selection process.	Model can generate great results for domain-specific documents or scientific papers.	Complex to implement; may require a high amount of resources.	W	X	X	X	X	✓

1: PageRank; 2: TextRank; 3: LexRank; 4: Hyperlinked Induced Topic Search (HITS); 5: Word-based; S: Sentence; W: Word; X: None.

3. Literature Review

Text summarization with basic features has been demonstrated in earlier research works. In our literature review, the oldest research work in text summarization was performed on scientific documents. In 1958, Ref. [49] proposed the method for summarization using basic features like word frequency, phrase frequency, and position in the sentence. In 1969, Ref. [50] performed the same task but used key phrases as an extra feature. After that, a better probabilistic model and features were introduced. Research work published by [51] in 1999 used the naïve Bayes classifier with the TF-IDF feature for improvement in the summarization model. After this, new methods like graph ranking algorithms and machine learning/deep learning models have been introduced. Recent research work has mainly focused on these methods.

Table 3 shows a comparative analysis of methodologies used by researchers in different models for text summarization processes. It shows that researchers mainly used schemes like extractive or abstractive text summarization on single or multiple documents [52]. The methodologies like deep learning, machine learning, data mining, and graph-based text summarization are compared along with the findings. The methodologies presented in these papers are mainly emphasizing preprocessing steps. It has emerged that improvised text summarization can be achieved by applying post-processing with a text summarizing model. Sentence selection algorithms are generally used to display sentences with the most information. They have various applications in different domains [53]. Nowadays, many users buy from e-stores and review the products. Most purchases are completed

based on these reviews given by the users, but reading all reviews can be time-consuming. The solution to this problem is to select a certain amount of top sentences from all reviews and display them as a summarized review.

Table 3. Comparativeanalysis of state-of-the-art schemes.

Author	Methodology	Type	1	2	3	4	5	I/p Type
Patel et al. [54]	Combination of word-level and sentence-level features to obtain best results for multiple documents. Word Features: Given fixed weightage to a word based on its specific position (title, keyword), type (noun, numerical), and domain. Sentence Features: Sentence position, length, and cosine similarity.	E	X	X	X	✓	X	M
Mao et al. [48]	Use of LexRank for calculating scoring of node. Three models: (1) Linear combination. (2) Use graph score as a feature. (3) Use the decision of supervised result in graph scoring.	E	✓	X	✓	X	X	S
Van Lierde et al. [55]	Use DBSCAN for semantic clustering of words to generate a topic-based model with the constructed hypergraph	E	X	X	✓	X	✓	S
Chaves et al. [15]	Calculate cosine similarity between sentence words and keywords as well as with words of other sentences. Obtain averaged similarity of all words in a sentence and assign it as a sentence score.	E	X	✓	X	X	X	S
Jindal et al. [56]	Apply TF-TDF and rapid automatic keyword extraction (RAKE) for word-level features. Find optimal no. of clusters and apply fuzzy c clustering for sentence-level features.	E	X	X	X	✓	✓	S
Du et al. [57]	Apply genetic algorithm for optimal weights of above features. Uses a fuzzy logic system to calculate sentence scores.	E	X	X	X	✓	✓	S
Moradi et al. [58]	Find topic-wise using itemset mining. Apply hierarchical clustering on support values to group similar types of sentences.	E	X	X	X	✓	✓	Both
Bhargava et al. [59]	Label generation by mapping document and its pre-summary made by human. Convert sentence to vector using embedded layer. Then, pass it through the convolution layer for feature extraction.	E	X	✓	X	X	X	S
Anand et al. [60]	Generate labeled data by using various similarity measures between legal documents and head notes. Generate embedded vector combined with cosine similarity.	E	X	✓	X	X	X	M
Alami et al. [61]	Generation of word2vec model. Used three variations of autoencoders: a. auto encoder b. variation AR c. extreme learning machine AE 3. Use the voting method to obtain cumulative ranks of sentences.	E	X	✓	X	X	X	S
Azadani et al. [62]	Map document to concept using MetaMap developed for Unified Medical Language System (UMLS). Use the itemset mining method to obtain frequent itemsets from documents. Jaccard similarity to obtain similarity measure between two itemsets.	E	X	X	✓	X	✓	S
Liang et al. [63]	Combination of cross-entropy and reinforcement learning to improve performance of seq2seq attention model	A	X	X	X	X	X	S
Adelia et al. [64]	Vectorize sentence with word2vec model.	A	X	✓	X	X	X	S
Moirangthem et al. [65]	Seq2seq attention model and pointer generator network. Improve performance of model and sentiment value of summary for large documents.	A	X	✓	X	X	X	S
Ghodratnama et al. [13]	Posed as a classification problem. First cluster the sentences and then send information to the classifier for weight updating of features. Linguistic feature: each sentence is a subgraph of three forms: subject, predicate, and object.	E	X	X	X	X	✓	M
Guo et al. [66]	Improvement in the grammar of summary. Used pointer network for out-of-vocabulary problems. Multi-head self-attention model is used to generate a semantic summary.	A	X	✓	X	X	X	S
Mackey et al. [12]	Generate an inverted index of training dataset in Hadoop. Use the TF-IDF score to rank sentences.	E	X	X	X	X	X	S
Cagliero et al. [67]	Generate word to sentence frequent itemset matrix. Apply Singular Value Decomposition (SVD) on the resulting matrix to identify concepts covered by sentences. Select the sentence with the most covered concepts.	E	X	X	X	X	✓	S
Fang et al. [37]	Calculate word's rank using sentence rank. Calculate new sentence rank by using the weight of every word. Use a linear combination of each sentence rank as sentence selection rank	E	X	X	✓	X	X	S
Giarelis et al. [17]	Presented an in-depth analysis on the TS process, datasets, and outlined the evaluation methods and metric selection principles	Both A and E	✓	✓	N	N	✓	M
El-Kassas et al. [38]	Calculate node weightage using parameters like word type, the domain of the word, and the position of the word in the document (e.g., title, subject, keyword, etc.). Generate candidate summary by calculating the average weight nodes of the selected path. Apply K-means clustering in the candidate summary to remove redundancy.	E	X	X	✓	X	✓	M
Alomari et al. [18]	A comprehensive review of the application of deep reinforcement learning and transfer learning techniques in the context of abstractive text summarization. The study delves into the current state-of-the-art works and explores how these methods can be utilized to generate concise and coherent summaries from large textual content.	A	X	✓	X	X	✓	M

E: Extractive; A: Abstractive; 1: Machine Learning; 2: Deep Learning; 3: Graph; 4: Fuzzy-logic-based; 5: Data Mining Techniques (Itemset mining and clustering); S: Single Document; M: Multiple Documents.

With the recent research advancements in machine learning methods, researchers proposed many models for extractive text summarization with machine learning models. Oussam Rouane et al. [68] proposed a model with K-means clustering for concept recognition, which prevents the repetition of a particular type of sentence and covers the major concepts in a generated summary. Chih-Fong Tsai et al. [69] utilized multiple reviews of a hotel by classifying it into a helpful review or not and used this feature to improve the sentence-scoring method. Mudasir Mohd et al. [70] derived a novel method for sentence ranking, which is applied to clusters generated with the use of K-means clustering on word2vec vectors. B. Cao et al. [71] present an unsupervised learning-based text summarization approach for short text extracted for patients about their disease with consideration of privacy issues.

Deep learning models have also promised a significant amount of advancement in document summarization [46]. Recurrent neural network (RNN) and its different variants are one of the most preferred models by researchers because of their capability to model complex underlying relationships in sequential data. Mohamed Seghir Hadj Ameur et al. [72] used a bi-directional gated recurrent unit (GRU) to learn dependencies between words for extractive text summarization. Qingyu Zhou et al. [73] applied a two-layer normalized RNN model on vectors encoded (BERT) to extract summary sentences, which are also discussed in [74]. Rupal Bhargava et al. [36] proposed a paraphrase detection algorithm to reduce redundancy in the text by removing similar paraphrases along with generative adversarial network (GAN).

Milad Moradi et al. [44] proposed an undirected graph model with sentence vectors as nodes and cosine similarity as the weight of an edge. They used bidirectional encoder representations from Transformers (BERT) encoding for sentence2vec conversion. Three different ranking algorithms, PageRank, HITS, and positional power function (PPF), are used to calculate sentence rank. Abeer Alzuhair et al. [45] compared different similarity measures and ranking methods in an undirected weighted graph model. Cosine similarity, Jaccard similarity, and identity similarity are used for similarity measures comparison and PageRank and HITS for a ranking method comparison. Kang Yang et al. [46] implemented three different graph models by including three different sets of extra information with the sentence as a node and cosine similarity as a weighted edge. They used words along with their POS tags, Bigrams, and Trigrams as extra information for each graph, respectively. The TextRank method has been used to rank a sentence on a single graph made by combining all three different graphs using a naïve Bayesian fashion. Rupal Bhargava et al. [47] used a directed graph model for ATS with the use of the word, its POS-Tag, and its relative position in sentences.

4. *GETS*: The Proposed Scheme

In this paper, the graph-based sentence-centric scheme for the selection of representative sentences to generate a summary for the given input text approach is proposed, in which sentence is used as a node and Jaccard similarity as a weight of the edge between nodes (sentences). We used the ROUGE metric to evaluate the model. It demonstrates the influence of words in the sentence-scoring process. To generate an improved version of the summary, one of the important parameters can be word weightage. So, we have identified and implemented one extra process that can be employed to extend the capabilities of the proposed scheme, which is implemented in the data pre-processing phase and post-processing phase. Figure 7 shows the complete process flow and execution flow with different components of the proposed scheme.

4.1. System Architecture and Flowchart

As shown in Figure 7, the given model can be divided into three parts: 1. pre-processing—containing text pre-processing and feature extraction; 2. processing; and 3. post-processing, described as follows and also shown in Figure 8. The process starts with sentence segmentation, which extracts sentences from the raw CSV. Consequently,

a sentence-level CSV is generated and saved in a Spark-based database. PySpark APIs will then be used to perform all actions on the Spark tables. Pre-processing includes fundamental operations like lowercasing, white noise removal, etc. The flow continues with additional activities, including POS tagging and Jaccard similarity calculation, which aid in rank generation during the processing stage. The feature extraction flow involves word frequency computation and keyword generation, both of which contribute to word weighting. Graphs are generated using the Jaccard similarity score in the process flow, and a combined rank is determined using the random walk algorithm and word weighting. Subsequently, the flow splits into two branches. The first branch generates summaries based on the combined rank, while the second branch utilizes GraphFrames to perform graph clustering on the sentence graphs, producing a cluster-based rank. A summary is then generated based on this rank. As the summary of the flow, both of these summaries are evaluated during the evolution stage using the ROUGE score across various document lengths.

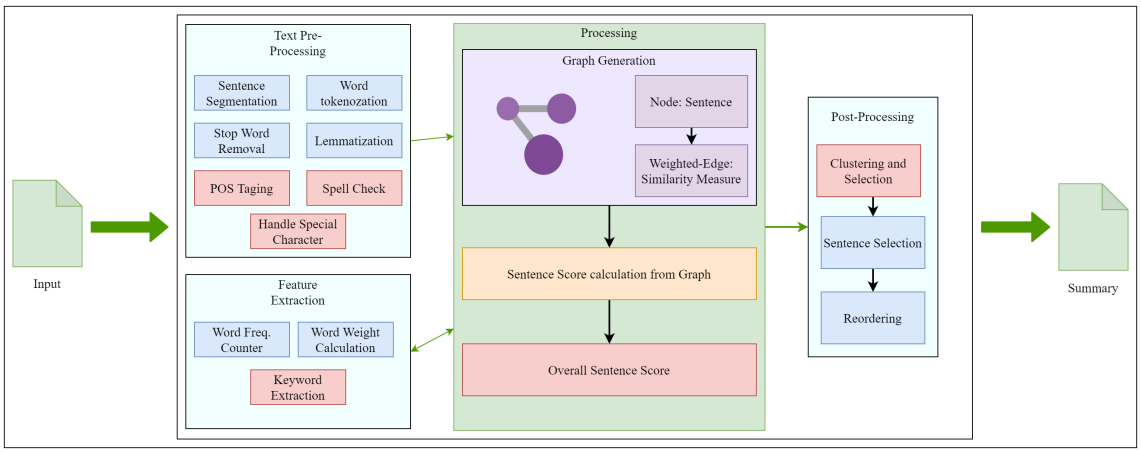

Figure 7. Graph-based extractive text summarization architecture.

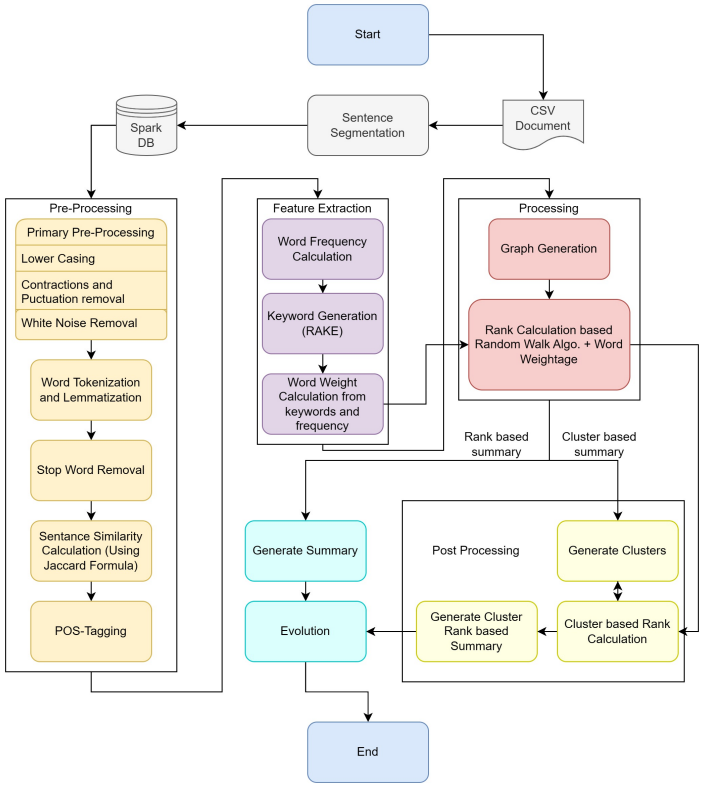

Figure 8. Flowchart for execution flow of proposed scheme.

4.1.1. Phase I—Data Pre-Processing Steps

The most used pre-processing steps are identified and implemented; these processes are as follows: spell check, case removal, tokenization, stopword removal, handle special characters, stemming/lemmatization, POS tagging, TF-IDF, word frequency calculation, and keyword extraction. Following equations are used to calculate the rank of each sentence.

Random Walk: Equation (6) is used to calculate a graph-based random walk score for each sentence node.

$$R_i = (1-d) + d \sum_{i,j=0}^{N} \frac{SM_{i,j}}{SM_j} R_j \tag{6}$$

where R = Rank list of sentences, R_i = Rank of ith sentence, d = damping factor to handle node with zero degree; $\in [0,1]$, N = Number of sentences, SM = Similarity score matrix $(N \times N)$ calculated using Jaccard Similarity, $SM_{i,j}$ = Element at ith row and jth column of SM.

Frequency-based word score: Equation (7) calculates a score of each word based on sentence scores obtained through the random walk.

$$WSf_j = \frac{\sum_{i=1}^{N} C_{ij} R_i}{\sum_{i=1}^{N} C_{ij}} \tag{7}$$

where WSf = Frequency-based Word score, WSf_j = Score of jth word, M = Total numbers of words, C = Count matrix $(N \times M)$, contains count of each word in sentence, C_{ij} = Frequency of jth word in ith sentence.

Type-based word score: Equations (8) and (9) calculate normalized score if the word is noun, keyword, or both. Meanwhile, Equation (10) combined both mentioned scores, so, for words that are not nouns and keywords, the WSt score will be 0.

$$Word_{noun} = \frac{freq(word)}{\sum_i freq(Noun_i)} \tag{8}$$

$$Word_{keyword} = \frac{freq(keyword)}{\sum_i freq(KeyWord_i)} \tag{9}$$

$$WSt = Word_{noun} + Word_{keyword} \tag{10}$$

where $Word_{noun}$ = Normalized score if the word is a noun, $Word_{keyword}$ = Normalized score if word is keyword, WSt = Type based Word score, $freq(word)$ = Frequency of word in a document, $Noun$ = List of nouns present in a document, $KeyWord$ = List of keywords present in a document.

Word Score: Equation (11) simply combines both different types of word scores to assign a final score.

$$WS = WSt + WSf \tag{11}$$

where WS = Final word score.

Word influenced rank: Include word weightage into sentence score using Equation (12).

$$R_i^* = \frac{\sum_{j=1}^{M} SC_{ij} WS_j}{\sum_{j=1}^{M} C_{ij}} \tag{12}$$

where R^* = Word influenced rank list of sentences, R_i^* = Word influenced rank of ith sentence, SC_{ij} = Similarity between ith sentence and jth sentence.

Hybrid Rank: The linear combination has been used to include the influence of random walk score and word-influenced score as shown in Equation (13).

$$R_i = \alpha R_i + (1-\alpha) R_i^* \tag{13}$$

where $\alpha \in [0, 1]$; to balance the overall score.

Smoothing Counts: The word count for every sentence has been normalized using Equation (14) to improve processing time.

$$SC_{ij} = \frac{C_{ij}}{\sum C_i} \tag{14}$$

where SC = Normalized count matrix $(N \times M)$ of C, C_{ij} = Frequency of jth word in ith sentence.

Cluster Rank: Calculates the rank of the cluster based on the similarity scores of sentences for a given cluster.

$$CR_i = \sum_{j \in Ne} SM_{ij} \tag{15}$$

where CR_i = Cluster Rank of ith cluster, SM_{ij} = Similarity score between ith and jth sentence, i and $j \in C_i$ cluster.

4.1.2. Phase II—Processing with Graph

Three processes come under graph processing, namely graph generation, graph-based score calculation, and overall score calculation. In graph generation, the sentences are modeled as nodes and Jaccard similarity scores between those nodes are modeled as the weight of edges between them. This scenario is represented as a similarity matrix mentioned in data pre-processing. In graph-based score calculation, the score is computed based on the random walk Equation (6). In overall score calculation, it combines both the graph-based score and word-influence score to finalize the rank of given sentences. Algorithm 1 describes step-by-step execution of the processing phase.

Algorithm 1 *GETS*: Sentence Scoring Algorithm

Input: S_i: Sentence list
SM: Sentence Similarity Matrix/Graph
SC: Sentence–word Smoothen Count Matrix calculated using Equation (14)
d, α, ϵ: Parameters
NOS: Number of sentences to be selected
Output: S_o: a set of sentences
Initialize $i \leftarrow 0$
Initialize R^i
while $\|R^i - R^{i-1}\|^2 \geq \epsilon$
Calculate R^{i-1} using Equation (6)
Calculate WS using Equation (11)
Calculate R^* using Equation (12)
Calculate R^i using Equation (13)
$i \leftarrow i + 1$
$S_i \leftarrow$ sort-in-score-descending-order (S_i)
Selection: $S_o \leftarrow S_o \cup top\_S_i(NOS)$

4.1.3. Phase III—Post Processing

This stage includes the miscellaneous processes like sentence sorting based on rank, sentence selection, and sentence rearrangement, as mentioned in Algorithm 1. We have included graph clustering as an additional process to understand the effect of graph clustering in the sentence selection process. Algorithm 2 represents the detailed steps of post-process stage:

1. *Graph Clustering:* Connected components algorithm has been used to identify the clusters within the graph. Initially, the sentence graph generated in the previous stage will be fully connected. To remove the edges, first are edges between sentences that

do not have any similarity; secondly, a new threshold β is used. So, if the weight of an edge is less than the threshold, β is removed.

2. *Cluster Rank sort:* This stage sorts the sentences within the cluster based on the sentence ranks. Given rank is calculated using Equations (13) and (15).
3. *Sentence Selection and Reordering:* As the sentence is sorted, top sentences are selected from each cluster based on the number of required sentences, then reordered based on their occurrence in the original document.

Algorithm 2 Sentence selection process using Graph Clustering

Input: G: Sentence graph
β: Parameters
NOS: Number of sentences to be selected
Output: S_o: a set of sentences
For each $edge(E)$ in G:
if $weight(E) < \beta$:
$remove(E)$
clusters = $get\_clusters(G)$
For each $cluster(C)$ in clusters:
For each $node(N)$ in cluster(C):
Calculate $cluster\_rank$ using Equation (15)
Calculate $final\_rank$ using $cluster\_rank$ and Equation (13)
$S_i \leftarrow sort - in - score - descending - order(C, final_rank)$
Selection: $S_o \leftarrow S_o \cup top\_S_i(NOS)$

5. *GETS*: Performance Analysis

5.1. Dataset Selection

We have used large-scale dataset "WikiHow" as well as "Opinosis" for the training and testing of the proposed scheme. WikiHow dataset contains instruction articles on different topics, hosted on wikihow.com. It includes more than 230,000 articles [75]. Diversity is the main advantage of this dataset. Most of the preferred datasets are composed of news articles. These articles are written by journalists, which often use the same type of writing methods, like inverted pyramid writing style [75]. Meanwhile, topics on WikiHow were written by different individuals. We have used a small subset of this dataset, which is available with the sentence along with its labels for inclusion. As the given dataset comes with the extractive type of reference summaries, the WikiHow dataset is used to evaluate the quality of generated extractive summaries by the proposed scheme with the mentioned base model.

We have also used a small-scale dataset named "Opinosis" to further test the proposed scheme for justification of the post-processing step included in the proposed scheme. It comprises 51 topics with reviews from multiple users for each topic. It contains an average of 100 sentences for each topic [76]. The dataset also comes with the reference summary known as the gold summary made by humans. The main advantage of the mentioned dataset is that it contains the repetitive type of sentences in the documents; because of that, this dataset is used to evaluate our post-processing method, which is focused on reducing the similar type of sentences in summary.

5.2. Evaluation Measures

In this subsection, we present the details of different metrics that are used to evaluate extractive text summaries generated by the photograph-based text summarization model [77–81]. Evaluation of the summary can be completed on multiple bases. For example, consider the evaluation based on the application of the ATS system. In some scenarios, ATS systems have been employed to extract useful information. In this case, summaries are evaluated based on information covered in the input document, and, opposite to the given case, some researchers have developed ATS systems to generate human-like summaries.

So, in this situation, evaluation is completed based on the presence of grammatical errors in the summaries. To cover most types of measures, we have followed categories defined by Ref. [82] for the evaluation of summaries. In this paper, we applied content-based measures for evaluation of the extractive summary generated by the proposed scheme [79,83,84].

1. **Cosine Similarity [85]:** It is a vector-space-based similarity measure. Sentences are first converted into vectors and then their similarity score is calculated using Equation (16).

$$Cosine\ Similarity(X, Y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i (x_i)^2}\sqrt{\sum_i (y_i)^2}} \tag{16}$$

   where X and Y are a vector representation of generated summary and referenced summary, respectively.
2. **Unit Overlap [86]:** It measures similarity based on the common words or lemmas between generated summary and referenced summary, which can be formulated using Equation (17).

$$Overlap(X, Y) = \frac{\|X \cap Y\|}{\|X\| + \|Y\| - \|X \cap Y\|} \tag{17}$$

   where X and Y are the set of words or lemmas of generated summary and referenced summary, respectively.
3. **Longest Common Subsequence (LCS) [87]:** It is the simplest similarity measure, which calculates similarity based on the longest common subsequence between generated summary and referenced summary. Equation (18) provides the basic formula of LCS measure.

$$LCS(X, Y) = \frac{length(X) + length(Y) - edit(X, Y)}{2} \tag{18}$$

   where X and Y are a sequence of words or lemma form of generated summary and referenced summary, respectively. The $edit(X, Y)$ is the edit distance between X and Y. Edit distance is the number of operations required to convert X into Y.
4. **Bilingual Evaluation Understudy (BLEU) [79]:** BLEU is frequently used in the field of machine translation. BLEU is defined by Equation (19).

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{19}$$

   where BP is the brevity penalty, w_n are the weights for each N-gram (usually equal when N = 4), and p_n is the precision for each N-gram.
5. **ROUGE [78]:** Recall-oriented understudying for gisting evaluation (ROUGE) is a powerful evaluation measure to assess the generated summary [88]. It evaluates the generated summary by comparing it with various human-generated reference summaries. Based on its comparison method, it has various variants, for example, the ROUGE-N comparison of N-gram, the ROUGE-L comparison of the longest subsequence, and ROUGE-S, which uses the skip-gram model. The basic formula of ROUGE-N is presented in Equation (20).

$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{20}$$

   where $gram_N$ is an N-gram, $Count_{match}(gram_N)$ is the maximum number of N-grams co-occurring in a candidate summary and a set of reference summaries, and $Count(gram_N)$ is the number of N-grams in the reference summaries.

6. **Latent Semantic Analysis (LSA) [80].** LSA is a method mainly used to reduce dimensionality within a given text corpus and transforms a document to the term–document matrix, which we denote by A. On the document, we apply the SVD, which is presented in Equation (21).

$$A = U\Sigma V^T \tag{21}$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix containing singular values. The dimensionality reduction in LSA is mainly attributed to matching the top k largest singular values in Σ and corresponding columns/rows in U and V^T. The resultant matrix is denoted as $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$, where $\hat{(.)}$ function represents the lower dimensional semantic space. LSA is not a metric in itself; rather, it employs cosine similarity in between document vectors to measure the semantic similarity.

7. **Pyramid [81]:** This method defines a new unit called summary content unit (SCU) to compare information covered in generated summary with the reference summary. SCU is defined by human judges, which is semantically motivated, variable length, and can be of one or more words/lemmas. The best try is that SCU covers most of the information stored in the sentences. So, sentences are given a rank based on the SCUs covered by them. The pyramid is generated based on these SCUs. At the top of the pyramid are those SCUs that appeared in most of the summaries.

Table 4 presents a comparative overview of evaluation metrics. The metrics are summarized in terms of objective, relative metrics, and the applications in which they are used. From the table, it can be inferred that the metric selection for a given application depends on the nature of the application, its associated tasks, and the associated performance measures. For example, cosine similarity is suitable for vector space models and should be used when semantic similarity is paramount. Unit overlap and LCS are effective in tasks where sequence preservation is important, like machine translation and text-to-speech analysis. Complex metrics like BLEU, ROUGE, and Pyramid are more suitable for the TS domain. While BLEU and Pyramid are widely used in summarization and understanding of the context, ROUGE is more suitable and adjustable for ETS, where recall is a critical component.

Table 4. Summary of TS evaluation metrics.

Metric	Objective	Merits	Applications
Cosine Similarity	Measures the cosine of the angle between two vectors.	Captures semantic similarity, efficient for high-dimensional vectors.	Information retrieval, text mining.
Unit Overlap	Computes the overlap of words in two texts.	Simple to calculate, highly interpretable.	Text similarity, text classification.
LCS	Finds the longest subsequence common to two sequences.	Robust to paraphrasing.	Plagiarism detection, genetic sequences analysis.
ROUGE	Measures the overlap of N-grams, word sequences, and word pairs.	Captures recall-based metrics with variants (ROUGE-N, ROUGE-L, and others).	Text summarization, machine translation.
Pyramid	Evaluates content units in summaries.	Comprehensive assessment of summaries.	Text summarization, particularly multi-document.
BLEU	Compares N-grams of machine output with that of a human reference.	Precision-oriented, good for aggregate data.	Machine translation, text summarization.
LSA	Unearths the latent semantic relationships within a corpus.	Can capture contextual and conceptual similarity.	Information retrieval, text summarization, topic modeling.

5.3. Selection Choice

The aforementioned metrics offer a unique perspective to the summarization problem. In particular, we have selected the ROUGE-N metric (including ROUGE-1 and ROUGE-2) as the principal metric [78]. The selection choice is based on the robustness and wide

acceptance of this metric in ETS. In ETS, it is more important to capture details about the information source, and, thus, recall is a critical measure. However, the BLEU metric is more oriented towards precision than recall value, which makes it a weak candidate for extractive summarization tasks. Similarly, the LSA metric operates under the assumption that words used in similar contexts share semantic meaning. Due to this, LSA might overlook nuances in sentence construction and semantic orientation. Further, LSA requires a high semantic space for computation, which slows down the processing of real-time big data frameworks. The Pyramid scoring metric evaluates the weight of summary content but is labor-intensive and not suitable for large-scale dataset evaluations. The considered framework on the dataset focuses to maximize the semantic context, and, as these metrics do not directly measure the impact of the recall on the information, a high score on precision would not correlate directly with the summary effectiveness. Applying these metrics to our dataset, thus, the application of metrics BLEU, LSA, and Pyramid would fetch high precision scores owing to the semantic overlap but reduce the model's ability to capture key information, which is the most crucial aspect of ETS.

In our evaluation, we have considered the WikiHow [75] and Opinosis datasets [76]. The WikiHow dataset is normally sequential and structured, and the Opinosis dataset is known for user reviews. In these datasets, the evaluation of cosine similarity, overlap coefficient, and LCS would fundamentally revolve around the semantic and textual overlap. For WikiHow, the semantic relationships would be captured by these metrics; thus, they would yield high scores. In contrast, the Opinosis dataset, being user-generated, is prone to contain repetitive phrases and sentiments. Thus, the model's capability to render the overlaps into succinct summaries would be limited. The metrics might fall short in assessing whether the summary captures the key ideas and information of the original text, which is a crucial aspect of ETS.

In this context, the ROUGE metric measures the overlap of the N-grams between system and reference summaries and thus is contextually meaningful to our framework. Unlike other metrics, the ROUGE metric is not primarily dependent on the structural and semantic overlap but focuses on the salient information of the source text. Given the diversity of the datasets, using ROUGE as the primary evaluation measure allows the framework to assess summaries more effectively in terms of relevance. Thus, a balanced assessment of semantic equivalence and information makes it an effective fit for the problem.

6. Evaluation Setup

To evaluate the proposed scheme, the "WikiHow" and "Opinosis" datasets have been used. In the WikiHow dataset, 500 documents have been selected and classified into the five different classes as shown in Table 5. Each class was assigned 100 documents, which is useful to learn the ability to model over the different lengths of the documents. In this study, two different models have been used. Model M1 follows Algorithm 1, while model M2 is an extension to M1, which employs graph clustering as the post-processing phase that uses Algorithm 2. The parameters are set as defined in Table 6 for the execution of respective operations described in the third column. We have used the recall-based ROUGE parameter to evaluate our model. The length of the generated summary is the same as the reference summary. The project's source code, dataset, and all relevant supporting documents have been deployed on GitHub, utilizing a public shared repository. This allows for easy access and collaboration with the broader community (Source Code Link: https://github.com/shirbhargav/Graph-based-Extractive-Text-Summarization-Sentence-Scoring-Scheme-for-Big-Data-Applications (accessed on 14 August 2023)).

Table 5. Classes of documents based on the length.

Class	Length of Documents
C1	Between 0–50
C2	Between 50–100
C3	Between 100–150
C4	Between 150–200
C5	Greater Than 200

Table 6. Parameters.

Parameter	Values	Description
d	0.8	Damping factor to handle node with zero degrees (refer Equation (6))
α	0.6	Where $\alpha \in [0,1]$; to balance the word-infused rank and random walk rank (refer Equation (13))
ϵ	10^{-5}	A convergence factor for word-based rank calculation (refer Algorithm 1)
β	Q3 percentile of all similarity values	To remove the edge with very low similarity relations between sentences in the graph (refer Algorithm 2)

6.1. Discussion and Limitations

As per the discussion, the recall-oriented understudying for gisting evaluation (ROUGE) evaluation measure is selected. Table 7 shows the average ROUGE-1, 2, and L values obtained for each class using models M1 and M2. If we focus on the individual performance of each model, then it is clear that both models provide the better result for class C3 in all three ROUGE score areas. Another visible trend that can be observed is that the ROUGE-2 value is very low compared to the other two ROUGE scores. The reason behind this trend is that ROUGE-2 compares the pairs of consecutive words between the generated and reference summary for each sentence and the probability of having the same pair of consecutive words in a generated summary becomes very low.

Table 7. Recall-based ROUGE score with/without clustering for WikiHow dataset.

	Without Clustering			With Clustering		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
C1	0.4121	0.1116	0.3523	0.3942	0.0952	0.3436
C2	0.4492	0.1194	0.3742	0.4219	0.0946	0.3572
C3	0.4850	0.1304	0.3780	0.4397	0.0915	0.3505
C4	0.4761	0.1171	0.3654	0.4296	0.0828	0.3333
C5	0.4487	0.1004	0.3536	0.4007	0.0714	0.3137

Figures 9–11 show the performance comparison between model M1 and M2 in terms of ROUGE-1, ROUGE-2, and ROUGE-L, respectively. If we compare these results with the selected base model (BERT-based state of art extractive text summarization model presented by Ming Zhong et al. [89]), it is clear from Figure 12 and Table 8 that both of the proposed schemes M1 and M2 outperform the base model. The proposed schemes have generated better results on ROUGE-1 and ROUGE-L scores compared to the base model. According to the result presented in Table 8 and the referenced paper [89], the authors propose a

model called "MATCHSUM", and they claim that it outperforms other competing models, including "BERTEXT" and "BERTEXT + 3gram-Blocking", "BERTEXT + 4gram-Blocking", and even the baseline model with "BERT-large" as the pre-trained encoder. The comparison is completed using the ROUGE-1 score, which is a common metric used to evaluate the quality of text summarization.

From Figures 9–11, it is visible that M2 does not perform as well as model M1. Figure 12 discusses the ROUGE score comparison against models without clusters and with clusters for different parameters.. Further, we evaluate the proposed post-processing model (M2) on the Opinosis dataset, which contains the repetitive sentences in the documents. As shown in Table 9 and Figure 13, it is clear that the proposed post-processing model (M2) provides better results compared to model M1 (without post-processing). All three ROUGE scores improve for model M1 over model (M1). The reason that M2 does not perform as well as model M1 with the "WikiHow" dataset is the characteristics of the dataset. Documents in the WikiHow dataset comprise the instruction-type sentences related to the topic and the instructions have a low probability to be repeated in the later part of a document. So, during the clustering process, for a sufficient number of documents, sentences either fall under the same clusters or just become dangling nodes, which leads to the poor quality of the summary. From the graphs, it is clear that, as the length of the document increases, the imbalance in clustering also increases, which can be reflected by the differences between the results of the two models for classes C4 and C5.

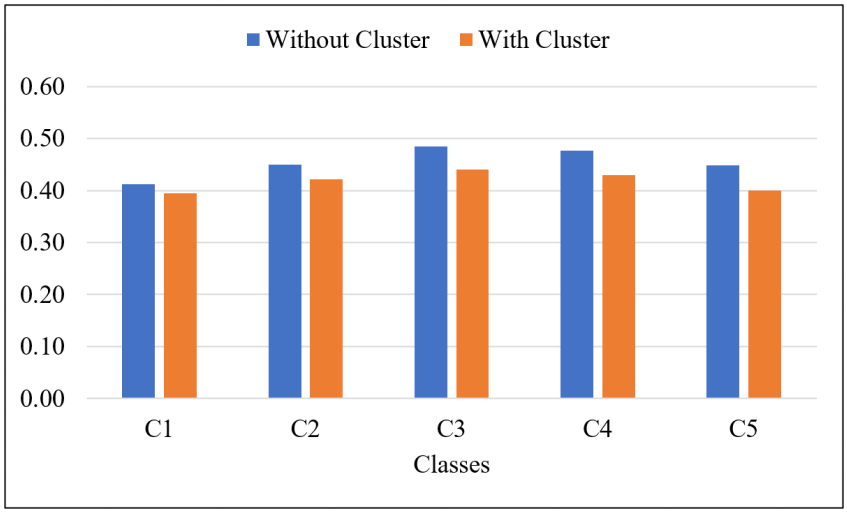

Figure 9. ROUGE-1 score comparison between models M1 and M2 for WikiHow dataset.

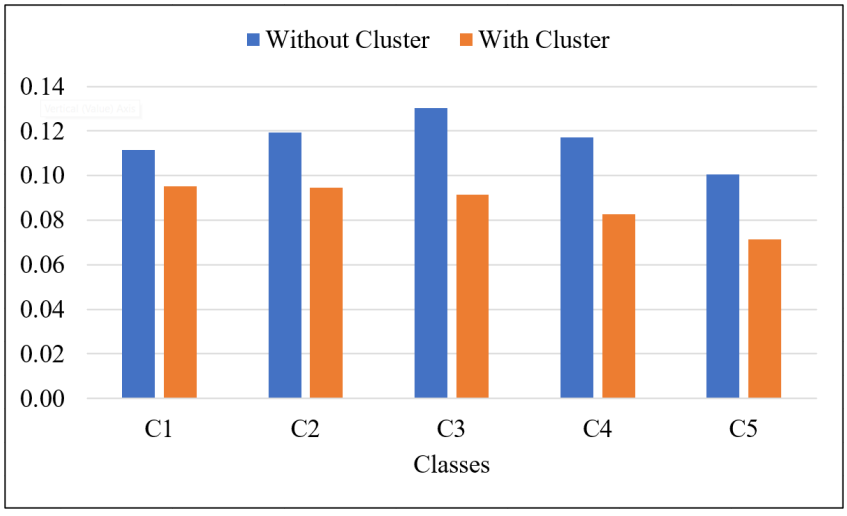

Figure 10. ROUGE-2 score comparison between models M1 and M2 for WikiHow dataset.

Table 8. Recall-based ROUGE score with/without clustering and BERT model [89] for WikiHow dataset.

	ROUGE-1	ROUGE-2	ROUGE-L
Without Clustering (M1)	0.4542	0.1158	0.3647
With Clustering (M2)	0.4172	0.0871	0.3396
BERTEXT [89]	0.3031	0.0871	0.2824
BERTEXT + 3gram-Blocking [89]	0.3037	0.0845	0. 2828
BERTEXT + 4gram-Blocking [89]	0.3040	0.0867	0.2832
MATCHSUM (BERT-base) [89]	0.3185	0.0898	0.2958

Table 9. Recall-based ROUGE score with/without clustering for Opinosis dataset.

	Without Clustering	With Clustering	Improvement %
ROUGE-1	0.2225	0.2622	4%
ROUGE-2	0.0429	0.0376	−1%
ROUGE-L	0.1954	0.2216	3%

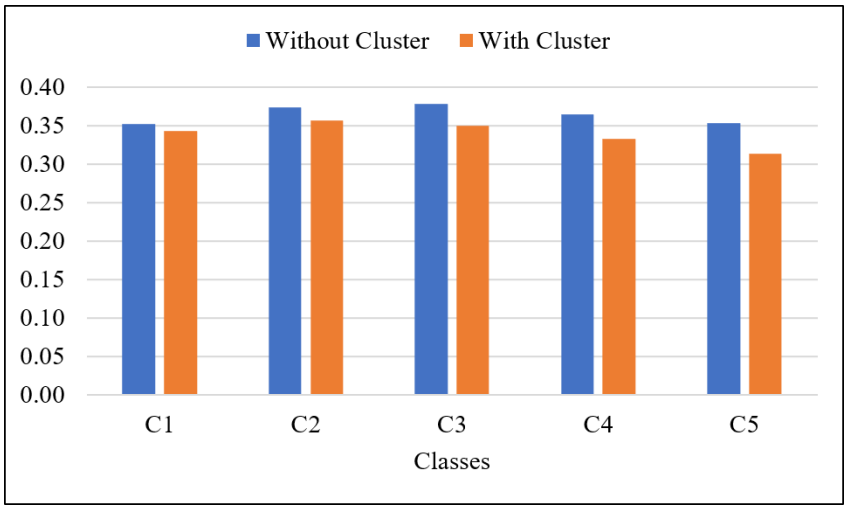

Figure 11. ROUGE-L score comparison between models M1 and M2 for wikiHow dataset.

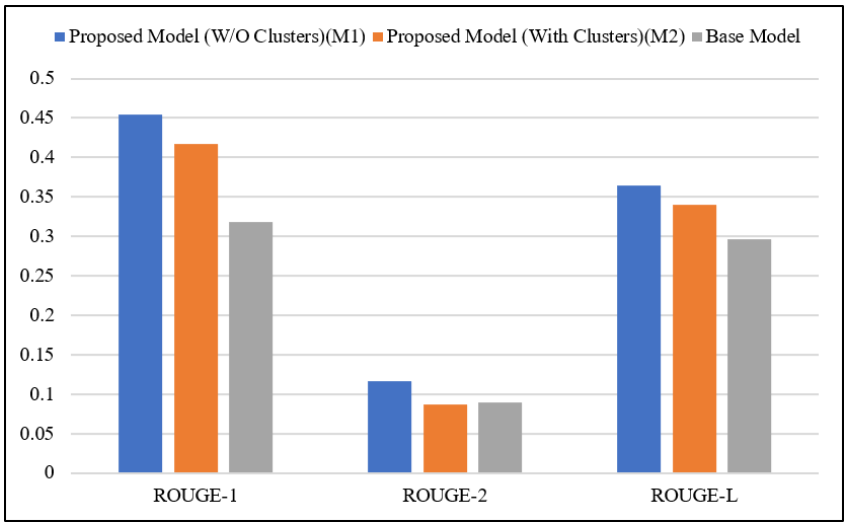

Figure 12. ROUGE score comparison between model proposed scheme (average of all five clusters) and base model [89].

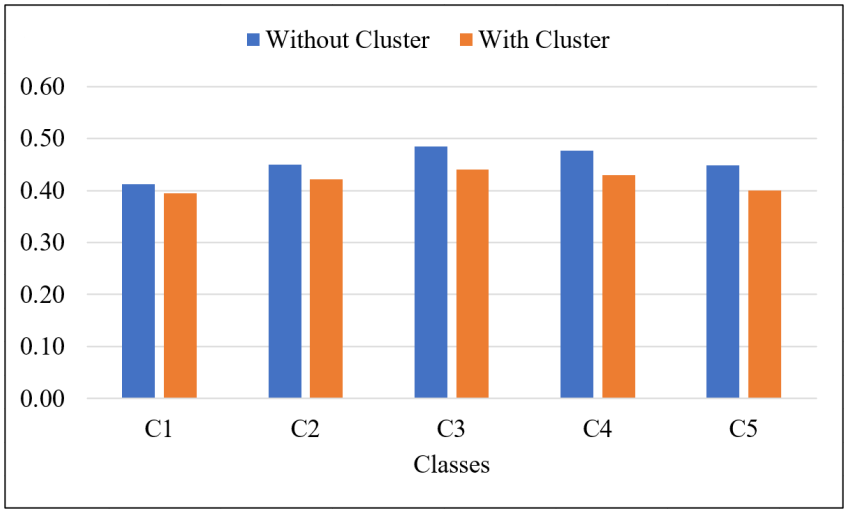

Figure 13. ROUGE score comparison between models M1 and M2 for Opinosis dataset.

Text summarization and sentiment analysis are highly correlated processes. The reason behind the high correlation is that one model can be used to improve or optimize another model. The outcome of the experimental analysis presented may be used to improvise businesses on e-commerce portals analyzing target customer reviews and feedback. The same model can be applied to opinion and sentiment analysis for large-scale crowdsourcing for launching a new product, service, or policy by the government or an enterprise. It can be applied to recommendation systems (RSs) for e-commerce businesses because the RS for many e-commerce and social media sites mainly considers the statistical datasets for predictive and descriptive models. The proposed scheme provides a new direction where text data can be used more effectively for these systems.

6.2. The Way Forward

The proposed framework paves the way toward the design of a hybrid framework, where both abstractive and extractive summarization can be unified. The motivation for the integration of both schemes is to address the dual issue of context richness provided by ETS and concise, human-like summaries generated by ATS. To achieve this, the framework should map out the semantic network information in a coherent manner. To address this, deep neural architectures, such as BERT and generative pretrained Transformers (GPT) [90], could be instrumental. These architectures are integrated with the self-attention mechanism [91], which allows the model to learn about contextual relationships between words in the text and the semantics of the document. In addition, the encoder–decoder models, like Seq2Seq [92], can be integrated with attention models to aid in the translation of extracted semantics (from ETS) to ATS.

BERT variants such as Robustly optimized BERT approach (RoBERTa) [93], A Lite BERT (ALBERT) [94], and DistilBERT [95] employ a bidirectional Transformer to understand the deep understanding of the context. RoBERTa improves the BERT model by inclusion of mode training data and optimization of the BERT process. ALBERT reduces the parametric performance loss of BERT, which makes it an effective fit for large-scale big data applications. DistilBERT is designed for minimized model sizes, with comparable performance to BERT, suitable for low-powered and constrained applications such as IoT.

GPT, on the other hand (successors GPT-2 [96], GPT-3 [97], and GPT-4 [98]), is based on Transformer architecture and is useful in the abstractive part of the framework. The architecture includes the unidirectional or casual self-attention mechanism. It can generate highly fluent and coherent text. GPT-2 is trained with 1.5 billion parameters and uses a 48-layer Transformer architecture. GPT-3 has 175 billion parameters, with 137 Transformer layers, and GPT-4 has 100 trillion parameters, with 175 layers. The increase in parameters allows the Transformer model to learn long-range dependencies better, which generates

more realistic and coherent text. The optimization algorithm of GPT models has improved, which makes it more real-time and effective, with more layers.

The hybrid framework could significantly benefit the realm of big data, specifically in the context of social media analytics. The vast amount of user-generated content on social media requires extractive as well as realistic summarization, and thus the hybrid framework could serve as a potent tool in this direction. It can leverage contextual sentiment analysis, information extraction, and trend summarization in a scalable manner. A notable example of the hybrid framework is the application of big data e-commerce platforms for product recommendation engines, where the framework could identify key features and descriptions of a product, which should resonate with customers based on demographics and relevant information like gender, age, previous selection choices of buyer, and popular opinion choices extracted from social communities.

7. Conclusions and Future Work

In recent years, e-commerce web portals have exponentially increased the amount of textual data over the internet in the form of reviews and comments about a product or service. This paper proposes a graph-based sentence scoring scheme for sentence selection to represent the summary of a given text. As an extension of the proposed scheme, graph-based clustering is applied as a post-processing step for the selection of representative sentences as a required summary of a given dataset. It shows that the proposed schemes provide better results than a BERT-based extractive text summarization model. The results show that the graph clustering process is not an ideal selection as a post-processing step for all types of datasets. The datasets with repetitive sentences benefit more from the proposed post-processing model. It can apply to applications where the impact of user reviews and comments may be analyzed.

As an extension of the work, the authors intend to propose a hybrid scheme of extractive and abstractive text summarization, where the designed model would be capable to handle multi-modal data summarization, where the hybrid framework would be designed as a staged framework. Initially, the extractive model would be trained on salient features of the input, and then the abstractive model would generate concise summaries. The architecture would be Transformer-based with a self-attention supervised training process over a large corpus of text documents. For visual and sequential data, convolutional neural networks and long short-term memory would be combined. This extension would have significant implications in digital content generation and social analytics, where the end-to-end framework would address a wide range of challenges with intelligent solutions.

Author Contributions: Conceptualization: J.P.V., S.B. and M.B.; writing—original draft preparation: J.P.V., S.B. and M.B.; methodology: P.B., A.B. and S.C.; writing—review and editing: P.B., A.B. and S.C.; Software: J.P.V., S.B., P.B., A.B., J.W. and A.M.; Visualization: J.W. and A.M.; Investigation: A.B., S.C., J.W. and A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No data are associated with this research work.

Acknowledgments: Authors would like to thank Kuwait College of Science and Technology (KCST) for supporting this work.

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

References

1. Essameldin, R.; Ismail, A.A.; Darwish, S.M. An Opinion Mining Approach to Handle Perspectivism and Ambiguity: Moving Toward Neutrosophic Logic. *IEEE Access* **2022**, *10*, 63314–63328. [CrossRef]
2. Elahi, S.; Nika, A.; Tekin, C. Online Context-Aware Task Assignment in Mobile Crowdsourcing via Adaptive Discretization. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 305–320. [CrossRef]
3. Hassani, H.; Beneki, C.; Unger, S.; Mazinani, M.T.; Yeganegi, M.R. Text Mining in Big Data Analytics. *Big Data Cogn. Comput.* **2020**, *4*, 1. [CrossRef]

4. Miah, S.J.; Vu, H.Q.; Alahakoon, D. A social media analytics perspective for human-oriented smart city planning and management. *J. Assoc. Inf. Sci. Technol.* **2022**, *73*, 119–135. [CrossRef]
5. Bhattacharya, P.; Patel, S.B.; Gupta, R.; Tanwar, S.; Rodrigues, J.J.P.C. SaTYa: Trusted Bi-LSTM-Based Fake News Classification Scheme for Smart Community. *IEEE Trans. Comput. Soc. Syst.* **2022**, *9*, 1758–1767. [CrossRef]
6. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
7. Schouten, K.; Frasincar, F. Survey on Aspect-Level Sentiment Analysis. *IEEE Trans. Knowl. Data Eng.* **2015**, *28*, 813–830. [CrossRef]
8. Bhattacharya, P.; Trivedi, C.; Obaidat, M.S.; Patel, K.; Tanwar, S.; Hsiao, K.F. BeHAutH: A KNN-Based Classification Scheme for Behavior-Based Authentication in Web 3.0. In Proceedings of the 2022 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Dalian, China, 17–19 October 2022; pp. 1–5. [CrossRef]
9. Mansour, S. Social Media Analysis of User's Responses to Terrorism Using Sentiment Analysis and Text Mining. *Procedia Comput. Sci.* **2018**, *140*, 95–103. [CrossRef]
10. AL-Khassawneh, Y.A.; Hanandeh, E.S. Extractive Arabic Text Summarization-Graph-Based Approach. *Electronics* **2023**, *12*, 437. [CrossRef]
11. Novgorodov, S.; Guy, I.; Elad, G.; Radinsky, K. Descriptions from the Customers: Comparative Analysis of Review-Based Product Description Generation Methods. *ACM Trans. Internet Technol.* **2020**, *20*, 44. [CrossRef]
12. Mackey, A.; Cuevas, I. Automatic text summarization within big data frameworks. *J. Comput. Sci. Coll.* **2018**, *33*, 26–32.
13. Ghodratnama, S.; Beheshti, A.; Zakershahrak, M.; Sobhanmanesh, F. Extractive Document Summarization Based on Dynamic Feature Space Mapping. *IEEE Access* **2020**, *8*, 139084–139095. [CrossRef]
14. El-Kassas, W.S.; Salama, C.R.; Rafea, A.A.; Mohamed, H.K. Automatic text summarization: A comprehensive survey. *Expert Syst. Appl.* **2020**, *165*, 113679. [CrossRef]
15. Chaves, A.; Kesiku, C.; Garcia-Zapirain, B. Automatic Text Summarization of Biomedical Text Data: A Systematic Review. *Information* **2022**, *13*, 393. [CrossRef]
16. Deng, Z.; Ma, F.; Lan, R.; Huang, W.; Lu, X. A Two-stage Chinese text summarization algorithm usingkeyword information and adversarial learning. *Neurocomputing* **2020**, *425*, 117–126. [CrossRef]
17. Giarelis, N.; Mastrokostas, C.; Karacapilidis, N. Abstractive vs. Extractive Summarization: An Experimental Review. *Appl. Sci.* **2023**, *13*, 7620. [CrossRef]
18. Alomari, A.; Idris, N.; Sabri, A.Q.M.; Alsmadi, I. Deep reinforcement and transfer learning for abstractive text summarization: A review. *Comput. Speech Lang.* **2022**, *71*, 101276. [CrossRef]
19. Dave, N.; Mistry, H.; Verma, J.P. Text data analysis: Computer aided automated assessment system. In Proceedings of the 2017 3rd International Conference on Computational Intelligence Communication Technology (CICT), Ghaziabad, India, 9–10 February 2017; pp. 1–4. [CrossRef]
20. Jigneshkumar Patel, H.; Prakash Verma, J.; Patel, A. Unsupervised Learning-Based Sentiment Analysis with Reviewer's Emotion. In Proceedings of the Evolving Technologies for Computing, Communication and Smart World, Singapore, 26 November 2020; pp. 69–81.
21. Zaeem, R.N.; German, R.L.; Barber, K.S. PrivacyCheck: Automatic Summarization of Privacy Policies Using Data Mining. *ACM Trans. Internet Technol.* **2018**, *18*, 1–18. [CrossRef]
22. Cai, M. Natural language processing for urban research: A systematic review. *Heliyon* **2021**, *7*, e06322. [CrossRef]
23. Verma, J.P.; Patel, B.; Patel, A. Web Mining: Opinion and Feedback Analysis for Educational Institutions. *Int. J. Comput. Appl.* **2013**, *84*, 17–22.
24. Priyadarshana, Y.H.P.P.; Ranathunga, L. Verb Sentiment Scoring: A Novel Approach for Sentiment Analysis Based on Adjective-Verb-Adverb Combinations. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Sydney, Australia, 31 July–3 August 2017; pp. 533–540.
25. Arya, C.; Diwakar, M.; Singh, P.; Singh, V.; Kadry, S.; Kim, J. Multi-Document News Web Page Summarization Using Content Extraction and Lexical Chain Based Key Phrase Extraction. *Mathematics* **2023**, *11*, 1762. [CrossRef]
26. Bichi, A.A.; Keikhosrokiani, P.; Hassan, R.; Almekhlafi, K. Graph-based extractive text summarization models: A systematic review. *J. Inf. Technol. Manag.* **2022**, *14*, 184–202.
27. Srivastava, R.; Singh, P.; Rana, K.; Kumar, V. A topic modeled unsupervised approach to single document extractive text summarization. *Knowl.-Based Syst.* **2022**, *246*, 108636. [CrossRef]
28. Huang, J.; Wu, W.; Li, J.; Wang, S. Text Summarization Method Based on Gated Attention Graph Neural Network. *Sensors* **2023**, *23*, 1654. [CrossRef]
29. Hernandez-Castaneda, A.; García-Hernández, R.A.; Ledeneva, Y.; Millan-Hernández, C.E. Extractive Automatic Text Summarization Based on Lexical-Semantic Keywords. *IEEE Access* **2020**, *8*, 49896–49907. [CrossRef]
30. Jayashree, R.; Vinay, S. A Jaccards Similarity Score Based Methodology for Kannada Text Document Summarization. In Proceedings of the 2020 International Conference on Advances in Computing, Communication & Materials (ICACCM), Dehradun, India, 21–22 August 2020; pp. 8–11. [CrossRef]
31. Bidoki, M.; Moosavi, M.R.; Fakhrahmad, M. A semantic approach to extractive multi-document summarization: Applying sentence expansion for tuning of conceptual densities. *Inf. Process. Manag.* **2020**, *57*, 102341. [CrossRef]

32. Tkachuk, A. Robustness of rank minimization heuristics for form-finding of tensegrity structures. *Comput. Struct.* **2022**, *266*, 106786. [CrossRef]
33. Fatima, Z.; Zardari, S.; Fahim, M.; Andleeb Siddiqui, M.; Ibrahim, A.A.A.; Nisar, K.; Naz, L.F. A novel approach for semantic extractive text summarization. *Appl. Sci.* **2022**, *12*, 4479. [CrossRef]
34. Yan, D.; Huang, Y.; Liu, M.; Chen, H.; Cheng, J.; Wu, H.; Zhang, C. Graphd: Distributed vertex-centric graph processing beyond the memory limit. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *29*, 99–114. [CrossRef]
35. Stewart, E. (Blog) Facebook's Fake Accounts Problem Seems Bad. Available online: Vox.com (accessed on 3 December 2020).
36. Bhargava, R.; Sharma, G.; Sharma, Y. Deep Text Summarization using Generative Adversarial Networks in Indian Languages. *Procedia Comput. Sci.* **2020**, *167*, 147–153. [CrossRef]
37. Fang, C.; Mu, D.; Deng, Z.; Wu, Z. Word-sentence co-ranking for automatic extractive text summarization. *Expert Syst. Appl.* **2017**, *72*, 189–195. [CrossRef]
38. 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]
39. Kleinberg, J.M. Authoritative Sources in a Hyperlinked Environment. *J. ACM* **1999**, *46*, 604–632. [CrossRef]
40. Brin, S.; Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.* **1998**, *30*, 107–117. [CrossRef]
41. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 16–25 July 2004; pp. 404–411.
42. Talpur, B.A.; O'Sullivan, D. Multi-class imbalance in text classification: A feature engineering approach to detect cyberbullying in twitter. *Informatics* **2020**, *7*, 52. [CrossRef]
43. Margaris, D.; Vassilakis, C. Exploiting rating abstention intervals for addressing concept drift in social network recommender systems. *Informatics* **2018**, *5*, 21. [CrossRef]
44. Moradi, M.; Dashti, M.; Samwald, M. Summarization of biomedical articles using domain-specific word embeddings and graph ranking. *J. Biomed. Inform.* **2020**, *107*, 103452. [CrossRef]
45. Alzuhair, A.; Al-Dhelaan, M. An Approach for Combining Multiple Weighting Schemes and Ranking Methods in Graph-Based Multi-Document Summarization. *IEEE Access* **2019**, *7*, 120375–120386. [CrossRef]
46. Yang, K.; Al-Sabahi, K.; Xiang, Y.; Zhang, Z. An Integrated Graph Model for Document Summarization. *Information* **2018**, *9*, 232. [CrossRef]
47. Bhargava, R.; Sharma, Y.; Sharma, G. ATSSI: Abstractive Text Summarization using Sentiment Infusion. *Procedia Comput. Sci.* **2016**, *89*, 404–411. [CrossRef]
48. Mao, X.; Yang, H.; Huang, S.; Liu, Y.; Li, R. Extractive summarization using supervised and unsupervised learning. *Expert Syst. Appl.* **2019**, *133*, 173–181. [CrossRef]
49. Luhn, H.P. The automatic creation of literature abstracts. *IBM J. Res. Dev.* **1958**, *2*, 159–165. [CrossRef]
50. Edmundson, H.P. New methods in automatic extracting. *J. ACM* **1969**, *16*, 264–285. [CrossRef]
51. Aone, C.; Okurowski, M.E.; Gorlinsky, J. Trainable, scalable summarization using robust NLP and machine learning. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montreal, QC, Canada, 10–14 August 1998; Volume 1, pp. 62–66. [CrossRef]
52. Etaiwi, W.; Awajan, A. SemG-TS: Abstractive Arabic Text Summarization Using Semantic Graph Embedding. *Mathematics* **2022**, *10*, 3225. [CrossRef]
53. Huang, Y.; Sun, L.; Han, C.; Guo, J. A High-Precision Two-Stage Legal Judgment Summarization. *Mathematics* **2023**, *11*, 1320. [CrossRef]
54. Patel, D.; Shah, S.; Chhinkaniwala, H. Fuzzy logic based multi document summarization with improved sentence scoring and redundancy removal technique. *Expert Syst. Appl.* **2019**, *134*, 167–177. [CrossRef]
55. Van Lierde, H.; Chow, T.W. Query-oriented text summarization based on hypergraph transversals. *Inf. Process. Manag.* **2019**, *56*, 1317–1338. [CrossRef]
56. Jindal, S.G.; Kaur, A. Automatic Keyword and Sentence-Based Text Summarization for Software Bug Reports. *IEEE Access* **2020**, *8*, 65352–65370. [CrossRef]
57. Du, Y.; Huo, H. News Text Summarization Based on Multi-Feature and Fuzzy Logic. *IEEE Access* **2020**, *8*, 140261–140272. [CrossRef]
58. Moradi, M. CIBS: A biomedical text summarizer using topic-based sentence clustering. *J. Biomed. Inform.* **2018**, *88*, 53–61. [CrossRef]
59. Bhargava, R.; Sharma, Y. Deep Extractive Text Summarization. *Procedia Comput. Sci.* **2020**, *167*, 138–146. [CrossRef]
60. Anand, D.; Wagh, R. Effective Deep Learning Approaches for Summarization of Legal Texts. *J. King Saud Univ. Comput. Inf. Sci.* **2019**, *34*, 2141–2150. [CrossRef]
61. Alami, N.; Meknassi, M.; En-nahnahi, N. Enhancing unsupervised neural networks-based text summarization with word embedding and ensemble learning. *Expert Syst. Appl.* **2019**, *123*, 195–211. [CrossRef]
62. Azadani, M.N.; Ghadiri, N.; Davoodijam, E. Graph-based biomedical text summarization: An itemset mining and sentence clustering approach. *J. Biomed. Inform.* **2018**, *84*, 42–58. [CrossRef]

63. Liang, Z.; Du, J.; Li, C. Abstractive Social Media Text Summarization using Selective Reinforced Seq2Seq Attention Model. *Neurocomputing* **2020**, *410*, 432–440. [CrossRef]
64. Adelia, R.; Suyanto, S.; Wisesty, U.N. Indonesian Abstractive Text Summarization Using Bidirectional Gated Recurrent Unit. *Procedia Comput. Sci.* **2019**, *157*, 581–588. [CrossRef]
65. Moirangthem, D.S.; Lee, M. Abstractive summarization of long texts by representing multiple compositionalities with temporal hierarchical pointer generator network. *Neural Netw.* **2020**, *124*, 1–11. [CrossRef]
66. Guo, Q.; Huang, J.; Xiong, N.; Wang, P. MS-Pointer Network: Abstractive Text Summary Based on Multi-Head Self-Attention. *IEEE Access* **2019**, *7*, 138603–138613. [CrossRef]
67. Cagliero, L.; Garza, P.; Baralis, E. ELSA: A Multilingual Document Summarization Algorithm Based on Frequent Item-sets and Latent Semantic Analysis. *ACM Trans. Inf. Syst.* **2019**, *37*, 1–33. [CrossRef]
68. Rouane, O.; Belhadef, H.; Bouakkaz, M. Combine clustering and frequent itemset mining to enhance biomedical text summarization. *Expert Syst. Appl.* **2019**, *135*, 362–373. [CrossRef]
69. Tsai, C.F.; Chen, K.; Hu, Y.H.; Chen, W.K. Improving text summarization of online hotel reviews with review helpfulness and sentiment. *Tour. Manag.* **2020**, *80*, 104122. [CrossRef]
70. Mohd, M.; Jan, R.; Shah, M. Text Document Summarization using Word Embedding. *Expert Syst. Appl.* **2020**, *143*, 112958. [CrossRef]
71. Cao, B.; Wu, J.; Wang, S.; Gao, H.; Fan, J.; Deng, S.; Yin, J.; Liu, X. Unsupervised Derivation of Keyword Summary for Short Texts. *ACM Trans. Internet Technol.* **2021**, *21*, 1–23. [CrossRef]
72. Ameur, M.S.H.; Belkebir, R.; Guessoum, A.A. Robust Arabic Text Categorization by Combining Convolutional and Recurrent Neural Networks. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2020**, *19*, 1–16. [CrossRef]
73. Zhou, Q.; Yang, N.; Wei, F.; Huang, S.; Zhou, M.; Zhao, T. A Joint Sentence Scoring and Selection Framework for Neural Extractive Document Summarization. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 671–681. [CrossRef]
74. Abdel-Salam, S.; Rafea, A. Performance study on extractive text summarization using BERT models. *Information* **2022**, *13*, 67. [CrossRef]
75. Koupaee, M.; Wang, W.Y. WikiHow: A Large Scale Text Summarization Dataset. *arXiv* **2018**, arXiv:1810.09305.
76. Ganesan, K.; Zhai, C.; Han, J. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In Proceedings of the 23rd International Conference on Computational Linguistics, Beijing, China, 23–27 August 2010; pp. 340–348.
77. Jalil, Z.; Nasir, M.; Alazab, M.; Nasir, J.; Amjad, T.; Alqammaz, A. Grapharizer: A Graph-Based Technique for Extractive Multi-Document Summarization. *Electronics* **2023**, *12*, 1895. [CrossRef]
78. 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.
79. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
80. Landauer, T.K.; Foltz, P.W.; Laham, D. An introduction to latent semantic analysis. *Discourse Process.* **1998**, *25*, 259–284. [CrossRef]
81. Nenkova, A.; Passonneau, R.J. Evaluating content selection in summarization: The pyramid method. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: Hlt-Naacl 2004, Boston, MA, USA, 2–7 May 2004; pp. 145–152.
82. Steinberger, J.; Jezek, K. Evaluation measures for text summarization. *Comput. Inform.* **2009**, *28*, 251–275.
83. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. Bertscore: Evaluating text generation with bert. *arXiv* **2019**, arXiv:1904.09675.
84. Sellam, T.; Das, D.; Parikh, A.P. BLEURT: Learning robust metrics for text generation. *arXiv* **2020**, arXiv:2004.04696.
85. Takano, Y.; Iijima, Y.; Kobayashi, K.; Sakuta, H.; Sakaji, H.; Kohana, M.; Kobayashi, A. Improving Document Similarity Calculation Using Cosine-Similarity Graphs. In *Advanced Information Networking and Applications*; Barolli, L., Takizawa, M., Xhafa, F., Enokido, T., Eds.; Springer: Cham, Switzerland, 2020; pp. 512–522.
86. Kryściński, W.; Keskar, N.S.; McCann, B.; Xiong, C.; Socher, R. Neural text summarization: A critical evaluation. *arXiv* **2019**, arXiv:1908.08960.
87. Yavuz, S.; Chiu, C.C.; Nguyen, P.; Wu, Y. CaLcs: Continuously Approximating Longest Common Subsequence for Sequence Level Optimization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 3708–3718. [CrossRef]
88. Plaza, L.; Daaz, A.; Gervas, P. A semantic graph-based approach to biomedical summarisation. *Artif. Intell. Med.* **2011**, *53*, 1–14. [CrossRef]
89. Zhong, M.; Liu, P.; Chen, Y.; Wang, D.; Xuanjing Huang, X.Q. Extractive Summarization as Text Matching. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, online, July 2020; pp. 6197–6208.
90. Zhu, Q.; Luo, J. Generative Pre-Trained Transformer for Design Concept Generation: An Exploration. *Proc. Des. Soc.* **2022**, *2*, 1825–1834. [CrossRef]
91. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
92. Zhang, Y.; Xiao, W. Keyphrase Generation Based on Deep Seq2seq Model. *IEEE Access* **2018**, *6*, 46047–46057. [CrossRef]

93. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
94. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2019**, arXiv:1909.11942.
95. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2020**, arXiv:1910.01108.
96. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
97. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
98. Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y.T.; Li, Y.; Lundberg, S.; et al. Sparks of Artificial General Intelligence: Early experiments with GPT-4. *arXiv* **2023**, arXiv:2303.12712.